Computational Aspects of Preference Aggregation

Vincent Conitzer

CMU-CS-06-145
July 2006

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA

Thesis committee:
Tuomas Sandholm, Chair
Avrim Blum
Tom Mitchell
Craig Boutilier (University of Toronto)
Christos Papadimitriou (University of California, Berkeley)

Submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

This research was sponsored by the National Science Foundation under grant nos. IIS-0234694, IIS-0427858, IIS-0234695, and IIS-0121678, as well as a Sloan Fellowship awarded to Tuomas Sandholm, and an IBM Ph.D. Fellowship. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Artificial Intelligence, Multiagent Systems, Electronic Commerce, Game Theory, Mechanism Design, Auctions and Exchanges, Voting.

Abstract

In a *preference aggregation setting*, a group of agents must jointly make a decision, based on the individual agents' privately known preferences. To do so, the agents need some protocol that will elicit this information from them, and make the decision. Examples include voting protocols, auctions, and exchanges. *Mechanism design* is the study of designing preference aggregation protocols in such a way that they work well in the face of strategic (self-interested) agents. In most real-world settings, mechanism design is confronted with various computational issues. One is the complexity of *executing* the mechanism. Particularly in *expressive* preference aggregation settings (such as combinatorial auctions), many mechanisms become hard to execute. Another is the complexity of *designing* the mechanism. When general mechanisms do not apply to, or are suboptimal for, the setting at hand, a custom mechanism needs to be designed for it, which is a nontrivial problem that is best solved by computer (*automated mechanism design*). Finally, there is the complexity of *participating* in the mechanism. In complex settings, agents with limited computational capabilities (*bounded agents*) will not necessarily be able to act optimally, which should be taken into account in the mechanism design process.

My thesis statement is that **we can employ the study of computational aspects of the mechanism design process to significantly improve the generated mechanisms in a hierarchy of ways**, leading to better outcomes (and a more efficient process). The dissertation outlines this hierarchy, and illustrates and addresses representative issues at various levels of the hierarchy with new results. It also serves as a significant step towards a longer-term research goal: realizing a mechanism design approach that addresses all of these issues simultaneously, comprehensively, and optimally, in settings with real-world complexity.

Acknowledgements

> *At times our own light goes out and is rekindled by a spark from another person. Each of us has cause to think with deep gratitude of those who have lighted the flame within us.*
>
> **Albert Schweitzer**

First and foremost, I want to thank my advisor, Tuomas Sandholm. Tuomas has been a great advisor. Over the course of countless research meetings, he has taught me how to take a loose idea or result and push it to its limits by discovering variations and generalizations, implications and applications. He has also taught me to think of the big picture and present my work accordingly. In addition, Tuomas' advice has extended beyond research to teaching, career planning, and even personal issues; and he has always been willing to take extra time out of his busy schedule when important or unusual issues came up. Meanwhile, he worked behind the scenes to make sure that I did not have to worry about funding or adminstrative problems, and could focus on research. Finally, and perhaps most importantly, Tuomas has been endlessly motivating and encouraging. Tuomas brings his infective energy and enthusiasm to everything he does, and it is amazing how he manages to stay involved in so many different threads.

I also especially want to thank the other members of my committee, Avrim Blum, Craig Boutilier, Tom Mitchell, and Christos Papadimitriou, for their valuable feedback on this dissertation. Few people are fortunate enough to have such a fantastic committee.

Special thanks go to Barbara Grosz and Avi Pfeffer for sparking my initial interest in AI and putting me on the way to graduate school, allowing me to spend the last five years to think about more interesting problems than how to spend a large salary. Barbara, Avi, and David Parkes have since made sure that I continue to feel a part of Harvard's computer science department by including me in various events at conferences.

Working at CombineNet has taught me much about how my work relates and applies to practice, and I am grateful for all the interactions that I have had with the great people there, including Bryan Bailey, Egon Balas, Craig Boutilier, Michael Concordia, Andrew Fuqua, Andrew Gilpin, Sam Hoda, David Levine, Paul Martyn, Jim McKenzie, George Nemhauser, David Parkes, Rob Shields, Yuri Smirnov, Brian Smith, and Subhash Suri. I also want to thank Andrew Davenport, Jayant Kalagnanam, and everyone else that I interacted with at IBM Research for a wonderful and productive summer there. I am also very grateful for the IBM Ph.D. Fellowship that supported me this past year (as well as for all the NSF funding that funded the remainder of my studies).

Back at CMU, I want to thank Andrew Gilpin for organizing the Game Theory Discussion Group for multiple years, which has been a great forum for us to discuss and explore our research. I have also benefited greatly from many technical discussions with Kate Larson, Andrew Gilpin, Anton Likhodedov, David Abraham, Rob Shields, Benoît Hudson, Alex Nareyek, Paolo Santi, Felix Brandt, Marty Zinkevich, Rudolf Müller, Nina Balcan, Michael Benisch, Michael Bowling, Shuchi Chawla, Liz Crawford, George Davis, Jon Derryberry, Nikesh Garera, Daniel Golovin, Jason Hartline, Sam Hoda, Sham Kakade, Zhijian Lim, Daniel Neill, XiaoFeng Wang, Andrew Moore, Roni Rosenfeld, Manuela Veloso, and many others.

I am also indebted to many other researchers in distributed AI, multiagent systems, electronic commerce, and economics with whom I have had valuable discussions. While the following list is undoubtedly incomplete (not only due to my failing memory, but also due to the anonymity of the review process), special thanks go out to Alon Altman, Moshe Babaioff, Liad Blumrosen, Giro Cavallo, Raj Dash, Edith Elkind, Boi Faltings, Kobi Gal, Rica Gonen, Georg Gottlob, Amy Greenwald, Barbara Grosz, Joe Halpern, Edith Hemaspaandra, Nathanaël Hyafil, Sam Ieong, Atsushi Iwasaki, Adam Juda, Radu Jurca, Gal Kaminka, Michael Kearns, Sebastién Lahaie, Jérôme Lang, Ron Lavi, Kevin Leyton-Brown, Victor Lesser, Michael Littman, Bob McGrew, Ahuva Mu'alem, Eugene Nudelman, Naoki Ohta, David Parkes, David Pennock, Adrian Petcu, Avi Pfeffer, Ryan Porter, Ariel Procaccia, Juan Antonio Rodríguez-Aguilar, Amir Ronen, Jeff Rosenschein, Michael Rothkopf, Rahul Savani, Jeffrey Shneidman, Grant Schoenebeck, Peter Stone, Moshe Tennenholtz, Rakesh Vohra, Bernhard von Stengel, Éva Tardos, Eugene Vorobeychik, Michael Wellman, Makoto Yokoo, and Shlomo Zilberstein.

Fortunately, my years in Pittsburgh were not all work and no play. I especially want to thank Atul, Kevin, Dave, Anand, Kristen, Lucian, Monica, Vahe, Jernej, Stefan, Bianca, Naz, Nikesh, Leo, Mugizi, and Paul for being such good friends here for many years. I will also miss everyone from Real Mellon Soccer, but am comforted to know that you are left in Aaron's capable hands (and feet).

I would not be who I am today without my mom, my dad, Jaap, Marischa, Ruben, and Jessica. I want to thank my family for their unconditional love, and am sad to always be living so far away from them. Perhaps technology will eventually make the distance less significant.

Finally, I want to thank Christina for her love and support, especially in this last hectic year. While I was working on such things as proving the #P-hardness of manipulating randomly and incrementally strategy-proofed preference aggregation mechanisms, she took care of less essential details such as food, clothing, and finding a place to live next year. More importantly, she makes my life more enjoyable. Christina, I love you and hope that you did not miss me too much while I was working on this dissertation.

Contents

Chapter 1

Introduction

There are many important settings in which multiple parties (or *agents*) must jointly make a decision—to choose one *outcome* from a space of many possible outcomes—based on the individuals' preferences (and potentially other privately held information). For example, individual persons may need to decide how to allocate various tasks and resources among themselves; firms may need to decide on how to structure the supply chain to efficiently bring a product to the consumer; the governments of nations may need to decide on what form, if any, an international treaty will take; *etc.* Such *preference aggregation settings* are pervasive in human (and perhaps even animal) life, and have long been studied, especially in economics and political science. In recent years, computer science has joined the set of fields with significant interest in preference aggregation. As computer systems become increasingly interconnected, more and more problems come to the forefront that are fundamentally about selecting good outcomes in the face of conflicting preferences. Examples include scheduling multiple users' jobs; routing network traffic for multiple users; choosing which companies' advertisements to display on a webpage; ranking the webpages of multiple authors in response to a search query; *etc.* In addition, software agents have the potential to aid humans in, or potentially even take over, some preference aggregation tasks, such as trading items over the Internet. The interest of computer scientists in preference aggregation is driven in part by such applications; however, another driver is the fact that computationally nontrivial problems must be solved in almost all complex preference aggregation settings, so that computer scientists can contribute even to settings that are not otherwise related to computer science.

To aggregate their preferences effectively, the agents need to use some *protocol* that will elicit the agents' preferences (and other pertinent information) from them, and select the outcome. There are two great challenges in designing a good protocol. First, a good protocol should, when provided with accurate information, arrive at an outcome that is considered good for the agents. Finding such a good outcome may require significant communication and computation, especially when the outcome and preference spaces are combinatorial in nature. Second, when the agents are self-interested, they will misreport their information to the protocol (also known as *manipulating*) when it is beneficial for them to do so. Thus, a good protocol should take this strategic behavior into account, and select outcomes in such a way that a good outcome is reached even though the agents are strategic (for instance, by making sure the agents have no incentive to misreport). The theory of *mechanism design* studies how to make protocols strategy-proof in this sense.

Computation plays a significant role in both the execution and the design of good protocols. This role is clearly apparent in some cases—for instance, selecting a good outcome, even with all the agents' true information in hand, may require the solution of a computationally hard optimization problem. However, computation can also have more subtle roles in the design of protocols—for instance, when anticipating the extent of an agent's strategic manipulation of the mechanism, it can be helpful to have a good assessment of the agent's computational limits in manipulating.

1.1 A hierarchy of uses for computation in preference aggregation

I propose the following hierarchy (Figure 1.1) for categorizing various needs for computational tools in preference aggregation settings. Computational tools may improve the aggregation of the agents' preferences over outcomes through:

(1) The straightforward **optimization** of the outcome (choosing the outcome that maximizes the sum of the agents' utilities, also known as the *social welfare*, or some other objective), given all the agents' preferences and other information, in potentially complex allocation or other outcome selection tasks—without consideration of strategic behavior.

(2) Enabling the use of a **mechanism** for the selection of the outcome, where the mechanism is designed so that a good outcome will be chosen in spite of strategic behavior by the agents.

(3a) Rather than enabling a known general mechanism, computing a *custom* mechanism on the fly for the setting at hand (**automated mechanism design**).

(3b) Making use of the agents' computational limits, or **bounded rationality**, in the mechanism design process to reach better outcomes.

(4) Generalizing all of the previous tools to enable **automated mechanism design for bounded agents**.

Section 1.2 introduces the nodes of the hierarchy using an example application setting. Section 1.3 discusses the meaning and use of the hierarchy itself (that is, the relationships represented by the directed edges). Section 1.4 discusses research directions in preference aggregation that are orthogonal to this hierarchy. Finally, Section 1.5 gives an outline of the remainder of the dissertation.

1.2 The hierarchy's nodes illustrated by example

In this section, I will introduce the nodes of the hierarchy using the example of a *combinatorial auction*. Combinatorial auctions have recently become a popular research topic (*e.g.* [Rothkopf *et al.*, 1998; Sandholm, 2002a; Nisan, 2000; Sandholm *et al.*, 2005c; Gonen and Lehmann, 2000; Nisan and Ronen, 2000; Lehmann *et al.*, 2002; Bartal *et al.*, 2003; Lavi *et al.*, 2003; Yokoo, 2003; Parkes,

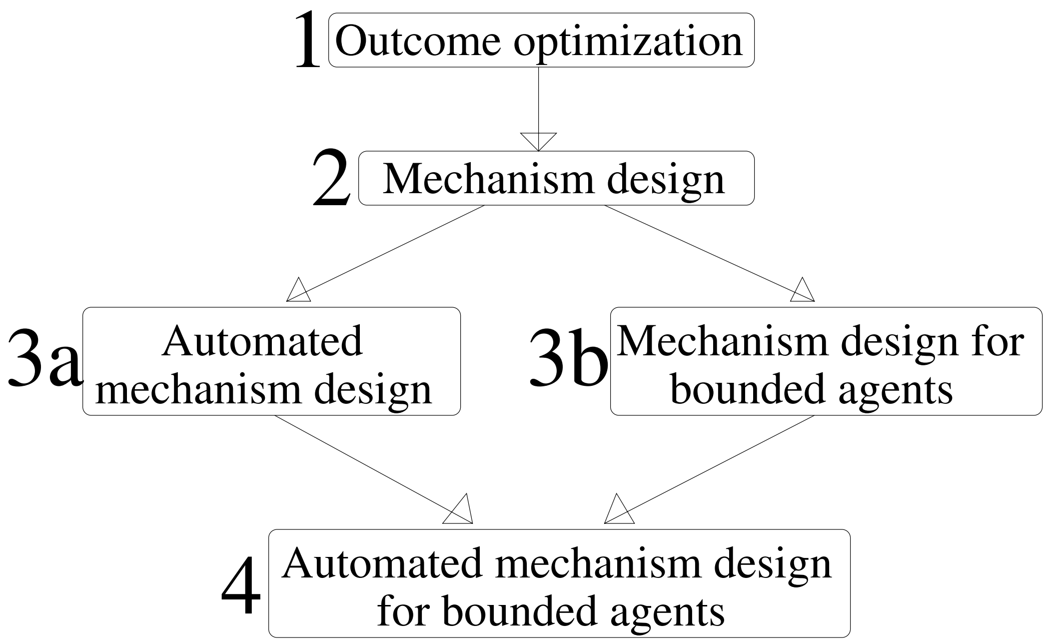


Figure 1.1: The hierarchy of uses for computational tools in preference aggregation.

1999a; Wurman and Wellman, 2000; Ausubel and Milgrom, 2002]). They are arguably the canonical example of a preference aggregation domain that is expressive enough to introduce a variety of nontrivial computational issues, and they have increasing practical importance. (In fact, some of the results to be presented in this dissertation strictly concern combinatorial auctions.) Nevertheless, these issues occur in many other important preference aggregation domains, and combinatorial auctions here only serve as an (important) illustrative example.

In a combinatorial auction, a seller has multiple items for sale, and bidders can bid on *bundles* (subsets) of items. For example, suppose Sally has an apple and an orange for sale. Al would enjoy eating the apple and places a bid of \$2 on the apple (more precisely, the bundle consisting of the apple alone). The orange by itself is worth nothing to Al. However, he would prefer having both the apple and the orange, to make a fruit salad; and so, Al bids \$3 on the bundle of the apple and the orange together. Meanwhile, Bill is only interested in the orange, and bids \$2 on the orange. Thus the bids can be represented as follows: $v_A(\{\text{apple}\}) = 2$, $v_A(\{\text{apple, orange}\}) = 3$, $v_B((\{\text{orange}\}) = 2$. (For the purposes of this example, I will assume that all bids are submitted simultaneously—a so-called *sealed-bid* auction. This contrasts with, for instance, the more commonly known *English auction* for a single item in which bidders can continue to improve their bids.) We are now ready to start introducing the nodes of the hierarchy using this example.

1.2.1 Node (1): Outcome optimization

In our example, who should win which items? Giving Al both items will generate a value of \$3, but giving the Al the apple and Bill the orange will generate a value of \$2 + \$2 = \$4. So, given the information that we have, the latter is the most sensible outcome of the auction. This problem of determining which of the bids win is known as the *winner determination* or *clearing* problem. In spite of its apparent simplicity, once the numbers of items and bids become large, this problem becomes computationally hard: it is NP-complete [Rothkopf *et al.*, 1998], even to approximate [Sandholm, 2002a].

The clearing problem is a good example of a problem at the shallowest node **(1)** in the hierarchy: the straightforward **optimization** of the outcome, without consideration of strategic behavior. At this node, we assume that we have a full characterization of each agent's preferences over the outcomes (in our example, this characterization is given by the bids), and we take this characterization at face value. That is, we do not worry about whether the estimates of the agents' preferences that we have are inaccurate (for instance, perhaps the agents reported their preferences to us directly, and they may have misrepresented them to effect a better outcome for themselves). This approach is appropriate when, for instance, we obtained our estimates of the agents' preferences from a disinterested source (not the agents themselves). The optimization algorithm may also be used as a subroutine in a mechanism that does take strategic behavior into consideration—we will discuss this in more detail later.

1.2.2 Node (2): Mechanism design

The ability to solve the clearing problem well is necessary for reaching good outcomes in combinatorial auctions, but, as we will see in this subsection, it is not sufficient. So far, we have not yet discussed what the bidders in the auction should pay. Let us suppose that the auction is a *first-price* auction, in which bidders simply pay the values of their winning bids. In first-price auctions, bidders typically do not bid their true valuations, because if they did, they would gain nothing from having their bids accepted. Rather, bidders will bid lower than their true valuations, to have a chance of making some profit in the auction. To decide how much lower they will bid, they will take into account their perceived chances of winning for each amount that they may bid. In our example, perhaps Al's true marginal value for having the orange (in addition to the apple) was much higher than \$3 - \$2 = \$1; perhaps it was \$10, but Al (in retrospect, incorrectly) thought it was extremely unlikely that Bill would place a bid higher than \$1. Meanwhile, perhaps Bill's true value for the orange was \$9, but Bill (in retrospect, correctly) thought that Al would think that bidding \$3 on the bundle of the apple and orange would in all likelihood be enough to win that bundle, and therefore Bill chose to bid only \$2. Now, even though we solved the clearing problem optimally with respect to the bids provided, a number of things still went wrong. First, from the perspective of economic efficiency, the orange *should* have ended up with Al rather than Bill, because Al's marginal value \$10 for it is greater than Bill's marginal value \$9 for it. Second, Sally received a total revenue of only \$4, whereas it would seem that there was enough competition on the orange alone for her to receive at least \$9.

These problems are the result of the auction format (a sealed-bid first-price auction). For example, an *ascending-price* auction in which bidders can continue to raise their bids may have been more successful here. (Various research has been devoted to the implementation of ascending-price or iterative auctions and exchanges in multi-item settings [Parkes, 1999a; Wurman and Wellman, 2000; Ausubel and Milgrom, 2002; Parkes *et al.*, 2005]. Such protocols are themselves special cases of a more general framework in which the auctioneer is enhanced by *elicitor software* that incrementally and adaptively elicits the bidders' preferences using specific queries until enough information has been elicited to determine the final allocation and payments[Conen and Sandholm, 2001].) An ascending-price auction, however, can run into a variety of other problems in other settings. Yet another approach may be to apply a type of *second-price* auction. In a second-price (or *Vickrey*) auction for one item, the winner pays the value of the second highest bid. (The generalization of the second-price auction to multiple items is known as the *VCG mechanism* [Vickrey, 1961; Clarke, 1971; Groves, 1973].) (Generalizations of) second-price auctions have the nice property that bidders are motivated to bid their true value for an item.

Different formats for auctions (or other preference aggregation domains) are known as *mechanisms*, and the process of constructing them is known as *mechanism design*. The mechanism specifies—for each combination of preferences (or *types*) that may be reported—what the outcome should be, as well as additional variables, such as payments to be made. Over the past several decades, economists have put together a limited library of mechanisms that motivate agents to report their preferences (*i.e.*, bid) truthfully, pursue some objective under this constraint (for instance, social welfare), and that can be applied in settings with varying levels of generality. For example, mechanisms have been discovered that, under some assumptions, will motivate agents to report truthfully in *any* preference aggregation setting, while still always choosing the socially optimal outcome. These are the VCG mechanism mentioned previously, and the dAGVA [d'Aspremont and Gérard-Varet, 1979; Arrow, 1979] mechanism.

Even when these general mechanisms are known, applying them to complex combinatorial settings (such as combinatorial auctions) is typically highly nontrivial, sometimes requiring the solution to multiple winner determination problems. For example, to apply the VCG mechanism to a setting, one typically requires the solution to (#bidders+1) outcome optimization problem instances: the original one, as well as each one in which one of the bidders is omitted from the instance. Moreover, using approximation approaches that are effective for the outcome optimization problem *per se* to execute the mechanism can destroy the strategic properties of the mechanism: agents will no longer be motivated to tell the truth [Nisan and Ronen, 2001; Sandholm, 2002b]! Thus, the agents will report their preferences strategically rather than truthfully. Moreover, there is no guarantee that the resulting strategic equilibrium will produce an outcome that is anywhere close to the socially optimal one. The resulting challenge is to design special approximation algorithms that do incent the agents to report truthfully. Or, put differently, the challenge is to design special truthful mechanisms whose outcomes are at least reasonably good, and can be computed in polynomial time. This is known as *algorithmic mechanism design* [Nisan and Ronen, 2001].

Node **(2)** in the hierarchy is concerned with the issues described in this subsection: designing mechanisms that encourage agents to tell the truth in (expressive) preference aggregation settings, as well as designing efficient algorithms for computing the outcomes of these mechanisms, thereby enabling their use.

1.2.3 Node (3a): Automated mechanism design

Depending on her goals, Sally will prefer some mechanisms to others. Which mechanism is the optimal one? To make this more concrete, suppose Sally assesses the situation as follows:

> *I want to maximize my expected revenue, but I want to cap my revenue at 6. As a secondary objective, I want to maximize Al and Bill's social welfare (combined utility), but I do not want Al's utility for the outcome to exceed Bill's by more than 3. I think Bill's utility for the orange alone is probably (80%) 9, but maybe (20%) it is 1. As for Al's utilities, ...*

Et cetera. When Sally looks into the library of general mechanisms, she will make some troubling discoveries. First of all, nobody has ever studied what to do under exactly her idiosyncratic goals (presumably because her goals lack simplicity and elegance). She will likely discover some relevant facts, though:

1. Many people have actually studied the problem of how to maximize expected revenue when selling two items, but there is still no general characterization of how to do this [Avery and Hendershott, 2000; Armstrong, 2000; Vohra, 2001].

2. When two parties are negotiating over an item, it is in general not possible to design a mechanism such that the item always ends up with the party who likes it best without money flowing in or out of the system consisting of the two parties [Myerson and Satterthwaite, 1983]. This is relevant to Sally in cases where she is clearly going to make her revenue cap, from which point on money can only flow between Al and Bill. (Alternatively, money can flow outside of the system consisting of Al, Bill, and Sally, which is not desirable.)

It may seem at this point that Sally is in over her head and should simply choose one of the known mechanisms that she thinks comes at least somewhat close to what she is looking for. But, while her situation may seem overwhelming, Sally does have a leg up on the economists studying these problems. Rather than being concerned with a general characterization for all settings in the domain of combinatorial auctions, Sally is only concerned with her own setting, which involves an apple, an orange, two buyers Al and Bill, Sally's objective, and specific prior probabilities for the buyers' preferences. If she finds the mechanism that is the best for her particular setting, she will not care whether or not (or to what extent) it generalizes to all possible settings. Still, Sally, who wants to get back to growing apples and oranges, may not be patient or qualified enough to solve even this restricted problem by hand. To address this, we introduced the **automated mechanism design** approach [Conitzer and Sandholm, 2002b], in which we solve the mechanism design problem for a given setting by computer, as an optimization problem. This is the topic of node **(3a)**: computing a *custom* mechanism on the fly for the setting at hand.

1.2.4 Node (3b): Mechanism design for bounded agents

It is the nature of mechanism design to assume that agents are strategic in how they interact with the mechanism. However, assuming fully strategic agents implies assuming perfectly rational agents. As a result, traditional mechanism design may often be too conservative in the extent to which it expects the agents to manipulate. To make this concrete, consider a mechanism designer that is considering Sally's mechanism design problem (and suppose that the setting is a little more complex, involving a larger variety of fruits). The designer's analysis may proceed as follows.

> *... In the case where Al bids \$3 on the apple, \$4 on the apple and the orange, \$1 on the banana, ... And Bill bids... Then, it would be very good for the objective I am pursuing if I can give Al the apple only, and charge him \$2.50. Unfortunately, if I did this, Al would have an incentive to misreport his preferences as follows: bid \$3.50 on the apple, \$6 on the banana and the grapefruit, \$3 on the tangerine... – because for this bid we already decided on an outcome that would be better for Al. So, I have to do something else...*

(Of course, most of this analysis would not be done explicitly, but rather implicitly in the mathematical analysis—although the optimization-based approach of automated mechanism design would be closer to this in terms of how explicit the analysis is.) This analysis would be overly conservative if Al was too computationally limited to ever discover the beneficial misreport of his preferences "\$3.50 on the apple, \$6 on the banana and the grapefruit, \$3 on the tangerine...", and thus it may leave money (more precisely, objective value) on the table. Ideally, mechanism design should take into account such computational limitations. This is the topic of node **(3b)**: making use of the agents' **bounded rationality** in the mechanism design process to reach better outcomes.

The way this plays out technically is as follows. The classical theory of mechanism design tells us that there is no benefit to using mechanisms in which the agents have incentives to make insincere revelations (that is, *non-truthful mechanisms*, as opposed to truthful mechanisms): if the agents behave strategically, then for every non-truthful mechanism, there is another truthful mechanism that performs just as well. This result is known as the *revelation principle* [Gibbard, 1973; Green and Laffont, 1977; Myerson, 1979, 1981], and the basic idea behind its proof is remarkably simple. Suppose we envelop a non-truthful mechanism with an *interface layer*, to which agents input their preferences. Then, the interface layer interacts with the original mechanism on behalf of each agent, *playing strategically in the agent's best interest* based on the reported preferences. (Compare, for example, proxy agents on eBay [eBay UK, 2004].) The resulting mechanism is truthful: an agent has no incentive to misreport to the interface layer, because the layer will play the agent's part in the original mechanism in the agent's best interest. Moreover, the final outcome of the new, truthful mechanism will be the same, because the layer will play strategically optimally—just as the agent would have. However, in complex settings, the last step in this argument may be incorrect: it is possible that the agent, due to computational limitations (bounded rationality), would not have been able to play optimally in the original, non-truthful mechanism. If this is so, then the outcome for the non-truthful mechanism would have been different—perhaps better. Thus, mechanism design for computationally bounded agents should not focus strictly on truthful mechanisms, but rather try to construct better, non-truthful mechanisms based on models of the computational boundedness of the agents. The advantage of doing so is that strictly better outcomes may be obtained than by any truthful mechanism, because (only) by using a non-truthful mechanism, we can exploit the agents' computational inability to always act in a strategically optimal way.

1.2.5 Node (4): Automated mechanism design for bounded agents

Finally, ideally, Sally would like to maintain the benefits of automated mechanism design while also making use of the agents' bounded rationality. This is certainly nontrivial, because the settings in which the agents' bounded rationality comes into play tend to be the same as those that are too complex for current automated mechanism design techniques to handle. Specifically, bounded rationality becomes relevant mostly in settings where agents can have exponentially many possible preferences (types), and automated mechanism design usually does not scale to very large numbers of types. Nevertheless, towards the end of this dissertation, we will present one approach to automated mechanism design for bounded agents.

1.3 How to use the hierarchy

Whereas the previous section introduced the individual nodes, in this section, I will focus on the meaning and intended use of the hierarchy itself.

1.3.1 Interpretation

I will first discuss how the hierarchy (that is, the relationship between the nodes represented by the directed edges in Figure 1.1) should be interpreted. Each node inherits the computational complexities typically associated with its ancestor nodes. For instance, a nontrivial mechanism may need to be created for a complex allocation task whose outcome optimization problem is already hard. The computational issues that each node introduces thus become closely intertwined with those of nodes at shallower levels. Throughout this dissertation, for clarity, I will try as much as possible to separate out the computational issues specific to the node under consideration from those of the shallower levels, but the reader should bear in mind that some of the most valuable future research will be done in settings where computational issues from multiple levels are nontrivially present simultaneously.

A related point is that in order for the hierarchy to make sense, the underlying setting (for instance, a combinatorial auction) should be assumed *fixed*—rather than assuming that the setting has a level of complexity that is of "appropriate" difficulty to the node in the hierarchy at hand. This precludes misleading arguments such as the following: "Outcome optimization is only interesting in complex combinatorial settings. Automated mechanism design, on the other hand, is nontrivial even in small, flatly represented settings, and should thus be studied there first. Therefore, automated mechanism design does not necessarily address issues associated with outcome optimization." If the setting is fixed beforehand, then either outcome optimization is trivial, and automated mechanism design may be manageable; or outcome optimization is nontrivial, and automated mechanism design is likely to be difficult for current techniques. Nevertheless, in either case, automated mechanism design must deal with the optimization issues presented by the setting.

1.3.2 Complexity of node vs. complexity of setting

Many (but not all) of the computational issues to be discussed in this thesis are in part the result of the complexity of the setting. For example, single-item auctions do not introduce many computational questions because of the simplicity of the setting, whereas structured auctions are a source of numerous difficult computational questions. The complexity of the setting is closely tied to its *representation*. Simple settings can be flatly represented (that is, all possible outcomes and preferences can be explicitly listed), whereas more complex settings require a *structured* representation. For example, in a large combinatorial auction, it is not feasible to simply list all possible allocations and read through them one by one to find the best one. Rather, we have a combinatorial description of the space of possible allocations, and require sophisticated algorithms to search through this space.

The following table shows how the complexity of each node in the hierarchy depends on whether the setting is flat (that is, all possible outcomes and preferences can be explicitly listed) or structured (that is, all possible outcomes and preferences cannot be explicitly listed, but there is a way of representing all of them in a more concise way, as in a combinatorial auction). The entries in the table without special emphasis do not introduce any (additional) questions related to computation. The entries in *italics* do introduce such questions, and the dissertation will address them in detail. The entries in **bold** are mostly beyond the scope of this dissertation and represent important agendas for future research.

Node	**Flat Settings**	**Structured Settings**
(1) Outcome optimization	trivial	*sometimes computationally hard*
(2) Mechanism design	standard setting for classical mechanism design	*need to balance computation, optimality, and incentives*
(3a) Automated mechanism design	*sometimes computationally hard*	*some approaches for special classes* **no known (computationally efficient) general approaches**
(3b) Mechanism design for bounded agents	bounded rationality is irrelevant	*design mechanisms to beneficially place computational burden on agents* **no known general theory**
(4) Automated mechanism design for bounded agents	coincides with automated mechanism design	*incrementally make mechanisms more strategy-proof* **no known general theory**

The complexity of each node vs. the complexity of the setting.

The fact that these issues are harder in structured settings than in flat settings should, of course, not be interpreted to imply that it is a bad idea to find good concise representations of settings of interest by using their inherent structure. The issues described in the table are harder for structured settings than for flat settings *of the same description length*. However, a good structured representation can significantly reduce the problem's description length (and in many cases writing the problem down is not even feasible without some structured representation). Moreover, the special structure of a setting can often be used to help solve the problem.

1.3.3 Are the shallow nodes outdated?

I should emphasize that it is *not* my opinion that research strictly focused on the shallower levels of the hierarchy (such as solving complex outcome optimization problems without taking questions of mechanism design into account) is outdated or pointless, for the following three reasons.

First, sometimes issues deeper down in the hierarchy do not apply to the setting at hand. For instance, if all agents' preferences are commonly known beforehand, there is no need for mechanism design. Or, if we need to design the mechanism ahead of time for a general domain without any knowledge of the instances that will actually arise, automated design cannot contribute much.[1] Finally, we may not want to assume that the agents are in any way computationally restricted, for instance because we know the stakes are high enough that the agents will spare no effort in finding the most beneficial manipulation. Nevertheless, it is not typical that some of these issues can be assumed away at no cost, and thus perhaps the following two reasons are more important.

The second reason is that at least in the short run, some of the research most valuable to the world will consist of finding *new domains* in preference aggregation where computer science techniques can be fruitfully applied. Some very recent examples of new domains include the following:

- Fortnow *et al.* [2003] introduce an expressive framework for trading securities that generate payoffs in certain states of the world (where the states of the world are represented by Boolean formulas).

- Porter [2004] studies mechanism design for the scheduling of different agents' jobs on a processor.

- We [Conitzer and Sandholm, 2004e] study the domain of donations to charitable causes, and introduce an expressive bidding language for arriving at complex contracts over who pays what to which charities—a generalization of so-called "matching offers". (This topic will be discussed in upcoming chapters, Sections 2.3, 3.3, and 5.2.)

New domains such as these are most naturally studied first at the shallower levels of the hierarchy, and later deeper down in the hierarchy. Attempting to immediately study the new domain in the deepest levels of the hierarchy may leave important issues at shallower levels underinvestigated.

Third, better solutions for problems shallower up in the hierarchy can have significant implications for problems deeper down in the hierarchy. For instance, new algorithms for solving outcome optimization problems (such as clearing a combinatorial auction) to optimality may reduce the need for designing mechanisms that implement easily computable approximations. (For example, the VCG mechanism is always sufficient to implement the social welfare maximizing outcome, if optimal outcomes can be computed.) Also, deriving (possibly partial) general characterizations of mechanisms with desirable properties may help automated mechanism design, by reducing the search space to the set of mechanisms consistent with the characterization.

Thus, there is significant motivation for studying problems strictly at the shallower levels of the hierarchy (especially in new domains); but eventually, as this research matures, the benefits of additional research in the shallower levels will decrease, and we should shift to deeper levels of the hierarchy to create greater value for the world.

[1] Though perhaps it could still be used by running it on some random instances in the domain, and attempting to extract general principles from the mechanisms generated in these experiments.

1.4 Orthogonal research directions

While the hierarchy is useful in classifying much of the research on computational aspects of preference aggregation, there are also research directions that are orthogonal to the topics in the hierarchy. Perhaps the most important such direction is that of *preference elicitation*. In preference elicitation, the idea is to not have each agent reveal its entire preferences in one step, but rather to selectively and incrementally query the agents for aspects of their utility functions, in the hope that the outcome can be determined while having the agents reveal only a fraction of their preference information. There are several benefits to reducing the amount of preferences that the agents need to reveal. First, determining one's utility for any specific outcome can be computationally demanding [Sandholm, 1993a, 2000; Parkes, 1999b; Larson and Sandholm, 2001b]. Second, in many settings, the utility functions of agents are exponentially sized objects that are impractical to communicate. Finally, agents may prefer not to reveal their utility information for reasons of privacy or long-term competitiveness [Rothkopf *et al.*, 1990].

As an example, various *ascending* combinatorial auction protocols have been proposed [Parkes, 1999a; Wurman and Wellman, 2000; Ausubel and Milgrom, 2002; de Vries *et al.*, 2003]. Such auctions maintain current prices for the items/bundles, which allows bidders to focus their bidding efforts on bundles on which they are competitive. Conen and Sandholm [2001] have proposed a more general framework where the auctioneer is enhanced by elicitor software that incrementally and adaptively elicits the bidders' preferences using specific queries until enough information has been elicited to determine the final allocation and payments. This framework was experimentally evaluated for general combinatorial auctions by Hudson and Sandholm [2004] (building on work by Conen and Sandholm [2002]), showing that the amount of preference information that needs to be elicited is only a small fraction of the total amount of preference information. Another direction of interest with close ties to computational learning theory is the identification of classes of preferences for which elicitation requires only a polynomial number of queries [Zinkevich *et al.*, 2003; Blum *et al.*, 2004; Lahaie and Parkes, 2004; Santi *et al.*, 2004; Conitzer *et al.*, 2005]. Negative results have also been obtained: for instance, [Nisan and Segal, 2005] shows that in general, obtaining a better approximation than that generated by auctioning off all objects as a bundle requires the exchange of an exponential amount of information.

Preference elicitation is a direction that is orthogonal to the hierarchy because preference elicitation can be separately studied at each node of the hierarchy. For example, one may simply wish to elicit enough information to compute an optimal allocation with respect to revealed preferences (node **(1)**); alternatively, one may wish to elicit enough information to choose outcomes in a manner that discourages untruthful bidding [Conen and Sandholm, 2001; Hyafil and Boutilier, 2006] (node **(2)**); one may even attempt to automatically design the preference elicitation protocol along with (the rest of) the mechanism [Sandholm *et al.*, 2005a] (node **(3a)**); *etc.*

Another orthogonal objective is the following: rather than having all (relevant) information communicated to a center that then computes the outcome, have the agents compute the outcome themselves in a distributed (and possibly privacy-preserving) manner [Parkes and Shneidman, 2004; Brandt and Sandholm, 2004b,a, 2005b,c,a; Izmalkov *et al.*, 2005; Petcu *et al.*, 2006]. Moreover, in real-world implementation of preference aggregation, many other orthogonal issues come up. For example, when aggregating the preferences of humans, human-computer interaction issues must be considered.

In this dissertation, I will not cover research on topics that are orthogonal to the hierarchy. This is only for the purpose of coherence: it is certainly my opinion that such orthogonal topics concern crucial components of practical preference aggregation systems.

1.5 Outline

The rest of this dissertation is layed out as follows. In Chapter 2, we discuss various settings for expressive preference aggregation, including elections, allocations of tasks and resources using combinatorial auctions and exchanges, donations to (charitable) causes, and settings with externalities. Subsequently, in Chapter 3, which corresponds to node **(1)** in the hierarchy, we analyze the complexity of the outcome optimization problem in all of these settings. In Chapter 4, we briefly review some of the basic concepts in (classical) mechanism design, as well as some famous mechanisms and impossibility results. In Chapter 5, which corresponds to node **(2)** in the hierarchy, we present some difficulties and challenges for classical mechanism design in expressive preference aggregation settings. In Chapter 6, which corresponds to node **(3a)** in the hierarchy, we introduce automated mechanism design. In Chapter 7, we review basic concepts from game theory, as well as the basic argument for restricting attention to truthful mechanisms when all agents act in a strategically optimal way (which implies that they have no computational limitations). Chapters 8 and 9 correspond to node **(3b)** in the hierarchy—mechanism design for bounded agents. In Chapter 8, we display a setting in which there is non-truthful mechanism that always does at least as well as the best truthful mechanism, and, in the face of computational boundedness, does strictly better. In this chapter, we also show that in some voting settings, it is computationally hard to find beneficial manipulations (even given the other voters' votes); however, in the final section of the chapter, we show that under some reasonable restrictions on the voting setting, finding a successful manipulation remains easy in most manipulable instances. In Chapter 9 we take a first step towards rebuilding the theory of mechanism design in the face of computational limitations, by examining how hard various game-theoretic solution concepts are to compute—and hence, which concepts can serve as a reasonable basis for a theory of mechanism design for computationally bounded agents. (Being able to compute game-theoretic solutions is of interest for other reasons as well, as will be discussed in that chapter.) Finally, in Chapter 10, we introduce the first approach that combines aspects of both automated mechanism design and mechanism design for bounded agents. Rather than optimizing the entire mechanism in a single step, this approach incrementally identifies opportunities for the agents to successfully manipulate, and changes the mechanism to remove these opportunities.

To keep the dissertation coherent and of a reasonable length, much of the research that I have done as a Ph.D. student is excluded. The excluded material includes:

- A few topics that would have fit in the dissertation quite well but were excluded to keep the length reasonable. Topics excluded from Chapter 3 include work on computing optimal Kemeny rankings in voting [Conitzer *et al.*, 2006], and solving the winner determination problem in combinatorial auctions with k-wise dependent valuations [Conitzer *et al.*, 2005]. Topics excluded from Chapter 9 include a preprocessing technique for computing a Nash equilibrium [Conitzer and Sandholm, 2006g], mixed integer programming techniques for computing Nash equilibria [Sandholm *et al.*, 2005b], and computing the optimal strategy to commit to [Conitzer and Sandholm, 2006c].

- All work on *learning in games* (which could be considered relevant to node **(3b)** in the hierarchy) [Conitzer and Sandholm, 2006a, 2003d, 2004b].

- All work on *preference elicitation* (reducing the communication necessary for choosing the outcome) in combinatorial auctions [Santi *et al.*, 2004; Conitzer *et al.*, 2005], voting [Conitzer and Sandholm, 2002c, 2005b], and mechanism design [Conitzer and Sandholm, 2004c; Sandholm *et al.*, 2005a].

- All work on *cooperative* (also known as *coalitional*) game theory [Conitzer and Sandholm, 2004d; Yokoo *et al.*, 2005; Ohta *et al.*, 2006; Conitzer and Sandholm, 2006b].

- A few other individual topics, including the complexity of metareasoning [Conitzer and Sandholm, 2003f], interpreting voting rules as maximum likelihood estimates of the "correct" outcome [Conitzer and Sandholm, 2005a], and learning algorithms for online principal-agent settings [Conitzer and Garera, 2006].

- Some of the less-relevant results in papers that are otherwise covered.

Chapter 2

Expressive Preference Aggregation Settings

Freedom is, first of all, the chance to formulate the available choices, to argue over them – and then, the opportunity to choose.

C. Wright Mills

When the preferences of multiple agents need to be aggregated to choose an outcome (such as an allocation of resources), we require some process for doing so. One option is *ad hoc* negotiation, where the agents controlling the relevant resources attempt to improve the outcome locally, by proposing and accepting deals to each other as they see fit. The upsides of this approach are that no centralized computation is needed, and that the lack of constraints associated with this type of negotiation potentially allows the agents some amount of creativity in adapting to circumstances. The downside is the lack of guidance and oversight that the agents are confronted with. When deciding what deal to propose next (or whether to accept another agent's proposal), an agent needs to assess what is likely to happen in future negotiation. However, the unstructured approach of *ad hoc* negotiation makes this exceedingly difficult. In addition, the contract language needs to be complex to avoid getting stuck in local optima, making the process even less overseeable. As a result, in all but the simplest of settings, agents will likely only make ineffective deals and not come anywhere close to reaching the optimal outcome. (Early work by Sandholm and others [Sandholm, 1993b; Sandholm and Lesser, 1995; Sandholm, 1997; Andersson and Sandholm, 1999, 2000; Sandholm and Lesser, 2002] provides a more formalized version of such a distributed process.)

Another approach is to have a clear *protocol* that elicits information about the agents' preferences in a predictable, transparent, and commonly known manner to arrive at an outcome. Well-known examples include auction protocols (such as a *first-price sealed-bid auction*, where every agent submits a bid for the good for sale in a sealed envelope, and the highest bidder wins the item for the price she specifies), as well as voting protocols (such as the *Plurality* protocol, where everyone submits her most preferred candidate, and the candidate with the most votes wins). The clarity, transparency, and predictability of such protocols make it possible for agents to assess the likelihoods of future events and act in accordance. Unfortunately, naïvely designed protocols run the risk of being overly restrictive in the negotiation that they allow. For instance, suppose there are two items for sale, and we auction them off individually and sequentially. One bidder may consider the items complementary: neither item by itself would be useful to her, but together they are worth something. This bidder may be hesitant to bid high in the first auction, for fear that another bidder will win the second item—leaving her stuck with only the first item. This hesitancy may prevent her from winning the first item, even if the economically efficient outcome is for her to win both items. A likely event in this scenario is that the bidder seeks to strike a deal with the seller to buy both items outside of the auction, thereby reverting to *ad hoc* negotiation and the problems it entails.

The solution, of course, is to make sure that the protocols are not deemed too restrictive by the agents. In the example, the two items could be auctioned off simultaneously in a combinatorial auction, allowing bids on the bundle of both items. Protocols such as combinatorial auctions that allow the agents to express their full preferences, and that act on that information, are known as *expressive preference aggregation* protocols. In recent years, billions of dollars have been saved by applying such protocols to strategic sourcing [Sandholm *et al.*, 2006; Sandholm, 2006].

This dissertation will not consider any *ad-hoc* approaches to preference aggregation. Rather, it will focus on clear protocols that allow the agents to provide their preference information in expressive languages. The remainder of this chapter will introduce some preference aggregation settings, together with corresponding languages in which agents can express their preferences and criteria according to which the outcome can be selected. We will discuss computational aspects of these settings in later chapters. For example, Chapter 3 will discuss the computational complexity of and algorithms for choosing the optimal outcome in these settings. Some of the results in later chapters will not be specific to any particular setting, but the settings introduced in this chapter can serve as example domains.

The rest of this chapter is layed out as follows. In Section 2.1, we discuss *voting* (or *rank aggregation*) as an approach to preference aggregation. Here, each agent simply ranks all possible outcomes, and the outcome is chosen based on these rankings according to some *voting rule* (some example voting rules will be given). In Section 2.2, we discuss *allocation of tasks and resources*, and the use of combinatorial auctions and exchanges for doing so. In Section 2.3, we introduce a new application: letting multiple potential donors negotiate over who gives how much to which of multiple (say, charitable) causes [Conitzer and Sandholm, 2004e]. Finally, in Section 2.4, we study preference aggregation in settings with externalities and introduce a representation, a language for expressing agent preferences, and criteria for choosing an optimal outcome [Conitzer and Sandholm, 2005d].

2.1 Voting over alternatives (rank aggregation)

A very general approach to aggregating agents' preferences over outcomes is the following: let each agent rank all of the alternatives, and choose the winning alternative based on these rankings. (In some settings, rather than merely producing a winning alternative, one may wish to produce an aggregate ranking of all the alternatives.) This approach is often referred to as *voting* over the alternatives, and hence, in this context, agents are referred to as *voters*, the rankings that they submit as *votes*, and the alternatives as *candidates*.

For example, in a setting with three candidates a, b, c, voter 1 may vote $a \succ b \succ c$, voter 2 $b \succ a \succ c$, and voter 3 $a \succ c \succ b$. The winner (or aggregate ranking of the candidates) depends on which *voting rule* is used. Formally, letting C be the set of candidates, $R(C)$ the set of all possible rankings of the candidates, and n the number of voters, a voting rule is a mapping from $R(C)^n$ to C (if one only wishes to produce a winner) or to $R(C)$ (if one wishes to produce an aggregate ranking). One example rule is the *plurality* rule, where candidates are ranked simply according to how often they are ranked first by voters. In the example, a is ranked first twice, b once, and c never, so that the aggregate ranking produced by the plurality rule is $a \succ b \succ c$. Under the plurality rule, the voters effectively vote only for a single candidate (how the voter ranks the candidates below the top candidate is irrelevant).

Rules such as plurality may leave candidates tied, and typically these ties will need to be broken somehow (especially to choose a winning alternative). Throughout, we will make as few assumptions as possible on how ties are broken, but where we do make assumptions, we will make this clear. One may also wonder if we can allow for candidates to be tied in the *votes*. It is typically not difficult to extend voting rules and results to allow for this, but we will assume throughout that rankings are total orders on the candidates, *i.e.* they have no ties. (Some recent work has addressed extending voting theory to settings in which voters submit *partial orders* [Pini *et al.*, 2005; Rossi *et al.*, 2006]; this is significantly more involved than merely allowing for ties.)

But why should one use the plurality rule? Perhaps it would be desirable to give a vote's second-ranked candidate some points, or even to use a rule that is not based on awarding points to the candidates at all. We will see examples of such rules shortly. First, however, let us consider if perhaps there exists an "ideal" rule. If there are only two candidates, it is clear what the voting rule should do: the candidate that is ranked higher more often should win. This leads us to the following idea: for any pair of candidates, we can see which one is ranked more often. For instance, in the above example, a is ranked above b twice, whereas b is ranked above a only once—hence we say that a wins the *pairwise election* between a and b. Similarly, a defeats c in their pairwise election, and b defeats c. Hence, naturally, the aggregate ranking should be $a \succ b \succ c$ (which agrees with the plurality rule).

However, this line of reasoning is not always sufficient to produce a ranking (or even a winner). Consider a modified example in which voter 1 votes $a \succ b \succ c$, voter 2 $b \succ c \succ a$, and voter 3 $c \succ a \succ b$. Now, a defeats b in their pairwise election, b defeats c, and c defeats a—that is, we have a cycle, and our aggregate ranking cannot be consistent with the outcomes of all pairwise elections. This is known as a *Condorcet paradox*, and it shows that, unfortunately, in deciding whether a should be ranked higher than b in the aggregate ranking, we cannot simply ignore the position of c in the rankings.

Indeed, a famous theorem by Arrow [1963] states that there is no deterministic voting rule (for producing an aggregate ranking) that has all of the following properties:

- The rule is *non-dictatorial*, that is, at least two voters have the potential to affect the outcome.

- The rule is consistent with *unanimity*, that is, if all voters prefer a to b, then the aggregate ranking must rank a above b as well.

- The rule satisfies *independence of irrelevant alternatives*, that is, which of two alternatives is ranked higher in the aggregate ranking should be independent of how the other alternatives are ranked in the votes.

Arrow's theorem and the possibility of Condorcet paradoxes depend on the voters' being unrestricted in how they order the candidates. One well-known restriction that makes these problems disappear is *single-peakedness* of the voters' preferences. We say that preferences are single-peaked if there is a total order $<$ on the candidates, and for any voter i and any three candidates $a < b < c$, $a \succ_i b \Rightarrow b \succ_i c$ and $c \succ_i b \Rightarrow b \succ_i a$. In words, the candidates are arranged on a spectrum from left to right, and a voter never prefers a candidate that is further from the voter's most preferred candidate (the voter's "peak") to a closer one. (Note that this definition does not compare candidates on the left side of a voter's peak with those on the right side in terms of closeness, that is, the notion of "closer to the peak" only applies to pairs of candidates that are on the same side of the peak. Also note that the order $<$ must be the *same* for all voters.) If the voters' preferences are single-peaked, then there are no Condorcet cycles. If we order the voters by their peaks, then the peak of the voter in the middle of this ordering (the *median* voter) will win all pairwise elections, that is, it is a *Condorcet winner*. (This is assuming that a median voter exists, *i.e.* the number of voters is odd.)

Nevertheless, in many settings the votes do not have any (apparent) structure, so that it is still important to define voting rules for the general case. Next, we review the most common voting rules. We will define them according to how they rank candidates; the winner is the top-ranked candidate.

- *Scoring rules*. Let $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_m \rangle$ be a vector of integers. For each vote, a candidate receives α_1 points if it is ranked first in the vote, α_2 if it is ranked second, *etc.* Candidates are ranked by their scores. The *Borda* rule is the scoring rule with $\vec{\alpha} = \langle m-1, m-2, \ldots, 0 \rangle$. The *plurality* rule is the scoring rule with $\vec{\alpha} = \langle 1, 0, \ldots, 0 \rangle$. The *veto* rule is the scoring rule with $\vec{\alpha} = \langle 1, 1, \ldots, 1, 0 \rangle$.

- *Single transferable vote (STV)*. This rule proceeds through a series of $m - 1$ rounds. In each round, the candidate with the lowest plurality score (that is, the fewest votes ranking it first among the remaining candidates) is eliminated (and each of the votes for that candidate "transfers" to the next remaining candidate in the order given in that vote). The candidates are ranked in reverse order of elimination.

- *Plurality with run-off*. In this rule, a first round eliminates all candidates except the two with the highest plurality scores. Votes are transferred to these as in the STV rule, and a second round determines the winner from these two. Candidates are ranked according to Plurality scores, with the exception of the top two candidates whose relative ranking is determined according to the runoff.

- *Maximin* (aka. *Simpson*). For any two candidates a and b, let $N(a, b)$ be the number of votes that prefer a to b. The *maximin score* of a is $s(a) = \min_{b \neq a} N(a, b)$—that is, a's worst performance in a pairwise election. Candidates are ranked by their scores.

- *Copeland*. For any two candidates a and b, let $C(a, b) = 1$ if $N(a, b) > N(b, a)$, $C(a, b) = 1/2$ if $N(a, b) = N(b, a)$, and $C(a, b) = 0$ if $N(a, b) < N(b, a)$. The *Copeland score* of candidate a is $s(a) = \sum_{b \neq a} C(a, b)$. Candidates are ranked by their scores.

- *Bucklin*. For any candidate a and integer l, let $B(a, l)$ be the number of votes that rank candidate a among the top l candidates. For each candidate a, let $l(a)$ be the lowest l such that $B(a, l) > n/2$. Candidates are ranked inversely by $l(a)$. As a tiebreaker, $B(a, l(a))$ is used.

- *Slater*. The Slater rule produces a ranking that is inconsistent with the outcomes of as few pairwise elections as possible. That is, for a given ranking of the candidates, each pair of candidates a, b such that a is ranked higher than b, but b defeats a in their pairwise election, counts as an inconsistency, and a ranking is a Slater ranking if it minimizes the number of inconsistencies.

- *Kemeny*. This rule produces a ranking that minimizes the number of times that the ranking is inconsistent with a vote on the ranking of two candidates. That is, for a given ranking r of the candidates, each combination of a pair of candidates a, b and a vote r_a such that r ranks a higher than b, but r_i ranks b higher than a, counts as an inconsistency, and a ranking is a Kemeny ranking if it minimizes the number of inconsistencies.

We define one additional rule, the *cup* rule, which runs a single-elimination tournament to decide the winning candidate. This rule does not produce a full aggregate ranking of the candidates, and additionally requires a *schedule* for matching up the remaining candidates.

- *Cup*. This rule is defined by a balanced[1] binary tree T with one leaf per candidate, and a *schedule*, that is, an assignment of candidates to leaves (each leaf gets one candidate). Each non-leaf node is assigned the winner of the pairwise election of the node's children; the candidate assigned to the root wins. The *regular cup* rule assumes that the assignment of candidates to leaves is known by the voters before they vote. In the *randomized cup* rule, the assignment of candidates to leaves is chosen uniformly at random after the voters have voted.

Sometimes votes are *weighted*; a vote of weight K counts as K votes of weight 1. Different possible interpretations can be given to weights. They may represent the decision power of a given agent in a voting setting where not all agents are considered equal. The weight may correspond to the size of the community that the voter represents (such as the size of the state). Or, when agents vote in partisan groups (*e.g.*, in parliament), the weights may correspond to the size of the group (each group acts as one voter).

We will sometimes use the term "voting protocol" rather than "voting rule"; the meaning is roughly the same, except the word "protocol" is intended to encompass not only the mapping from rankings to outcomes (*i.e.*, aggregate rankings or winners), but also procedural aspects such as the manner in which the voters report their ranking (*e.g.*, whether all voters submit their rankings at the same time or not).

The general applicability of voting makes it an appealing approach to preference aggregation in unstructured domains. However, in more structured settings, this generality becomes a weakness, as using a voting approach does not exploit the structure of the domain. For example, many settings allow *payments* to be made by or to the agents. In principle, we can model these payments as part of the outcome, so that voter 1's vote may be something like:

[1] "Balanced" here means that the difference in depth between two leaves can be at most one.

"alternative a is chosen, voter 1 pays \$10, voter 2 pays \$5" $\succ$ "alternative b is chosen, voter 1 pays \$0, voter 2 pays \$3" $\succ$ "alternative a is chosen, voter 1 pays \$10, voter 2 pays \$6" $\succ \ldots$

Needless to say, this approach is extremely cumbersome (in principle the votes have infinite length!), and it does not exploit any of the knowledge that we have (or assumptions that we are willing to make) about how agents feel about payments. For example, we know that agents prefer smaller payments to larger ones; we may know that they do not care about other agents' payments; we may know that each dollar is as valuable as the next to an agent; *etc.*

Another drawback is that the voting approach does not allow us to make statements about how strong or weak agents' preferences over outcomes are, and hence how they feel about *distributions* over outcomes. For example, suppose an agent prefers a to b to c. Which does the agent prefer: b, or a coin flip between a and c? It is impossible to tell from the information given—we do not even know whether the agent's preference of a over b is stronger than that of b over c. Again, in principle, voters can vote over distributions over outcomes, *e.g.*:

$P(a) = .4, P(b) = .3, P(c) = .3 \succ P(a) = .5, P(b) = .2, P(c) = .3 \succ P(a) = .5, P(b) = .3, P(c) = .2 \succ \ldots$

but again this is impractical (if not impossible). Again, we can make very reasonable assumptions about agents' preferences over distributions: for example, Dutch book theorems [Mas-Colell *et al.*, 1995] suggest that agents will maximize their expected utility, because otherwise they will be susceptible to accepting a sequence of bets that is guaranteed to leave them worse off.

In the remainder of this chapter, we focus on utility-based approaches; we will return to voting (specifically, computing aggregate rankings using the Slater rule) in the next chapter, Section 3.1.

2.2 Allocation of tasks and resources

Some of the most common domains in which multiple agents' preferences must be aggregated involve the allocation of resources or tasks to the agents. I will restrict my attention to settings in which payments can be made, that is, agents can pay for resources allocated to them and tasks performed for them, or be compensated for resources they supply and tasks they perform. (Not all research on resource/task allocation makes the assumption that payments are possible: for example, Lipton *et al.* [2004] and Bouveret and Lang [2005] consider the problem of finding *envy-free* allocatons, that is, allocations under which no agent would prefer the share of another agent to its own.)

We will refer to distinct resources as *items*; the performance of a task can be thought of as an item as well, so from now on we can, without loss of generality, focus strictly on the allocation and provision of items.

Earlier, we discussed combinatorial auctions as a method for allocating a fixed set of n available items, I. Here, agent (or *bidder*) i will have a valuation function $v_i : 2^I \to \mathbb{R}$, mapping each bundle of items that could be allocated to that bidder to a real value. This is making the assumption of *no externalities*: given that a bidder does not win an item, that bidder does not care which (if any) other bidder receives the item instead. This is usually realistic, but not always: for example, a country may prefer certain other countries not to obtain certain weapons. We will discuss externalities in the next sections. Letting $A_i \subseteq I$ denote the subset of the items that we allocate to bidder i, our goal is to find an allocation of items to the n bidders that maximizes $\sum_{i=1}^{n} v_i(A_i)$, under the constraint that for any $i \neq j$, $A_i \cap A_j = \emptyset$ (we do not award an item twice). (Note that we do not require all items to be awarded—this is known as the *free disposal assumption*.) This optimization problem is called the *winner determination problem*.

Typically, we can assume $v_i(\emptyset) = 0$, but any other restriction on the agents' valuation function will come at a loss in what the agents can express. For example, if we were to assume that $v_i(S) = \sum_{s \in S} v_i(\{s\})$, then we can no longer model complementarity (multiple items being worth more than the sum of their individual values) or substitutability (multiple items being worth less than the sum of their individual values), which is what gives combinatorial auctions their advantage over sequential or parallel auctions of individual items. On the other hand, an arbitrary valuation function requires $2^n - 1$ real values to describe, which corresponds to an infeasibly large amount of communication by a bidder if the number of items is reasonably large. This leads us to the study of *bidding languages* that the bidders can use to express their valuations. The earliest-studied language is the *OR bidding language*, in which bidders simply submit valuations for multiple bundles [Rothkopf *et al.*, 1998; DeMartini *et al.*, 1999]. An example bid is $(\{a\}, 3)$ OR $(\{b, c\}, 4)$ OR $(\{c, d\}, 2)$, which expresses that the bidder is willing to pay 3 for the bundle consisting of a alone, 4 for the bundle consisting of b and c, and 2 for the bundle consisting of c and d. In addition, *any number* of the bidder's bundles may be awarded simultaneously, at the sum of the values of the individual bundles. For example, the example bid implies that $v_i(\{a, b, c\}) = 7$. Multiple bundles cannot be awarded simultaneously if the items overlap. For example, the example bid does *not* imply that $v_i(\{b, c, d\}) = 6$. Hence, under the OR bidding language, from the perspective of winner determination, we may as well imagine that each bundle-value pair came from a separate bidder. (A bidder that is only interested in a single bundle is called a *single-minded* bidder.)

The OR bidding language is not fully expressive. For example, imagine a combinatorial auction in which two cars a and b are for sale. Now imagine a bidder that values car a at $4,000 and car b at $6,000. However, the bidder only requires one car, so that winning both cars would still be worth only $6,000 to the bidder, because the bidder would simply not use car a (and we will assume that re-selling the car is impossible). This type of bidder is known as a *unit-demand* bidder, and it is impossible to capture these preferences using the OR bidding language. For example, bidding $(\{a\}, 4000)$ OR $(\{b\}, 6000)$ would imply $v_i(\{a, b\}) = 10000$. The subtitutability of the items is what causes the problem here. To address this, we can introduce *XOR-constraints* between different bundles, indicating that only one of these bundles can be accepted [Sandholm, 2002a,b]. For example, the above valuation function can be expressed by $(\{a\}, 4000)$ XOR $(\{b\}, 6000)$. Bidding with XOR constraints is fully expressive, that is, we can model any valuation function with them (even without using any ORs). To reduce the size of the bids, however, we may still wish to use ORs in addition to XORs [Nisan, 2000; Sandholm, 2002b]. The presence of XORs prevents us from pretending that each bundle-value pair was submitted by a separate bidder for the purpose of winner determination. A commonly used trick to circumvent this problem is the following: for a set of bundle-value pairs that is XORed together, create a *dummy item* that is added to all of these bundles [Fujishima *et al.*, 1999; Nisan, 2000]. Since the dummy item can only be awarded once, the dummy item effectively encodes the XOR-constraint, so that we can still imagine that each bid comes from a separate bidder (for the purposes of winner determination).

Combinatorial auctions do not capture all possible resource/task allocation settings (even those in which payments are possible). For example, rather than having a set of items available for sale, one may instead seek to *procure* a set of items which are distributed across multiple bidders (or *suppliers*). This is especially natural in the context of task allocation, where the items to be procured correspond to tasks that must be performed. In such a setting, one can hold a *combinatorial reverse auction* [Sandholm *et al.*, 2002], in which a set of items I needs to be procured, and each bidder i has a cost function $c_i : 2^I \rightarrow \mathbb{R}$, where $c_i(S)$ indicates the cost of providing bundle S. Letting $A_i \subseteq I$ denote the subset of the items that we award to bidder i (in the sense that we require bidder i to provide them), our goal is to minimize $\sum_{i=1}^{m} c_i(A_i)$, under the constraint that $I = \bigcup_{1 \leq i \leq n} A_i$. (Again, there is a free disposal assumption here in that we allow an item to be provided by multiple bidders.)

The free disposal assumption is not always realistic: for example, one may not be able to dispose of radioactive material freely. A combinatorial forward auction *without free disposal* [Sandholm *et al.*, 2002] is exactly the same as one with free disposal, with the exception that every item must be allocated to some bidder. Here, bids with a *negative* value may be useful, as they allow us to remove some of the items—which may allow us to accept better bids for the remaining items. Similarly, a combinatorial reverse auction without free disposal is exactly the same as one with free disposal, with the exception that no additional items can be procured. Here, bids with a *negative* value may occur—the (nondisposable) item may be a liability to the bidder. In both cases, we seek to identify a subset of the bids that constitutes an exact cover of the items (no item covered more than once), and to maximize the bidders' total utility under this constraint. Therefore, the settings are technically identical, and we can without loss of generality restrict our attention to forward auctions without free disposal.

In general, it is not the case that either all agents are seeking only to procure items, or all agents are seeking only to provide items. Rather, some agents may be seeking to procure items; others, to provide items; and yet others, to do both simultaneously. This leads to the model of a *combinatorial exchange* [Sandholm *et al.*, 2002], in which there is a set of m items I for sale, and bidder i has a valuation functions $v_i : \mathbb{Z}^m \rightarrow \mathbb{R}$. Here, $v_i(\lambda_1, \ldots, \lambda_m)$ is bidder i's value for receiving λ_j units of item j. (If λ_j is negative, that means that the bidder is providing units of that item.)

It should be noted that the notion of bidding for multiple units of the same item can be applied to combinatorial auctions and reverse auctions as well. However, there it is not strictly necessary, in the sense that we can re-model a combinatorial (reverse) auction with multiple units as one with single units, by describing each individual unit of an item that is for sale or to be procured as a separate item. The "sameness" of some of these new items will then be implied by the fact that the bidders' valuation or cost functions treat these items symmetrically. (This may, however, be grossly inefficient from a representational and computational standpoint.) In an exchange, however, the number of units of each item that is for sale/to be procured is not known *ex ante*, which prevents this trick.

We will return to combinatorial auctions (specifically, to the complexity of the winner determination problem) in the next chapter, Section 3.2. In the remainder of this chapter, we will focus on settings where an agent's utility depends on more than his own items, tasks, and payments—that is, we will drop the *no externalities* assumption.

2.3 Donations to (charitable) causes

When money is donated to a charitable (or other) cause (hereafter simply referred to as a *charity*), often the donating party gives *unconditionally*: a fixed amount is transferred from the donator to the charity, and none of this transfer is contingent on other events—in particular, it is not contingent on the amount given by other parties. Indeed, this is currently often the only way to make a donation, especially for small donating parties such as private individuals. However, when multiple parties support the same charity, each of them would prefer to see the others give more rather than less to this charity. In such scenarios, it is sensible for a party to use its contemplated donation to induce the others to give more. This is done by making the donation conditional on the others' donations. The following example will illustrate this, and show that the donating parties as well as the charitable cause may simultaneously benefit from the potential for such negotiation.

Suppose we have two parties, 1 and 2, who are both supporters of charity A. To either of them, it would be worth \$0.75 if A received \$1. It follows neither of them will be willing to give unconditionally, because $\$0.75 < \$1$. However, if the two parties draw up a contract that says that they will each give \$0.5, both the parties have an incentive to accept this contract (rather than have no contract at all): with the contract, the charity will receive \$1 (rather than \$0 without a contract), which is worth \$0.75 to each party, which is greater than the \$0.5 that that party will have to give. Effectively, each party has made its donation conditional on the other party's donation, leading to larger donations and greater happiness to all parties involved.[2]

One method that is often used to effect this is to make a *matching offer*. Examples of matching offers are: "I will give x dollars for every dollar donated.", or "I will give x dollars if the total collected from other parties exceeds y." In our example above, one of the parties can make the offer "I will donate \$0.5 if the other party also donates at least that much", and the other party will have an incentive to indeed donate \$0.5, so that the total amount given to the charity increases by \$1. Thus this matching offer implements the contract suggested above. As a real-world example, the United States government has authorized a donation of up to \$1 billion to the Global Fund to fight AIDS, TB and Malaria, under the condition that the American contribution does not exceed one third of the total—to encourage other countries to give more [Tagliabue, 2003].

However, there are several severe limitations to the simple approach of matching offers as just described.

1. It is not clear how two parties can make matching offers where each party's offer is stated in terms of the amount that the other pays. (For example, it is not clear what the outcome should be when both parties offer to match the other's donation.) Thus, matching offers can

[2]The preferences given in this example do not consider the possibility that an agent's utility depends not only on how much the charity receives but also on the extent to which that agent feels responsible for it. For example, an agent may feel better about the scenario in which the agent gives \$1 to a charity than about the scenario in which the agent loses \$1 gambling and another agent gives \$1 to the charity. However, we will not consider such preferences and assume that an agent only cares about the final amount received by each charity (as well as about the agent's own final budget).

only be based on payments made by parties that are giving unconditionally (not in terms of a matching offer)—or at least there can be no circular dependencies.[3]

2. Given the current infrastructure for making matching offers, it is impractical to make a matching offer depend on the amounts given to *multiple* charities. For instance, a party may wish to specify that it will pay \$100 given that charity A receives a total of \$1000, but that it will also count donations made to charity B, at half the rate. (Thus, a total payment of \$500 to charity A combined with a total payment of \$1000 to charity B would be just enough for the party's offer to take effect.)

In contrast, in this section we propose a new approach where each party can express its relative preferences for different charities, and make its offer conditional on its own appreciation for the vector of donations made to the different charities. Moreover, the amount the party offers to donate at different levels of appreciation is allowed to vary arbitrarily (it does need to be a dollar-for-dollar (or n-dollar-for-dollar) matching arrangement, or an arrangement where the party offers a fixed amount provided a given (strike) total has been exceeded). Finally, there is a clear interpretation of what it means when multiple parties are making conditional offers that are stated in terms of each other. Given each combination of (conditional) offers, there is a (usually) unique solution which determines how much each party pays, and how much each charity is paid.

In short, expressive preference aggregation for donations to charities is a new way in which electronic commerce can help the world. A web-based implementation of the ideas described in this section can facilitate voluntary reallocation of wealth on a global scale.

2.3.1 Definitions

Throughout, we will refer to the offers that the donating parties make as *bids*, and to the donating parties as *bidders*. In our bidding framework, a bid will specify, for each vector of total payments made to the charities, how much that bidder is willing to contribute. (The contribution of this bidder is also counted in the vector of payments—so, the vector of total payments to the charities represents the amount given by *all* donating parties, not just the ones other than this bidder.) The bidding language is expressive enough that no bidder should have to make more than one bid. The following definition makes the general form of a bid in our framework precise.

Definition 1 *In a setting with m charities $c_1, c_2, \ldots, c_m$, a* bid *by bidder b_j is a function $v_j : \mathbb{R}^m \to \mathbb{R}$. The interpretation is that if charity c_i receives a total amount of π_{c_i}, then bidder j is willing to donate (up to) $v_j(\pi_{c_1}, \pi_{c_2}, \ldots, \pi_{c_m})$.*

We now define possible outcomes in our model, and which outcomes are valid given the bids that were made.

Definition 2 *An* outcome *is a vector of payments made by the bidders $(\pi_{b_1}, \pi_{b_2}, \ldots, \pi_{b_n})$, and a vector of payments received by the charities $(\pi_{c_1}, \pi_{c_2}, \ldots, \pi_{c_m})$. A* valid *outcome is an outcome where*

[3] Typically, larger organizations match offers of private individuals. For example, the American Red Cross Liberty Disaster Fund maintains a list of businesses that match their customers' donations [Goldburg and McElligott, 2001].

1. $\sum_{j=1}^{n} \pi_{b_j} \geq \sum_{i=1}^{m} \pi_{c_i}$ *(at least as much money is collected as is given away);*

2. *For all* $1 \leq j \leq n$, $\pi_{b_j} \leq v_j(\pi_{c_1}, \pi_{c_2}, \ldots, \pi_{c_m})$ *(no bidder gives more than she is willing to).*

Of course, in the end, only one of the valid outcomes can be chosen. We choose the valid outcome that maximizes the *objective* that we have for the donation process.

Definition 3 *An* objective *is a function from the set of all outcomes to* $\mathbb{R}$.[4] *After all bids have been collected, a valid outcome will be chosen that maximizes this objective.*

One example of an objective is *surplus*, given by $\sum_{j=1}^{n} \pi_{b_j} - \sum_{i=1}^{m} \pi_{c_i}$. The surplus could be the profits of a company managing the expressive donation marketplace; but, alternatively, the surplus could be returned to the bidders, or given to the charities. Another objective is *total amount donated*, given by $\sum_{i=1}^{m} \pi_{c_i}$. (Here, different weights could also be placed on the different charities.)

2.3.2 A simplified bidding language

Specifying a general bid in our framework (as defined above) requires being able to specify an arbitrary real-valued function over $\mathbb{R}^m$. Even if we restricted the possible total payment made to each charity to the set $\{0, 1, 2, \ldots, s\}$, this would still require a bidder to specify $(s+1)^m$ values. Thus, we need a bidding language that will allow the bidders to at least specify *some* bids more concisely. We will specify a bidding language that only represents a subset of all possible bids, which can be described concisely.[5]

To introduce our bidding language, we will first describe the bidding function as a composition of two functions; then we will outline our assumptions on each of these functions. First, there is a *utility function* $u_j : \mathbb{R}^m \rightarrow \mathbb{R}$, specifying how much bidder j appreciates a given vector of total donations to the charities. (Note that the way we define a bidder's utility function, it does not take the payments the bidder makes into account.) Then, there is a *donation willingness* function $w_j : \mathbb{R} \rightarrow \mathbb{R}$, which specifies how much bidder j is willing to pay given her utility for the vector of donations to the charities. We emphasize that this function does *not* need to be linear, so that utilities should not be thought of as expressible in dollar amounts. (Indeed, when an individual is donating to a large charity, the reason that the individual donates only a bounded amount is typically not decreasing marginal value of the money given to the charity, but rather that the marginal value of a dollar to the bidder herself becomes larger as her budget becomes smaller.) So, we have $w_j(u_j(\pi_{c_1}, \pi_{c_2}, \ldots, \pi_{c_m})) = v_j(\pi_{c_1}, \pi_{c_2}, \ldots, \pi_{c_m})$, and we let the bidder describe her functions u_j and w_j separately. (She will submit these functions as her bid.)

[4] In general, the objective function may also depend on the bids, but the objective functions that we consider do not depend on the bids. The techniques presented in this dissertation will typically generalize to objectives that take the bids into account directly.

[5] Of course, our bidding language can be trivially extended to allow for fully expressive bids, by also allowing bids from a fully expressive bidding language, in addition to the bids in our bidding language.

Our first restriction is that the utility that a bidder derives from money donated to one charity is *independent* of the amount donated to another charity. Thus, $u_j(\pi_{c_1}, \pi_{c_2}, \ldots, \pi_{c_m}) = \sum_{i=1}^{m} u_j^i(\pi_{c_i})$. (We observe that this does *not* imply that the bid function v_j decomposes similarly, because of the nonlinearity of w_j.) Furthermore, each u_j^i must be piecewise linear. An interesting special case which we will study is when each u_j^i is a line: $u_j^i(\pi_{c_i}) = a_j^i \pi_{c_i}$. This special case is justified in settings where the scale of the donations by the bidders is small relative to the amounts the charities receive from other sources, so that the marginal use of a dollar to the charity is not affected by the amount given by the bidders.

The only restriction that we place on the payment willingness functions w_j is that they are piecewise linear. One interesting special case is a *threshold bid*, where w_j is a step function: the bidder will provide t dollars if her utility exceeds s, and otherwise 0. Another interesting case is when such a bid is *partially acceptable*: the bidder will provide t dollars if her utility exceeds s; but if her utility is $u < s$, she is still willing to provide $\frac{ut}{s}$ dollars.

One might wonder why, if we are given the bidders' utility functions, we do not simply maximize the sum of the utilities rather than surplus or total donated. There are several reasons. First, because affine transformations do not affect utility functions in a fundamental way, it would be possible for a bidder to inflate her utility by changing its units, thereby making her bid more important for utility maximization purposes. Second, a bidder could simply give a payment willingness function that is 0 everywhere, and have her utility be taken into account in deciding on the outcome, in spite of her not contributing anything.

2.3.3 Avoiding indirect payments

In an initial implementation, the approach of having donations made out to a center, and having a center forward these payments to charities, may not be desirable. Rather, it may be preferable to have a *partially decentralized* solution, where the donating parties write out checks to the charities directly according to a solution prescribed by the center. In this scenario, the center merely has to verify that parties are giving the prescribed amounts. Advantages of this include that the center can keep its legal status minimal, as well as that we do not require the donating parties to trust the center to transfer their donations to the charities (or require some complicated verification protocol). It is also a step towards a fully decentralized solution, if this is desirable.

To bring this about, we can still use the approach described earlier. After we clear the market in the manner described before, we know the amount that each donator is supposed to give, and the amount that each charity is supposed to receive. Then, it is straightforward to give some specification of who should give how much to which charity, that is consistent with that clearing.

Nevertheless, with this approach, a bidder may have to write out a check to a charity that she does not care for at all. (For example, an environmental activist who was using the system to increase donations to a wildlife preservation fund may be required to write a check to a group supporting a right-wing political party.) This is likely to lead to complaints and noncompliance with the clearing. We can address this issue by letting each bidder specify explicitly (before the clearing) which charities she would be willing to make a check out to. These additional constraints, of course, may change the optimal solution.

The setting of expressive preference aggregation for donations to charities is a special case of a more general phenomenon, namely that agents' actions may indirectly affect other agents' utilities. We study this more general setting in the next section. We will return to the more specific setting of expressive preference aggregation for donations to charities (specifically, to the complexity of computing optimal outcomes in this setting) in the next chapter, Section 3.3.

2.4 Public goods and externalities

A pervasive assumption in the research on combinatorial auctions and exchanges has been that there are *no allocative externalities*: no agent cares what happens to an item unless that agent itself receives the item. This is insufficient to model situations where there are certain items (such as nuclear weapons) that are such that bidders who do not win the item still care which other bidder wins it [Jehiel and Moldovanu, 1996]. More generally, there are many important preference aggregation settings where decisions taken by a few agents may affect many other agents. For example, many agents may benefit from one agent taking on a task such as building a bridge (and the extent of their benefit may depend on how the bridge is built, for example, on how heavy a load it can support). Alternatively, if a company reduces its pollution level, many individuals may benefit, even if they have nothing to do with the goods that the company produces. A decision's effect on an otherwise uninvolved agent is commonly known as an *externality* [Mas-Colell *et al.*, 1995]. In designing a good preference aggregation protocol, externalities must be taken into account, so that (potentially complex) arrangements can be made that are truly to every agent's benefit.

In this section, we define a representation for combinatorial preference aggregation settings with externalities. We will mostly focus on restricted settings that cannot model *e.g.* fully general combinatorial auctions and exchanges, so that we do not inherit all of the complexities from those settings.

We formalize the problem setting as follows.

Definition 4 *In a* setting with externalities*, there are n agents $1, 2, \ldots, n$; each agent i controls m_i variables $x_i^1, x_i^2, \ldots, x_i^{m_i} \in \mathbb{R}^{\geq 0}$; and each agent i has a utility function $u_i : \mathbb{R}^M \to \mathbb{R}$ (where $M = \sum_{j=1}^{n} m_j$). (Here, $u_i(x_1^1, \ldots, x_1^{m_1} \ldots, x_n^1, \ldots, x_n^{m_n})$ represents agent i's utility for any given setting of the variables.)*

In general, one can also impose constraints on which values for $(x_i^1, \ldots, x_i^{m_i})$ agent i can choose, but we will refrain from doing so in this section. (We can effectively exclude certain settings by making the utilities for them very negative.) We say that the *default outcome* is the one where all the x_i^j are set to 0,[6] and we require without loss of generality that all agents' utilities are 0 at the default outcome. Thus, the participation constraint states that every agent's utility should be nonnegative.

[6]This is without loss of generality because the variables x_i^j can be used to represent the *changes* in the real-world variables relative to the default outcome. If these changes can be both positive and negative for some real-world variable, we can model this with two variables $x_i^{j_1}, x_i^{j_2}$, the difference between which represents the change in the real-world variable.

Without any restrictions placed on it, this definition is very general. For instance, we can model a (multi-item, multi-unit) combinatorial exchange with it. Recall that in a combinatorial exchange, each agent has an initial endowment of a number of units of each item, as well as preferences over endowments (possibly including items not currently in the agent's possession). The goal is to find some reallocation of the items (possibly together with a specification of payments to be made and received) so that no agent is left worse off, and some objective is maximized under this constraint. We can model this in our framework as follows: for each agent, for each item in that agent's possession, for each other agent, let there be a variable representing how many units of that item the former agent transfers to the latter agent. (If payments are allowed, then we additionally need variables representing the payment from each agent to each other agent.) We note that this framework allows for *allocative externalities*, that is, for the expression of preferences over which of the other agents receives a particular item.

Of course, if the agents can have nonlinear preferences over bundles of items (there are complementarities or substitutabilities among the items), then (barring some special concise representation) specifying the utility functions requires an exponential number of values.[7] We need to make some assumption about the structure of the utility functions if we do not want to specify an exponential number of values. For the most part, we make the following assumption, which states that the effect of one variable on an agent's utility is independent of the effect of another variable on that agent's utility. We note that this assumption disallows the model of a combinatorial exchange that we just gave, unless there are no complementarities or substitutabilities among the items. This is not a problem insofar as our primary interest here is not so much in combinatorial exchanges as it is in more natural, simpler externality problems such as aggregating preferences over pollution levels. We note that this restriction makes the hardness results on outcome optimization that we present in the next chapter (Section 3.4) much more interesting (without the restriction, the results would have been unsurprising given known hardness results in combinatorial exchanges). However, for some of our positive results we will actually not need the assumption, for example for convergence results for an algorithm that we will present.

Definition 5 u_i decomposes (across variables) *if* $u_i(x_1^1, \ldots, x_1^{m_1}, \ldots, x_n^1, \ldots, x_n^{m_n}) = \sum_{k=1}^{n} \sum_{j=1}^{m_k} u_i^{k,j}(x_k^j)$.

When utility functions decompose, we will sometimes be interested in the special cases where the $u_i^{k,j}$ are step functions (denoted $\delta_{x \geq a}$, which evaluates to 0 if $x < a$ and to 1 otherwise), or piecewise constant functions (linear combinations of step functions).[8]

In addition, we will focus strictly on settings where the higher an agent sets its variables, the worse it is for itself. We will call such settings *concessions settings*. So, if there is no preference aggregation, each agent will selfishly set all its variables to 0 (the *default* outcome).

---

[7]Thus, the fact that finding a feasible solution for a combinatorial exchange is NP-complete [Sandholm *et al.*, 2002] does not imply that finding a feasible solution in our framework is NP-complete, because there is an exponential blowup in representation.

[8]For these special cases, it may be conceptually desirable to make the domains of the variables x_i^j discrete, but we will refrain from doing so in this dissertation for the sake of consistency.

Definition 6 *A* concessions setting *is a setting with externalities for which for any* $(x_1^1, \ldots, x_1^{m_1}, \ldots, x_n^1, \ldots, x_n^{m_n}) \in \mathbb{R}^M$*, for any* $i, 1 \leq j \leq m_i$*, and for any* $\hat{x}_i^j > x_i^j$*, we have* $u_i(x_1^1, \ldots, x_1^{m_1}, \ldots, \hat{x}_i^j, \ldots, x_n^1, \ldots, x_n^{m_n}) \leq u_i(x_1^1, \ldots, x_1^{m_1}, \ldots, x_i^j, \ldots, x_n^1, \ldots, x_n^{m_n})$.

In parts of this dissertation, we will be interested in the following additional assumption, which states that the higher an agent sets its variables, the better it is for the others. (For instance, the more a company reduces its pollution, the better it is for all others involved.)

Definition 7 *A concessions setting has* only negative externalities *if for any* $(x_1^1, \ldots, x_1^{m_1}, \ldots, x_n^1, \ldots, x_n^{m_n}) \in \mathbb{R}^M$*, for any* $i, 1 \leq j \leq m_i$*, for any* $\hat{x}_i^j > x_i^j$*, and for any* $k \neq i$*,* $u_k(x_1^1, \ldots, x_1^{m_1}, \ldots, \hat{x}_i^j, \ldots, x_n^1, \ldots, x_n^{m_n}) \geq u_k(x_1^1, \ldots, x_1^{m_1}, \ldots, x_i^j, \ldots, x_n^1, \ldots, x_n^{m_n})$.

We define *trivial* settings of variables as settings that are indistinguishable from setting them to 0.

Definition 8 *The value* r *is* trivial for *variable* x_i^j *if it does not matter to anyone's utility function whether* x_i^j *is set to* r *or to* 0*. (That is, for any* $x_1^1, \ldots, x_1^{m_1}, \ldots, x_i^{j-1}, x_i^{j+1}, \ldots, x_n^1, \ldots, x_n^{m_n}$*, and for any* k*, we have* $u_k(x_1^1, \ldots, x_1^{m_1}, \ldots, x_i^{j-1}, r, x_i^{j+1}, \ldots, x_n^1, \ldots, x_n^{m_n}) = u_k(x_1^1, \ldots, x_1^{m_1}, \ldots, x_i^{j-1}, 0, x_i^{j+1}, \ldots, x_n^1, \ldots, x_n^{m_n})$*). A setting of all the variables is* trivial *if each variable is set to a trivial value.*

Say that an outcome is *feasible* if no agent prefers the default outcome to it (and would therefore try to block the preference aggregation process). Our goal is to find a good feasible outcome. As in the setting of expressive preference aggregation for donations to charities, there can be multiple objectives. Interesting objectives include social welfare and total concessions (the sum of the variables). A more modest, but nevertheless interesting goal is to simply discover a nontrivial feasible solution. We will study how hard these problems are computationally in the next chapter, Section 3.4.

2.5 Summary

In this chapter, we reviewed four different settings for preference aggregation. We also reviewed corresponding languages in which agents can express their preferences, and criteria according to which the outcome can be selected.

In Section 2.1, we reviewed *voting settings*, where agents (or voters) can rank the alternatives (or candidates) in any order, and the winner or aggregate ranking of the alternatives is chosen based on the rankings (or votes) submitted by the voters. We reviewed some basic concepts from social choice theory (Condorcet cycles, Arrow's impossibility result, and single-peaked preferences), as well as a number of specific rules for choosing the outcome based on the submitted votes.

In Section 2.2, we reviewed allocation of tasks and resources, using combinatorial auctions, reverse auctions, and exchanges. We defined the winner determination problem and reviewed basic bidding languages, including OR- and XOR-based languages.

In Section 2.3, we introduced a new application: expressive preference aggregation for donations to charities. The idea here is that when donating money to a charity, it is possible to use the contemplated donation in negotiation to induce other parties interested in the charity to donate more. We introduced a bidding language for expressing very general types of matching offers over multiple charities, and formulated the corresponding clearing problem (deciding how much each bidder pays, and how much each charity receives).

Finally, in Section 2.4, we introduced a framework for negotiating in general settings with externalities. Again, we introduced a language in which agents can express their preferences, and criteria according to which to select the final outcome.

So far, we have not yet assessed how difficult it is computationally to solve the outcome optimization problem, *i.e.* to select the optimal outcome according to the given criteria based on the preferences submitted by the agents, in any of these settings. This is the topic of the next chapter.

Chapter 3

Outcome Optimization

The more alternatives, the more difficult the choice.

Abbé D'Allanival

Perhaps surprisingly, expressive preference aggregation has only recently started to be adopted in practice (mostly in the form of combinatorial auctions and combinatorial reverse auctions). One of the reasons for this is that in expressive preference aggregation protocols, the problem of choosing the best outcome once the preferences have been collected is typically computationally hard. For instance, in combinatorial auctions, the winner determination problem is NP-complete [Rothkopf *et al.*, 1998] (even to approximate [Sandholm, 2002a]). The computational tools for solving such outcome selection problems have only recently reached the maturity necessary to make some of these protocols feasible in practice [Sandholm *et al.*, 2006; Sandholm, 2006].

This chapter will discuss the outcome optimization problem for the domains discussed in Chapter 2. While computing the aggregated ranking is easy under most voting rules, there are some voting rules for which this is not the case [Bartholdi *et al.*, 1989b; Hemaspaandra *et al.*, 1997; Cohen *et al.*, 1999; Dwork *et al.*, 2001; Rothe *et al.*, 2003; Davenport and Kalagnanam, 2004; Ailon *et al.*, 2005], including the Kemeny and Slater rules. In Section 3.1, we introduce a powerful preprocessing technique for computing optimal rankings according to the Slater rule [Conitzer, 2006]. In Section 3.2, we give new conditions under which the winner determination problem in combinatorial auctions can be solved efficiently [Conitzer *et al.*, 2004]. In Section 3.3, we study computing optimal outcomes under the framework for negotiating over donations introduced in Section 2.3 [Conitzer and Sandholm, 2004e]. Finally, in Section 3.4 we study computing optimal outcomes under the framework for negotiating in settings with externalities introduced in Section 2.4 [Conitzer and Sandholm, 2005d].

3.1 A preprocessing technique for computing Slater rankings

This section describes work done at IBM Research, with guidance and advice from Andrew Davenport and Jayant Kalagnanam.

In voting, we would like our aggregate ranking to be consistent with the outcome of each pairwise election: if more voters prefer a to b than vice versa, the aggregate ranking should rank a ahead of b. Unfortunately, Condorcet cycles can prevent us from being able to achieve this consistency for all pairs of candidates. The *Slater* voting rule is arguably the most straightforward resolution to this problem: it simply chooses a ranking of the candidates that is inconsistent with the outcomes of as few pairwise elections as possible. Unfortunately, as we will discuss later in this section, computing a Slater ranking is NP-hard. This in stark contrast to the computational ease with which most voting rules can be executed (most rules require only a simple computation of a score for each candidate, or perhaps one score per candidate for every round of the rule). An exception is the related Kemeny rule, which is also NP-hard to compute (in fact, as we will show later in this section, this hardness is implied by the Slater rule's hardness). The hardness of computing Slater ranking suggests using a tree search-based algorithm to compute Slater rankings.[1]

In this section, we introduce a *preprocessing* technique that can reduce the size of an instance of the Slater problem before the search is started. We say that a subset of the candidates consists of *similar candidates* if for every candidate outside of the subset, all candidates inside the subset achieve the same result in the pairwise election against that candidate. Given a set of similar candidates, we can recursively solve the Slater problem for that subset, and for the original set with the entire subset replaced by a single candidate, to obtain a solution to the original Slater problem. In addition, we also make the following contributions:

- We show that if the results of the pairwise elections have a particular hierarchical structure, the preprocessing technique is sufficient to solve the Slater problem in linear time.
- For the general case, we give a polynomial-time algorithm for *finding* a set of similar candidates (if it exists). This algorithm is based on satisfiability techniques.
- We exhibit the power of the preprocessing technique experimentally.
- We use the concept of a set of similar candidates to give the first straightforward reduction (that is, not a randomized reduction or a derandomization thereof) showing that the Slater problem is NP-hard in the absence of pairwise ties.

3.1.1 Definitions

Recall that the Slater rule is defined as follows: find a ranking $\succ$ on the candidates that minimizes the number of ordered pairs (a, b) such that $a \succ b$ but b defeats a in their pairwise election. (Equivalently, we want to maximize the number of pairs of candidates for which $\succ$ *is* consistent with the result of the pairwise election—we will refer to this number as the *Slater score*.) We will refer to the problem of computing a Slater ranking as the Slater problem. An instance of the Slater problem can be represented by a "pairwise election" graph whose vertices are the candidates, and which has a directed edge from a to b if and only if a defeats b in their pairwise election. The goal, then, is to minimize the number of edges that must be flipped in order to make the graph acyclic.

Most elections do not have any ties in pairwise elections. For example, if the number of votes is odd, there is no possibility of a pairwise tie. (We note that in many real-world elections, the number

[1] Another approach is to look for rankings that are *approximately* optimal in the Slater sense [Ailon *et al.*, 2005; Coppersmith *et al.*, 2006]. Of course, this is not entirely satisfactory as it is effectively changing the voting rule.

of voters is intentionally made odd to prevent ties.) Hence, we will restrict our attention to elections without pairwise ties (in which case the pairwise election graph becomes a tournament graph). For our positive results, this is merely for simplicity—they can easily be extended to deal with ties as well. Our one negative result, the NP-hardness of computing Slater rankings, is made stronger by this restriction (in fact, without the restriction the hardness has effectively been known for a long time).

3.1.2 Sets of similar candidates

We are now ready to give the formal definition of a set of similar candidates.

Definition 9 *We say that a subset $S \subseteq C$ consists of* similar candidates *if for any $s_1, s_2 \in S$, for any $c \in C - S$, $s_1 \to c$ if and only if $s_2 \to c$ (and hence $c \to s_1$ if and only if $c \to s_2$).*

We emphasize that in this definition, it is *not* required that *every* vote prefers s_1 to c if and only if that vote prefers s_2 to c. Rather, the condition only needs to hold on the aggregated pairwise election graph, and hence it is robust to a few voters who do not perceive the candidates as similar.

There are a few trivial sets of similar candidates: 1. the set of all candidates, and 2. any set of at most one candidate. We will be interested in nontrivial sets of similar candidates, because, as will become clear shortly, the trivial sets have no algorithmic use.

The following is the key observation of this section:

Theorem 1 *If S consists of similar candidates, then there exists a Slater ranking $\succ$ in which the candidates in S form a* (contiguous) block *(that is, there do not exist $s_1, s_2 \in S$ and $c \in C - S$ such that $s_1 \succ c \succ s_2$).*

Proof: Consider any ranking $\succ_1$ of the candidates in which the candidates in S are split into $k > 1$ blocks; we will show how to transform this ranking into another ranking $\succ_2$ with the properties that:

- the candidates in S are split into $k - 1$ blocks in $\succ_2$, and
- the Slater score of $\succ_2$ is at least as high as that of $\succ_1$.

By applying this transformation repeatedly, we can transform the original ranking into an ranking in which the candidates in S form a single block, and that has at least as high a Slater score as the original ranking.

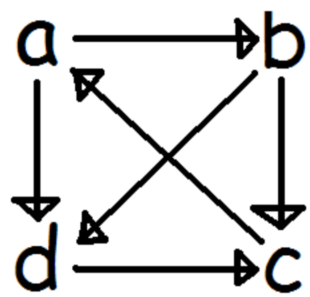
Consider a subsequence of $\succ_1$ consisting of two blocks of candidates in S, $\{s_i^1\}$ and $\{s_i^2\}$, and a block of candidates in $C - S$ that divides them, $\{c_i\}$: $s_1^1 \succ_1 s_2^1 \succ_1 \ldots \succ_1 s_{l_1}^1 \succ_1 c_1 \succ_1 c_2 \succ_1 \ldots \succ_1 c_l \succ_1 s_1^2 \succ_1 s_2^2 \succ_1 \ldots \succ_1 s_{l_2}^2$. Because S consists of similar candidates, a given candidate c_i has the same relationship in the pairwise election graph to *every* s_i^j. Hence, one of the following two cases must apply:

1. For at least half of the candidates c_i, for every s_i^j, $c_i \to s_i^j$
2. For at least half of the candidates c_i, for every s_i^j, $s_i^j \to c_i$.

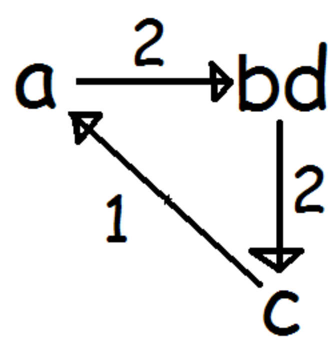
In case 1, we can replace the subsequence by the subsequence $c_1 \succ_2 c_2 \succ_2 \ldots \succ_2 c_l \succ_2 s_1^1 \succ_2 s_2^1 \succ_2 \ldots \succ_2 s_{l_1}^1 \succ_1 s_1^2 \succ_2 s_2^2 \succ_2 \ldots \succ_2 s_{l_2}^2$ to join the blocks without any loss to the Slater score of the ranking. Similarly, in case 2, we can replace the subsequence by the subsequence $s_1^1 \succ_2 s_2^1 \succ_2 \ldots \succ_2 s_{l_1}^1 \succ_1 s_1^2 \succ_2 s_2^2 \succ_2 \ldots \succ_2 s_{l_2}^2 \succ_2 c_1 \succ_2 c_2 \succ_2 \ldots \succ_2 c_l$ to join the blocks without any loss to the Slater score of the ranking. ■

Hence, if we know that S consists of similar candidates, then when we try to compute a Slater ranking, we can without loss of generality restrict our attention to rankings in which all the candidates in S form a (contiguous) block. The optimal internal ranking of the candidates in S within the block is independent of the rest of the ranking, and can be computed recursively.[2] Because of this, we can think of S as a single "super-candidate" with weight $|S|$. Ranking S above a candidate c such that $s \to c$ for all $s \in S$, or below a candidate c such that $c \to s$ for all $s \in S$, will increase the Slater score by $|S|$.

Consider the following pairwise election graph:

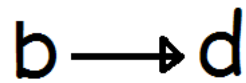


In this graph, $\{b, d\}$ is a set of similar candidates. Thus, we recursively solve the instance in which b and d are aggregated into a single candidate:



Some of the edges now represent multiple edges in the original graph; this is indicated by the weights on these edges. It is easy to see that the optimal Slater ranking for this graph is $a \succ bd \succ c$. In addition, we need to solve the Slater problem internally for the set of similar candidates:

[2]Note that if S is a trivial set of similar candidates, there is little use to this: if it is a set of at most one candidate, then the statement that that candidate will form a block by itself is vacuous, and if it is the set of all candidates, we need to recurse on the set of all candidates.



The optimal Slater ranking for this graph is (of course) $b \succ d$. So the solution to the original problem is $a \succ b \succ d \succ c$.

It is possible to have multiple disjoint sets S_i, each consisting of similar candidates. In this case, we can aggregate each one of them into a single super-candidate. The following lemma will clarify how to compute the Slater scores for such pairs of super-candidates:

Lemma 1 *If S_1 and S_2 are disjoint sets of similar candidates, then for any $s_1, s_1' \in S_1$ and any $s_2, s_2' \in S_2$, $s_1 \rightarrow s_2$ if and only if $s_1' \rightarrow s_2'$. (That is, the same relationship holds in the pairwise election graph for any pair of candidates in $S_1 \times S_2$.) Hence, ranking super-candidate S_1 above super-candidate S_2 such that $s_1 \rightarrow s_2$ for all $s_1 \in S_1, s_2 \in S_2$, or below a super-candidate S_2 such that $s_2 \rightarrow s_1$ for all $s_1 \in S_1, s_2 \in S_2$, will increase the Slater score by $|S_1| \cdot |S_2|$.*

Proof: Because S_1 consists of similar candidates, $s_1 \rightarrow s_2$ if and only if $s_1' \rightarrow s_2$. And, because S_2 consists of similar candidates, $s_1' \rightarrow s_2$ if and only if $s_1' \rightarrow s_2'$. ∎

Similar sets that overlap cannot be simultaneously turned into super-candidates. However, the following lemma shows that turning one of them into a super-candidate will (in a sense) preserve the structure of the other set: after aggregating one of the sets into a super-candidate, the other set will, in a sense, coincide with the union of the two sets, and we now show that this union must consist of similar candidates.

Lemma 2 *If S_1 and S_2 each consist of similar candidates, and $S_1 \cap S_2$ is nonempty, then $S_1 \cup S_2$ consists of similar candidates.*

Proof: Let $s \in S_1 \cap S_2$. For any $s', s'' \in S_1 \cup S_2$ and any $c \in C - (S_1 \cup S_2)$, we have that $s' \rightarrow c$ if and only if $s \rightarrow c$, and $s \rightarrow c$ if and only if $s'' \rightarrow c$. ∎

3.1.3 Hierarchical pairwise election graphs can be solved in linear time

In this subsection, we show that if the pairwise election graph has a certain hierarchical structure, then the Slater problem can be solved efficiently using the techniques from the previous subsection.

Definition 10 *A* valid candidate tree *is a tree with the following properties:*

- *The leaves are each labeled with a candidate, with each candidate appearing exactly once.*
- *For every internal vertex v, there is a tournament graph $\rightarrow_v$ over its children such that for any two distinct children $w_1 \neq w_2$ of v, for any descendants d_1 of w_1 and d_2 of w_2, $d_1 \rightarrow d_2$ if and only if $w_1 \rightarrow_v w_2$.*

Put alternatively, to find out the direction of the edge between any two candidates in the pairwise election graph, we can simply compare the vertices directly below their least common ancestor. There is always a trivial valid candidate tree, which simply connects every candidate directly to the root node R and uses the pairwise election graph $\rightarrow$ as the graph $\rightarrow_R$. This tree does not give us any insight. Instead, we will be interested in trees whose vertices have small degree (that is, each vertex has only a few children).

Figure 3.1 shows an example candidate tree, and Figure 3.2 shows the corresponding graph of pairwise elections.

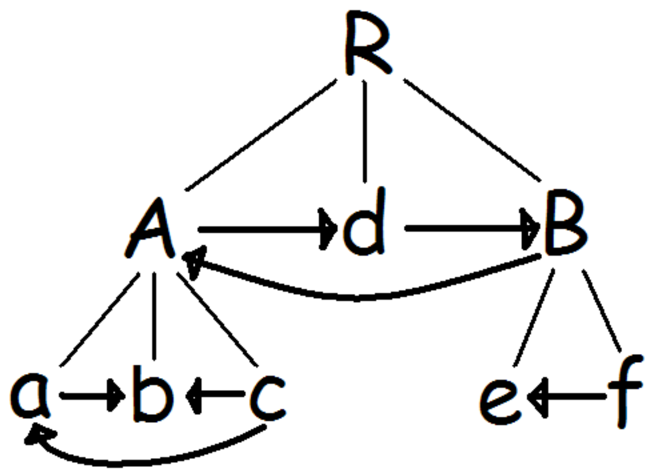


Figure 3.1: A valid candidate tree.

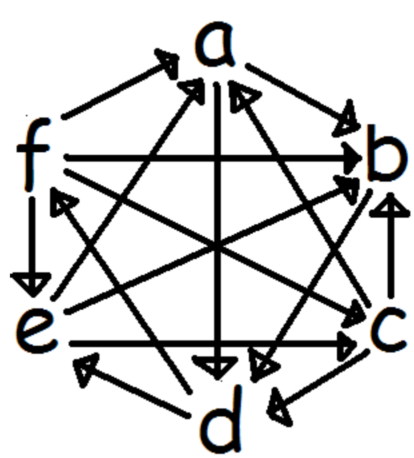


Figure 3.2: The pairwise election graph corresponding to the valid candidate tree.

The following observation will allow us to use the structure of the tree to solve the Slater problem efficiently:

Lemma 3 *For any vertex v in a valid candidate tree, the set D_v of candidates that are descendants* *of v constitutes a set of similar candidates.*

Proof: For any $d_1, d_2 \in D_v$ and $c \in C - D_v$, the least common ancestor of d_1 and c, or of d_2 and c, must be a (strict) ancestor of v. Hence, this least common ancestor must be the same in both cases, and moreover, so must the child of that ancestor from which d_1 and d_2 (and v) descend. Hence $d_1 \to c$ if and only if $d_2 \to c$. ∎

Hence, we can solve the Slater problem using the following very simple algorithm:

1. For every child of the root R, generate a super-candidate with weight equal to the number of candidates that descend from it.

2. Solve the Slater problem for the graph $\to_R$ over these super-candidates (using any algorithm).

3. Solve the Slater problem recursively for each subtree rooted at a child of the root R.

4. Order the candidates, first according to the ranking of the super-candidates that they are in, and then according to the recursive solutions.

Step 2 may be computationally expensive, depending on the number of super-candidates. However, if the degree of each vertex is small, then so is the number of super-candidates in this step. In particular, if the degree is bounded by a constant, then step 2 can be performed in constant time, and the running time of the entire algorithm is linear.

The algorithm produces the Slater ranking $f \succ e \succ c \succ a \succ b \succ d$ on the example given above.

3.1.4 An algorithm for detecting sets of similar candidates

In general, we do not know in advance whether there is a nontrivial set of similar candidates in a pairwise election graph. Rather, we need an algorithm that will take as input a pairwise election graph, and discover a nontrivial set of similar candidates if it exists. In this subsection, we present such an algorithm. The algorithm relies on transforming the problem of detecting a set of similar candidates into a Horn satisfiability problem.

Specifically, for every candidate c we generate a variable $In(c)$ which indicates whether the candidate is in the set of similar candidates. Then, for every ordered triplet of candidates $c_1, c_2, c_3 \in C$, if either $c_1 \to c_3$ and $c_3 \to c_2$, or $c_2 \to c_3$ and $c_3 \to c_1$, then we generate the clause $In(c_1) \wedge In(c_2) \Rightarrow In(c_3)$ (or, equivalently, $(\neg In(c_1) \vee \neg In(c_2) \vee In(c_3))$.

The instance described two subsections earlier produces the following clauses: $In(a) \wedge In(b) \Rightarrow In(c)$, $In(a) \wedge In(c) \Rightarrow In(b) \wedge In(d)$, $In(a) \wedge In(d) \Rightarrow In(b) \wedge In(c)$, $In(b) \wedge In(c) \Rightarrow In(a) \wedge In(d)$, $In(c) \wedge In(d) \Rightarrow In(a)$.

Theorem 2 *A setting of the variables $In(c)$ satisfies all the clauses if and only if $S = \{c \in C : In(c) = \text{true}\}$ consists of similar candidates.*

Proof: First, suppose that the setting satisfies all the clauses. For any $s_1, s_2 \in S$ and $c \in C - S$, if there were a clause $In(s_1) \wedge In(s_2) \Rightarrow In(c)$, it would not be satisfied. It follows that this clause was not generated, and hence either $s_1 \rightarrow c$ and $s_2 \rightarrow c$, or $c \rightarrow s_1$ and $c \rightarrow s_2$. Hence S consists of similar candidates.

Next, suppose that the setting does not satisfy all the clauses. Then, there must be some unsatisfied clause $In(c_1) \wedge In(c_2) \Rightarrow In(c_3)$, which means that $c_1, c_2 \in S$ and $c_3 \notin S$. Because the clause was generated, either $c_1 \rightarrow c_3$ and $c_3 \rightarrow c_2$, or $c_2 \rightarrow c_3$ and $c_3 \rightarrow c_1$, and hence S does not consist of similar candidates. ■

There are some settings of the variables $In(c)$ that always satisfy all the clauses: setting everything to *true*, and setting at most one variable to *true*. These settings correspond exactly to the trivial sets of similar candidates discussed earlier. Hence, our goal is to find a satisfying setting of the variables in which at least two, but not all, variables are set to *true*. In the example above, the only such solution is to set $In(b)$ and $In(d)$ to *true* and $In(a)$ and $In(c)$ to *false*, corresponding to the set of similar candidates that we used earlier in this section. Finding a nontrivial solution can be done in polynomial time with the following simple algorithm: for a given pair of candidates $c_1, c_2 \in C$, set $In(c_1)$ and $In(c_2)$ to *true*, and then follow the implications $\Rightarrow$ in the clauses. If this process terminates without setting all the $In(c)$ variables to *true*, we have found a nontrivial set of similar candidates. Otherwise, restart with a different pair of candidates, until we have tried every pair of candidates. The algorithm can be improved by keeping track of the initial pairs of candidates for which we have failed to find a similar set, so that when another initial pair leads to one of these pairs being set to *true*, we can fail immediately and continue to the next pair.

When we use this algorithm for finding similar sets to help us compute a Slater ranking, after finding a similar set, we need to compute a Slater ranking both on the instance consisting of the similar set only, and on the reduced version of the original instance where the similar set has been replaced by a single super-candidate. Thus, we will be interested in finding similar sets on these instances as well. It is natural to ask whether some of the computation that we did to find a similar set in the original instance can be reused to find similar sets in the two new instances. It turns out that, in the second new instance, this is indeed possible:

Lemma 4 *Suppose that, in the process of detecting a similar set, we failed with the pair of initial candidates $c_1, c_2 \in C$, before discovering that $S \subseteq C$ is a similar set. Then, in the reduced instance where S is replaced by a single super-candidate c_S,*

1. *we will also fail on initial pair c_1, c_2 if $c_1, c_2 \notin S$;*
2. *we will also fail on initial pair c_1, c_S if $c_1 \notin S, c_2 \in S$.*

Proof: Suppose c_1, c_2 (or, in the second case, c_1, c_S) belong to a nontrivial set S' of similar candidates in the reduced instance. Then, consider the same set in the original instance (if $c_S \in S'$, replace it with all members of S); call this set S''. S'' does not include all candidates because S' does not include all candidates in the reduced instance. Moreover, S'' is a set of similar candidates, for the following reasons. Take any $s_1, s_2 \in S''$ and $c \in C - S''$; we must show that $s_1 \rightarrow c$ if and only if $s_2 \rightarrow c$. If $s_1 \notin S$ and $s_2 \notin S$, then this follows from the fact that S' consists of similar candidates (even if $c \in S$, because in that case $s_i \rightarrow c$ if and only if $s_i \rightarrow c_S$ in the reduced instance). If exactly one of s_1 and s_2 is in S (without loss of generality, say $s_1 \in S$), then it must be that $c_S \in S' \Leftrightarrow S \subseteq S''$, so that $c \notin S$. Hence, $s_1 \to c$ if and only if $c_s \to c$, and because S' consists of similar candidates this is true if and only if $s_2 \to c$. Finally, if both s_1 and s_2 are in S, then $c \notin S$ because $c_S \in S' \Leftrightarrow S \subseteq S''$, hence $s_1 \to c$ if and only if $s_2 \to c$ because S consists of similar candidates. But if S'' consists of similar candidates, then we would not have failed on the initial pair c_1, c_2 in the original instance. Hence we have the desired contradiction. ■

For the first new instance consisting of S only, such reuse of computation is not possible, because we cannot have failed on a pair of candidates within S (since they were in fact in a similar set). We only know that we will fail on the pair starting with which we found S, because this pair will lead to all candidates in S being included in the similar set.

3.1.5 Experimental results

In this subsection, we experimentally evaluate the use of the techniques described above as a preprocessing technique that serves to reduce the sizes of the instances before a search algorithm is called. We compare two algorithms: a straightforward search technique, and the preprocessing technique combined with the same search technique. (Instead of the straightforward search technique, we could also have used a more sophisticated algorithm, *e.g.* a commercial solver such as CPLEX. The goal here, however, is not to show that our technique by itself outperforms any given algorithm, but rather it is to show that using the preprocessing technique can reduce a given algorithm's computation time. Moreover, if adding the preprocessing technique to a straightforward search algorithm already produces a fast algorithm, that is more impressive than having to add the technique to a solver such as CPLEX to obtain a fast algorithm.) The straightforward search technique decides, at every search node, whether a given edge in the graph should be consistent or inconsistent with the final ranking, and then propagates the effect of this decision to other edges (*e.g.* by transitivity, if it has been decided that edges (a, b) and (b, c) will both be consistent with the final ranking, then by transitivity so must the edge (b, c)). There is no sophisticated variable ordering heuristic: we simply choose the first edge that has not yet been set. As an admissible pruning heuristic, we use the total number of edges for which it has been decided that the final ranking will be inconsistent with them.

The preprocessing technique uses the algorithm described in the previous subsection to search for a set of similar candidates. If it finds one, it recursively solves the subproblems; otherwise, the search algorithm is called to solve the remaining irreducible problem instance. Each data point in the experiments is an average of 30 instances (the same instances for both algorithms).

In the first set of experiments, instances are generated as follows. Every candidate and every voter draws a random position in $[0, 1]$ (this can be thought of as their stance on one issue) and voters rank candidates by proximity to their own position. The results are in Figure 3.3:

On these instances, even straightforward search scales reasonably well, but when the preprocessing technique is added, all the instances solve immediately. This is not surprising: the voters' preferences in this domain are *single-peaked*, and it is well-known that for single-peaked preferences, there are no cycles in the pairwise election graph (*e.g.* [Mas-Colell *et al.*, 1995]), so that the final ranking can be read off directly from the graph. Given this, any k candidates in contiguous positions in the final ranking always form a set of similar candidates, so that the preprocessing technique can solve the instances entirely. (No time is spent in search after the preprocessing technique.)

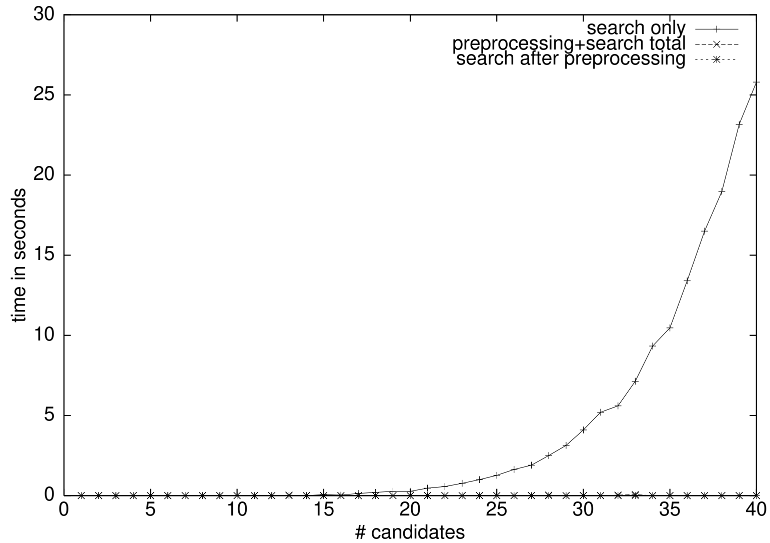


Figure 3.3: 1 issue, 191 voters, 30 instances per data point.

Of course, we do not want to examine only trivial instances. In the next experiment (Figure 3.4), the candidates and voters draw random positions in $[0, 1] \times [0, 1]$; in this two-dimensional setup the voters' preferences are no longer single-peaked.

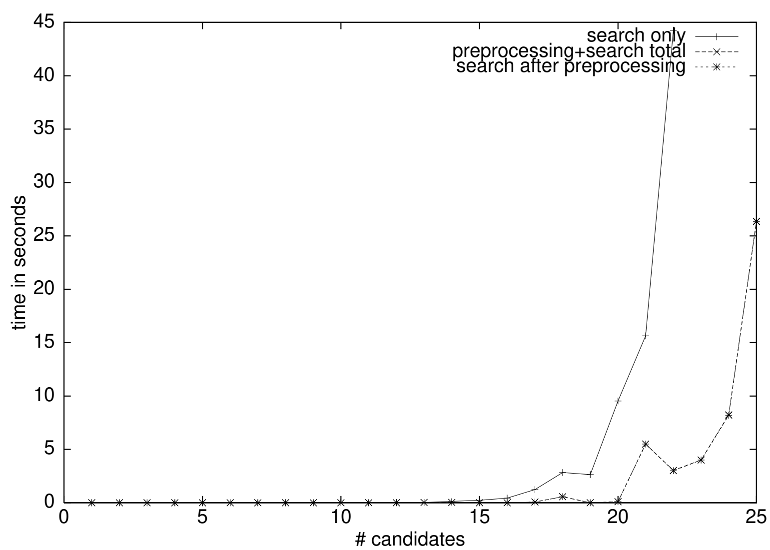


Figure 3.4: 2 issues, 191 voters, 30 instances per data point.

These instances are much harder to solve, but adding the preprocessing technique significantly speeds up the search. We note that essentially no time is spent in the preprocessing stage (the "preprocessing + search total" and "search after preprocessing" curves are essentially identical), hence the benefits of preprocessing effectively come for free.

We also considered changing the number of votes to a small number. Figure 3.5 shows the results with only 3 votes.

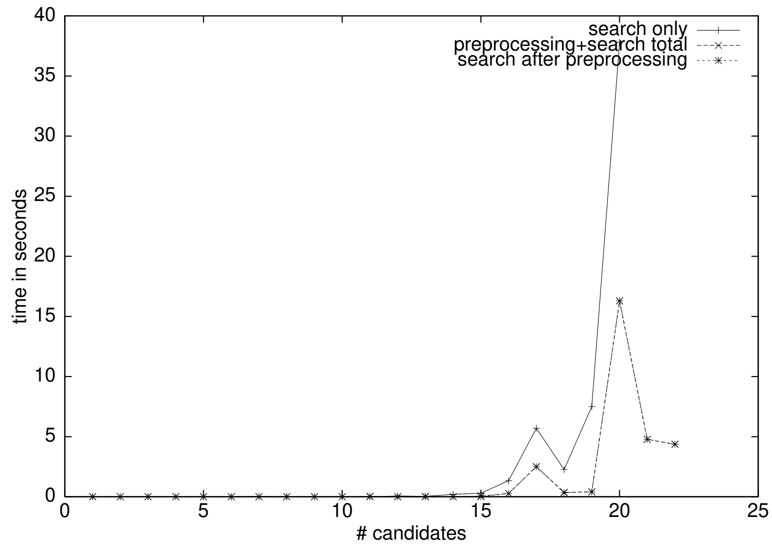


Figure 3.5: 2 issues, 3 voters, 30 instances per data point.

We experimented with introducing additional structure on the set of candidates. In the next experiment, there are 5 *parties* that draw random positions in $[0, 1] \times [0, 1]$; each candidate randomly chooses a party, and then takes a position that is the average of the party's position and another random point in $[0, 1] \times [0, 1]$. The results did not change significantly, as shown in Figure 3.6.

We also experimented with having the voters and candidates draw positions on an even larger number of issues (10 issues). Perhaps surprisingly, here the preprocessing technique once again solved all instances immediately (Figure 3.7).

3.1.6 NP-hardness of the Slater problem

In this subsection, we use the technique of sets of similar candidates in an entirely different manner: we show that it is useful in demonstrating the hardness of the Slater problem when there are no pairwise ties. In the case where pairwise ties between candidates are possible, the hardness of the Slater problem follows from the hardness of the Minimum Feedback Edge Set problem. However, as we have already pointed out, most elections do not have pairwise ties (for example, if the number of votes is odd, then there cannot be any pairwise ties). So, how hard is the problem when there are no ties? This problem is equivalent to the Minimum Feedback Edge Set problem on tournament graphs, and was conjectured to be NP-hard as early as 1992 [Bang-Jensen and Thomassen, 1992]. The conjecture remained unproven until 2005, when a randomized reduction was given [Ailon *et al.*, 2005]. A later derandomization of this proof finally proved the conjecture completely [Alon, 2006]. Interestingly, the observations about sets of similar candidates made above allow us to give

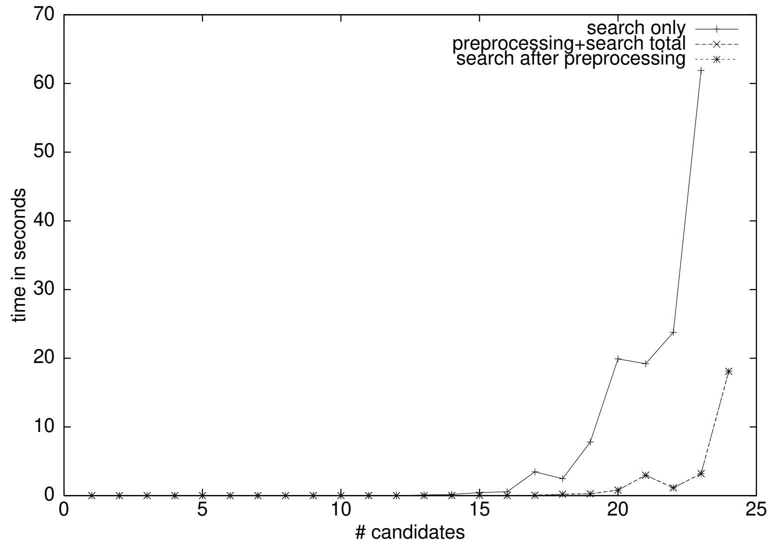


Figure 3.6: 2 issues, 5 parties, 191 voters, 30 instances per data point.

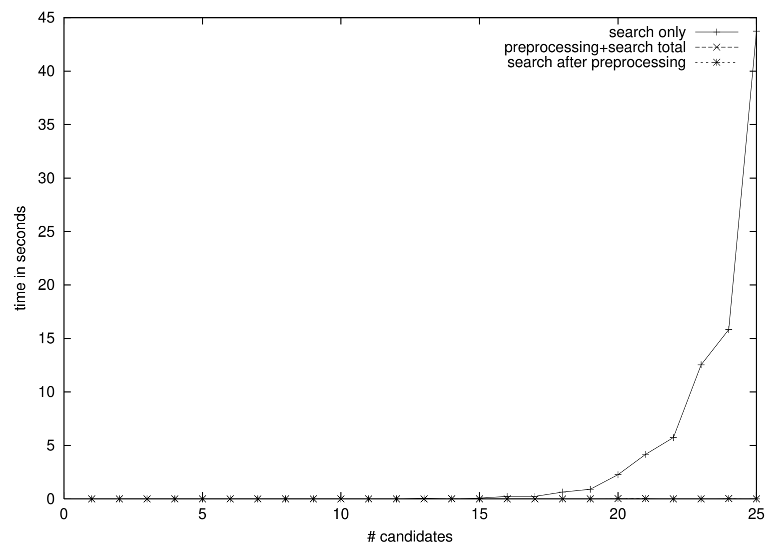


Figure 3.7: 10 issues, 191 voters, 30 instances per data point.

a more direct proof of this result (which does not rely on derandomizing a randomized reduction).[3]

Theorem 3 *The* Slater *problem is NP-hard (even in the absence of pairwise ties).*

[3]Interestingly, the previous reductions [Ailon *et al.*, 2005; Alon, 2006] also use what is effectively an extremely special case of the results about similar candidates presented in this section. That special case is, however, not sufficient for the reduction given here.

Proof: We reduce from the NP-complete Maximum Satisfiability (MAXSAT) problem. We show how to reduce an arbitrary MAXSAT instance, given by a set of clauses K over a set of variables V, and a target number t_1 of clauses to satisfy, to an instance of the Slater problem and a target score t_2, such that there is an ranking with Slater score at least t_2 if and only if there is a solution to the MAXSAT instance (that satisfies at least t_1 clauses). Let M be a sufficiently large number ($M > 6|K||V| + |K|^2$). For every variable $v \in V$, let there be the following 6 super-candidates, each of size M (that is, representing M individual candidates): $C_v = \{a_v, +_v, -_v, b_v, d_v, e_v\}$.[4] Let the individual candidates that any given single super-candidate represents constitute an acyclic pairwise election graph, so that we can order them perfectly and obtain a Slater score of $M(M-1)/2$. Let the super-candidates have the following relationships to each other in the aggregated graph:

- Fix some order $>$ over the variables (e.g. $x_1 > x_2 > \ldots > x_{|V|}$). Then, for any two super-candidates $c_v \in C_v, c_{v'} \in C_{v'}$ with $v > v'$, $c_v \to c_{v'}$.
- For any $v \in V$, for any $c_v \in \{a_v, +_v, -_v\}$ and $c'_v \in \{b_v, d_v, e_v\}$, $c_v \to c'_v$.
- For any $v \in V$, $a_v \to +_v, +_v \to -_v, -_v \to a_v$; $b_v \to d_v, b_v \to e_v, d_v \to e_v$.

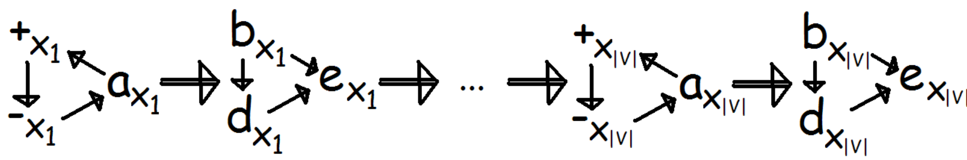


Figure 3.8: Illustration of part of the reduction.

Additionally, for every clause $k \in K$, let there be a single candidate (not a super-candidate) c_k, with the following relationships to the candidates corresponding to variables. Assume without loss of generality that opposite literals ($+v$ and $-v$) never occur in the same clause. Then,

- If $+v \in k$, then $+_v \to c_k, d_v \to c_k, e_v \to c_k$ and $c_k \to a_v, c_k \to -_v, c_k \to b_v$.
- If $-v \in k$, then $-_v \to c_k, d_v \to c_k, e_v \to c_k$ and $c_k \to a_v, c_k \to +_v, c_k \to b_v$.
- If $\{+v, -v\} \cap k = \emptyset$, then $b_v \to c_k, d_v \to c_k, e_v \to c_k$ and $c_k \to a_v, c_k \to +_v, c_k \to -_v$.

The relationships among the c_k are irrelevant. Finally, let the target Slater score be $t_2 = 6|V|M(M-1)/2 + 36M^2|V|(|V|-1)/2 + 14M^2|V| + t_1M$.

We now make some observations about the Slater problem instance that we have constructed. First, by Theorem 1, we can restrict our attention to rankings in which the individual candidates in any given super-candidate form a (contiguous) block. Recall that within such a block, we can order the individual candidates to obtain a Slater score of $M(M-1)/2$, which will give us a total of $6|V|M(M-1)/2$ points. Now, if our ranking of two super-candidates is consistent with the pairwise election graph, according to Lemma 1 this will increase the Slater score by M^2. By contrast, the total number of Slater points that we can obtain from *all* the edges in the pairwise

[4]The letter c is skipped only to avoid confusion with the use of c as an arbitrary candidate.

election graph that involve a candidate c_k corresponding to a clause is at most $|K| \cdot 6|V|M + |K|^2 < 6|K||V|M + |K|^2 M < M^2$. Hence, it is never worth it to sacrifice an agreement on an edge involving two super-candidates to obtain a better result with regard to the remaining candidates, and therefore we can initially restrict our attention to the super-candidates only as these are our primary concern. It is clear that for $v > v'$ we should rank all the candidates in C_v above all those in $C_{v'}$. Doing this for all variables will increase the Slater score by $36M^2|V|(|V|-1)/2$. Moreover, it is clear that for every v we should rank all the candidates in $\{a_v, +_v, -_v\}$ above all those in $\{b_v, d_v, e_v\}$, and $b_v \succ d_v \succ e_v$. Doing this for all variables will increase the Slater score by $(9M^2 + 3M^2)|V| = 12M^2|V|$. Finally, for every v, any one of the rankings $+_v \succ -_v \succ a_v$, $-_v \succ a_v \succ +_v$, and $a_v \succ +_v \succ -_v$ are equally good, leaving us a choice. Choosing one of these for all variables increases the Slater score by another $2M^2|V|$.

Now, as a secondary concern, we can analyze edges involving the c_k. Agreement on an edge between a c_k and one of the super-candidates will increase the Slater score by M. By contrast, the total number of Slater points that we can obtain from *all* the edges in the pairwise election graph that involve only candidates c_k is at most $|K|(|K|-1)/2 < |K|^2 < M$. Hence, it is never worth it to sacrifice an agreement on an edge involving a super-candidate to obtain a better result with regard to the edges involving only candidates c_k, and hence we can restrict our attention to edges involving a super-candidate. (In fact, the edges involving only candidates c_k will turn out to have such a minimal effect on the total score that we need not consider them at all.) Now, we note that whether a candidate c_k is ranked before *all* the candidates in C_v or after *all* of them makes no difference to the total score, because three of these candidates will have an edge into c_k, and three of them will have have an edge out of c_k. Nevertheless, a candidate c_k could be ranked *among* the candidates C_v for (at most) one $v \in V$. Because d_v and e_v always have edges into c_k and are always ranked last among the candidates in C_v, ranking c_k after at least two of the candidates in C_v will never make a positive contribution to the Slater score. Hence, there are only two possibilities to increase the Slater score (by exactly M) for a given c_k: either rank c_k directly after some $+_v$ such that $+v \in k$ and $+_v$ is ranked first among the C_v, or rank c_k directly after some $-_v$ such that $-v \in k$ and $-_v$ is ranked first among the C_v. Of course, for each variable v, we can rank at most one of $+_v$ and $-_v$ first. (We can also rank a_v first, but this will never help us.) Now we can see that this corresponds to the MAXSAT problem: say that we set v to *true* if $+_v$ is ranked first, and to *false* if $-_v$ is ranked first. Then, we can obtain an additional M points for a candidate c_k if and only if clause k is satisfied, and hence we can increase the Slater score by an additional $t_1 M$ points if and only if we can set the variables in such a way as to satisfy at least t_1 clauses. ■

3.1.7 Extension to the Kemeny rule

The *Kemeny* rule [Kemeny, 1959] has significant similarities to the Slater rule. One definition of the Kemeny rule that makes this clear is the following. Instead of minimizing the number of pairwise elections that the final ranking disagrees with, the Kemeny rule tries to minimize the total *weight* of such pairwise elections—where the weight of a pairwise election is the number of votes by which its winner defeated its loser. This also shows that the Kemeny ranking problem is more general than the Slater ranking problem, because it is easy to create votes that make all weights equal to each other. (For example, adding the votes $c_1 \succ c_2 \succ \ldots \succ c_m$ and $c_m \succ c_{m-1} \succ \ldots \succ c_3 \succ c_1 \succ c_2$

gives c_1 two extra votes in its pairwise election against c_2, but has a neutral effect on every other pairwise election.)

The Kemeny rule has an important interpretation as a maximum likelihood estimator of the "correct" ranking. Condorcet, an early social choice theorist, modeled elections as follows: there is a correct ranking of the candidates, but every voter only has a noisy perception of this correct ranking. Specifically, for every pair of candidates, any voter ranks the better candidate higher with probability $p > 1/2$, independently.[5] Given this noise model, the problem of finding the maximum likelihood estimate of the correct outcome, given the votes, is well-defined. Condorcet solved this problem for the cases of 2 and 3 candidates [de Caritat (Marquis de Condorcet), 1785]. Over two centuries later, Young [1995] observed that the Kemeny rule is in fact the solution to Condorcet's problem for arbitrary numbers of candidates. Because of this, the Kemeny rule is sometimes also referred to as the Kemeny-Young rule. We recently showed that some, but not all, of the other well-known voting rules can also be interpreted as maximum likelihood estimators, under different (more complex) noise models [Conitzer and Sandholm, 2005a].

The techniques presented in this section can be extended to apply to the Kemeny rule as well. However, to apply to the Kemeny rule, the definition of a set of similar candidates must be modified to state that for any fixed candidate outside the set, all candidates inside the set must receive exactly the same number of votes in the pairwise election against that candidate (rather than merely obtain the same overall result). This modified definition is much less likely to apply than the original version.

This concludes the part of this dissertation studying the complexity of executing voting rules; we will return to voting, specifically to the hardness of *manipulating* elections, in a few chapters, in Section 8.2. In the next section, we study the complexity of the winner determination problem in combinatorial auctions.

3.2 Combinatorial auctions with structured item graphs

This section describes work that we did jointly with Jonathan Derryberry (CMU).

As we mentioned at the beginning of this chapter, the winner determination problem in a general combinatorial auction (CA) is NP-complete [Rothkopf *et al.*, 1998]. Three main approaches have been pursued to address this: 1) designing optimal search algorithms that are often fast (*e.g.*, [Sandholm, 2002a; Sandholm *et al.*, 2005c; Gonen and Lehmann, 2000; Fujishima *et al.*, 1999; Boutilier, 2002]), but require exponential time in the worst case (unless P = NP), 2) designing approximation algorithms (*e.g.*, [Hoos and Boutilier, 2000; Zurel and Nisan, 2001; van Hoesel and Müller, 2001])—but unfortunately no polytime algorithm can guarantee an approximation (unless ZPP = NP) [Sandholm, 2002a], and 3) designing optimal polytime algorithms for restricted classes of CAs (e.g., [Rothkopf *et al.*, 1998; Tennenholtz, 2000; Penn and Tennenholtz, 2000; Sandholm and Suri, 2003]).

[5]Of course, the rankings of pairs of candidates cannot actually be completely independent, because if a is preferred to b, and b to c, then a must be preferred to c. Nevertheless, all rankings receive some probability under this model, which is all that is necessary for the maximum likelihood approach.

This section falls roughly within the third approach: we present hardness and easiness results for natural classes of CAs. However, we also develop a problem instance parameter (treewidth of the item graph, described below), such that *any* CA instance falls within our framework, and winner determination complexity is exponential in the parameter only. Like almost all of the work on polynomial-time solvable combinatorial auctions, we restrict our attention to the case where any two bids on disjoint subsets can be simultaneously accepted—that is, no XOR-constraints between bids are allowed. (This can be circumvented by adding dummy items that encode these constraints, as discussed in Section 2.2. However, the additional dummy items may significantly increase the time required to solve the instance.)

Consider graphs with the auction's items as vertices, which have the property that for any bid, the items occurring in it constitute a connected set in the graph. (For instance, the fully connected graph (with an edge between every pair of items) always has this property.) Such graphs can have potentially useful structure (for example, the graph may be a tree). For any type of structure, one can ask the following two questions: 1) how hard is the clearing problem when we are given a valid item graph with the desired structure? 2) if the graph is not given beforehand, how hard is it to construct a valid item graph with this structure (if it exists)? We will investigate both questions. 1) was previously solved for the special case where the graph is a tree or a cycle [Sandholm and Suri, 2003]; 2) was previously solved for the special case where the graph is a line (as pointed out by Sandholm and Suri [2003], using a result by Korte and Mohring [1989]) or a cycle (as pointed out by Sandholm and Suri [2003], using a result by Eschen and Spinrad [1993]). In each of these cases, a low-order polynomial algorithm was presented.

One practical use of such polynomially detectable and solvable special cases is to incorporate them into optimal search algorithms [Sandholm and Suri, 2003]. At every node of the search tree, we can detect whether the remaining problem is polynomially solvable, and if it is, we use the polynomial special-purpose algorithm for solving it. Otherwise the search will continue into the subtree.

Also, there are two pure uses for graph structures which make questions 1 and 2 easy. First, the auctioneer can decide on the graph beforehand, and allow only bids on connected sets of items. (In this case, 1 is most important, but 2 may also be useful if the auctioneer wants to make sure that bids on certain bundles are allowed.) Second, the auctioneer can allow bids on any bundle; then, once the bids have been submitted, attempt to construct an item graph that is valid for these bids; and finally, clear the auction using this graph. Clearly, the second approach is only practical if real instances are likely to have item graphs with some structure. To a lesser extent, this is also important for the first approach: if bidders must bid on bundles too different from their desired bundles, economic value will be lost.

Fortunately, real-world instances are likely to have graphs that are not fully connected. For instance, an item may receive only isolated bids that do not involve any other items; or an item may always co-occur with the same other item in bids. As a more detailed example, consider a combinatorial auction for tourist activities in the Bay Area. One item for sale may be a ticket to Alcatraz (in San Francisco). Another may be a ticket to the Children's Discovery Museum (in San Jose). A third item may be a Caltrain ticket to get back and forth between the two cities.[6] Supposing

[6]The reason that this example focuses on the Bay Area is that we published the corresponding paper in the proceedings of the AAAI-04 conference, which was held in San Jose.

(for now) that there is no alternative transportation between the two cities, the only bundle that is unlikely to receive a bid is {Alcatraz, Children's Discovery Museum}, because the bidder will need transportation (no matter which city she is based out of for the duration of her visit). Thus, a valid graph for this auction is the line graph in Figure 3.9 (because its only disconnected set is the one we just ruled out).

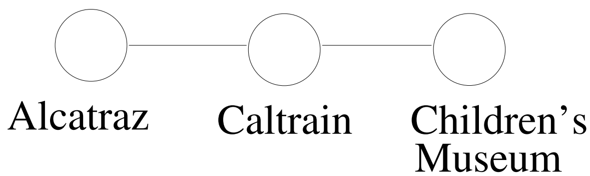


Figure 3.9: A valid item graph.

To extend the example, suppose that there are alternative modes of transportation which are also included in the auction: a Rental Car, and a Bus ticket. Now the following bundles are unlikely to receive a bid: {Alcatraz, Children's Discovery Museum} (because the bidder requires a form of transportation) and any bundle containing more than one mode of transportation. Thus, the graph in Figure 3.10 is a valid item graph (because we just ruled out all its disconnected sets). This graph

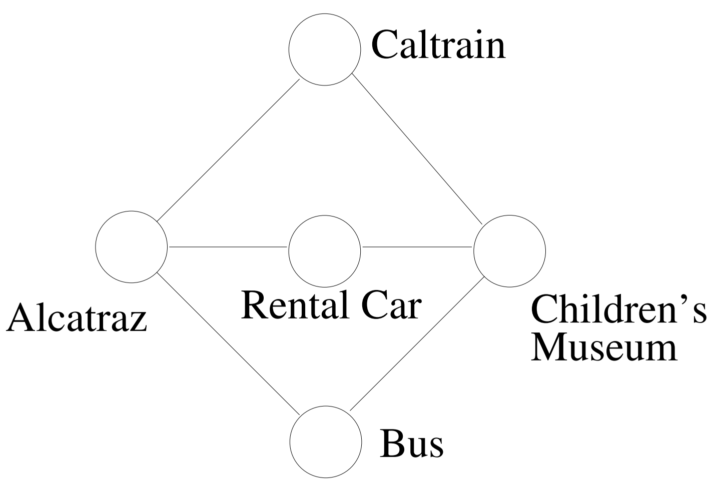


Figure 3.10: Another valid item graph.

does not fall under any of the previously studied structures (it is not a line, tree, or cycle). Still, it has interesting structure: for instance, it has a treewidth of 2.

The rest of this section is organized as follows. We first show that given an item graph with bounded treewidth, the clearing problem can be solved in polynomial time. Next, we show how to construct an item tree (treewidth = 1), if it exists, in polynomial time. This answers the proposed open question of whether this can be done [Sandholm and Suri, 2003]. (We leave open the question of whether an item graph with small treewidth (say, 2) can be constructed if it exists.) We show that constructing the item graph with the fewest edges is NP-complete (even when a graph of treewidth 2 is easy to construct). Finally, we study a variant where a bid is allowed to consist of k connected sets, rather than just one. We show that the clearing problem is NP-complete even for line graphs with $k = 2$, and the graph construction problem is NP-complete for line graphs with $k = 5$.

3.2.1 Item graphs

We first formally define item graphs.

Definition 11 *Given a combinatorial auction clearing problem instance, the graph $G = (I, E)$, whose vertices correspond to the items in the instance, is a* (valid) item graph *if for every bid, the set of items in that bid constitutes a connected set in G.*

We emphasize that an item graph, in our definition, does *not* need to have an edge connecting every pair of items that occurs in a bid. Rather, each pair only needs to be connected via a path consisting only of items in the bid. In other words, the subgraph consisting of each bid must form only one connected component, but it needs not be a clique.

3.2.2 Clearing with bounded treewidth item graphs

In this subsection, we show that combinatorial auctions can be cleared in polynomial time when an item graph with bounded treewidth is given. This generalizes a result by Sandholm and Suri [Sandholm and Suri, 2003] which shows polynomial time clearability when the item graph is a tree (treewidth = 1).[7] Linear-time approximation algorithms for clearing when the item graph has bounded treewidth have also been given, where the approximation ratio is the treewidth, plus one [Akcoglu *et al.*, 2002]. In contrast, we will clear the auction optimally.

First we will give a very brief review of treewidth.

Definition 12 *A* tree decomposition *T of a graph $G = (I, E)$ is a tree with the following properties.*

1. *Each $v \in T$ has an associated set I_v of vertices in G.*
2. *$\bigcup_{v \in T} I_v = I$ (each vertex of G occurs somewhere in T).*
3. *For each $(i_1, i_2) \in E$, there is some $v \in T$ with $i_1, i_2 \in I_v$ (each edge of G is contained within some vertex of T).*
4. *For each $i \in I$, $\{v \in T : i \in I_v\}$ is connected in T.*

We say that the width *of the tree is* $\max_{v \in T} |I_v| - 1$.

[7] The special case of a tree can also be solved in polynomial time using algorithms for perfect constraint matrices [de Vries and Vohra, 2003], but those algorithms are slower in practice.

While the general problem of finding a tree decomposition of a graph with minimum width is NP-complete, when the treewidth is bounded, the tree decomposition can be constructed in polynomial time [Areborg *et al.*, 1987]. Because we are only interested in the case where the treewidth of the item graph is bounded, we may assume that the tree decomposition is given to us as well as the graph itself.

The following (known) lemma will be useful in our proof.

Lemma 5 *If $X \subseteq I$ is a connected set in G, then $\{v \in T : I_v \cap X \neq \{\}\}$ is connected in T.*

We are now ready to present our first result.

Theorem 4 *Suppose we are given a combinatorial auction problem instance, together with a tree decomposition T with width tw of an item graph G. Then the optimal allocation can be determined in $O(|T|^2(|B|+1)^{tw+1})$ using dynamic programming, where B is the set of bids. (Both with and without free disposal.)*

Proof: Fix a root in T (the "top" of the tree). At every vertex $v \in T$ with items I_v, consider all functions $f : I_v \rightarrow B \cup \{0\}$, indicating possible assignments of the items to the bids. ($f(i) = 0$ indicates no commitment as to which bid item i is assigned to.) This set has size $(|B|+1)^{|I_v|}$. Consider the subset F_{I_v} of these functions satisfying: 1. If $f(i) = b$, b must bid on i; 2. All bids in the image $f(I_v)$ include items that occur higher up in T than v; 3. If $f(i) = b$ and b also bids on item $j \in I_v$, then $f(j) = b$ also.

The interpretation is that each function in F_{I_v} corresponds to a constraint from higher up in the tree as to which bids should be accepted. We now compute, for every node v (starting from the leaves and going up to the root), for every function $f \in F_{I_v}$, the maximum value that can be obtained from items that occur in v and its descendant vertices (but not in any other vertices), and that do not occur in bids in the image of f. (We observe that if v is the root node, there can be no constraints from higher up in the tree (that is, there is only one f function), and the corresponding value is the maximum value that can be obtained in the auction.) Denoting this value by $r(v, f)$, we can compute it using dynamic programming from the leaves up in the following manner:

- Consider all assignments $g : \{i \in I_v : f(i) = 0\} \rightarrow B \cup \{0\}$,[8] with the properties that: 1. If $g(i) = b$, b must bid on i; 2. The image of g does not include any bids that include items that occur higher in T than v. 3. If $g(i) = b$ and b also bids on item $j \in I_v$, then $f(j) = 0$ and $g(j) = b$ also. (Thus, g indicates which bids concerning the unallocated items in I_v we are considering accepting, but only bids that we have not considered higher in the tree.)

- The value of such an assignment is $\sum_{b \in g(I_v)} a(b) + \sum_{w \in T : p(w) = v} r(w, q_w(f, g))$, where $g(I_v)$ is the image of g, $a(b)$ is the value of bid b, $p(w)$ is the parent of w, and $q_w(f, g) : I_w \rightarrow B$ maps items occurring in a bid in the image of either f or g to that bid, and everything else to 0.

- The maximum such value over all g is $r(v, f)$.

[8] In the case of no free disposal, g cannot map to 0.

Because we need to do this computation once for each vertex v in T, the number of assignments g is at most $(|B|+1)^{|I_v|}$ where $|I_v| \leq tw+1$, and for each assignment we need to do a lookup for each of the children of v, this algorithm has running time $O(|T|^2(|B|+1)^{tw+1})$.

The allocation that the algorithm produces can be obtained going back down the tree, as follows: at the root node, there is only one constraint function f mapping everything to 0 (because no bid has items higher up the tree). Thus, consider the r-value maximizing assignment g_{root} for the root; all the bids in its image are accepted. Then, for each of its children, consider the r-value maximizing assignment under the constraint imposed by g_{root}; all the bids in the image of that assignment are also accepted, etc.

To show that the algorithm works correctly, we need to show that no bids that are accepted high up in the tree are "forgotten about" lower in the tree (and its items lower in the tree awarded to other bids). Because the items in a bid constitute a connected set in the item graph G, by Lemma 5, the vertices in T containing items from such a bid are also connected. Now, if a bid b is accepted at the highest vertex $v \in T$ containing an item in b (that is, the items in that vertex occurring in b are awarded to b), each of v's children must also award all its items occurring in b to b; and by the connectedness pointed out above, for each child, either there is at least one such an item in that child, or none of its descendants have any items occurring in b. In the former case, b is also in the image of the child's allocation function, and the same reasoning applies to its children, etc.; in the latter case the fact that b has been accepted is irrelevant to this part of the tree. So, either an accepted bid forces a constraint in a child, and the fact that the bid was accepted is propagated down the tree; or the bid is irrelevant to all that child's descendants as well, and can be safely forgotten about. ■

3.2.3 An algorithm for constructing a valid item tree

So far we discussed question 1: how to clear the auction *given* a valid item graph. In this subsection, we move on to the second question of *constructing* the graph. We present a polynomial-time algorithm that constructs an item tree (that is, an item graph without cycles), if one exists for the bids. This closes a recognized open research problem [Sandholm and Suri, 2003], and is necessary if one wants to use the polynomial item tree clearing algorithm as a subroutine of a search algorithm, as discussed in the introduction.

First, we introduce some notation. In a combinatorial auction with bid set B and item set I, define $items(b) \subseteq I$ to be the set of items in bid b. Also, let T_b refer to the subgraph of a tree containing only vertices represented by $items(b)$ and all edges among elements of $items(b)$.

With these definitions in hand, we are now ready to present the main theorem of this subsection. This theorem shows how to give a tree that "minimally violates" the key requirement of an item graph (namely, that each bid bids on only one component). Thus, if it is actually possible to give a valid item tree, such a tree will be produced by the algorithm.

Theorem 5 *Given an arbitrary set of bids B for items I, a corresponding tree T that minimizes*

$$\sum_{b \in B} \textit{the number of connected components in } T_b$$

can be found in $O(|B| \cdot |I|^2)$ time.

Proof: Consider the algorithm MAKETREE(B, I) shown below, which returns the maximum spanning tree of the complete undirected weighted graph over vertices I in which each edge (i, j) has a weight equal to the number of bids b such that $i, j \in items(b)$.

MAKETREE(B, I)

$A \leftarrow$ An $|I| \times |I|$ matrix of 0s
for each b **in** B
  do for each i **in** $items(b)$
    do for each $j \neq i$ **in** $items(b)$
      do $A(i, j) \leftarrow A(i, j) + 1$
return the maximum spanning tree of the graph A

The running time of MAKETREE(B, I) is $O(|B| \cdot |I|^2)$ from the triply nested **for** loops, plus the time needed to find the maximum spanning tree. The maximum spanning tree can be found in $O(|I|^2)$ time [Cormen *et al.*, 1990], so the running time of the algorithm as a whole is $O(|B| \cdot |I|^2) + O(|I|^2) = O(|B| \cdot |I|^2)$.

To see that MAKETREE(B, I) returns the tree T with the minimum sum of connected components across all T_b, note that the total weight of T can be written as

$$\sum_{b \in B} \text{the number of edges in } T \text{ among } items(b).$$

Because T is a tree, each subgraph T_b is a forest, and the number of edges in any forest equals the number of vertices in the forest, minus the number of components in the forest. It follows that we can rewrite the above expression as

$$s = \sum_{b \in B} |items(b)| - \text{ the number of components in } T_b.$$

Because $\sum_{b \in B} |items(b)|$ is a constant, maximizing s is the same as minimizing the sum of the number of connected components across all T_b. ■

In particular, if an item tree exists for the given bids, then in that tree, each bid bids on only one connected component, so the summation in Theorem 5 is equal to the number of bids. Because each term in the summation must always be greater than or equal to 1, this tree minimizes the summation. Thus, MAKETREE will return a tree for which the summation in Theorem 5 is equal to the number of bids as well. But this can only happen if each bid bids on only one connected component. So, MAKETREE will return an item tree.

Corollary 1 MAKETREE *will return a valid item tree if and only if one exists, in* $O(|B| \cdot |I|^2)$ *time. (And whether a tree is a valid item tree can be checked in* $O(|B| \cdot |I|)$ *time.)*

Implications for bid sets without an item tree

The above result presents an algorithm for constructing a tree T from a set of bids that minimizes the sum of the number of connected components across all T_b. Even when the tree returned is not a valid item tree, we can still use it to help us clear the auction, as follows. Suppose MAKETREE was "close" to being able to construct an item tree, in the sense that only a few bids were split into multiple components. Then, we could use brute force to determine which of these split bids to accept (we could search over all subsets of these split bids), and solve the rest of the problem using dynamic programming as in [Sandholm and Suri, 2003]. If the number of split bids is k, this algorithm takes $O(2^k \cdot |B| \cdot |I|)$ time (so it is efficient if k is small). We note, however, that MAKETREE(B, I) does not minimize the number of split bids (k), as would be desirable for the proposed search technique. Rather, it minimizes the total number of components (summed over bids). Thus, it may prefer splitting many bids into few components each, over splitting few bids into many components each. So there may exist trees that have fewer split bids than the tree returned by MAKETREE (and it would be interesting to try to come up with other algorithms that try to minimize the number of split bids).

Also, MAKETREE does not solve the general problem of constructing an item graph of small treewidth if one exists. The straightforward adaptation of the MAKETREE algorithm to finding an item graph of treewidth 2 (where we find the maximum spanning graph of treewidth 2 in the last step) does not always provide a valid item graph, even when a valid item graph of treewidth 2 exists. To see why, consider an auction instance for which the graph in Figure 3.11 is the unique item graph of treewidth 2 (for example, because for each edge, there is a bid on only its two endpoints). If there

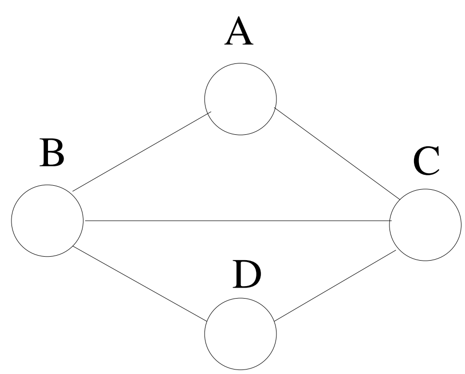


Figure 3.11: A counterexample.

are many bids on the bundle $\{A, B, D\}$, and few other bids, the adapted algorithm will draw the edge (A, D). As a result, it will fail to draw one of the other edges (because otherwise the graph would have treewidth 3), and thus the graph will not be a valid item graph.

For now, we leave open the question of how to construct a valid item graph with treewidth 2 (or 3, or 4, ...) if one exists. However, in the next subsection, we solve a related question.

3.2.4 Constructing the item graph with the fewest edges is hard

The more edges an item graph has, the less structure there is in the instance. A natural question is therefore to construct the valid item graph with the fewest edges. It should be pointed out that this is not necessarily the best graph to work on. For example, given our algorithm, a graph of treewidth 2 may be more desirable to work on than a graph with fewer edges of high treewidth. On the other hand, assuming that the items cannot be disjoint into two separate components (which is easy to check), a tree is always a graph with the minimum number of edges (and if a tree exists, then only trees have the minimum number of edges). So in this case, generating a graph of minimum treewidth is the same as generating a graph with the minimum number of edges.

We next show that constructing the graph with the fewest edges is hard. Interestingly, the question is hard already for instances with treewidth 2. (For instances of treewidth 1 (forests) it is easy: divide the items into as many separate components (with no bids across more than one component) as possible, and run our MAKETREE algorithm on each.) Thus, if a graph of treewidth 2 can be constructed in polynomial time (and P≠NP), the algorithm for doing so cannot be used to get the fewest edges—unlike the case of treewidth 1.

Theorem 6 *Determining whether an item tree with fewer than q edges exists is NP-complete, even when an item graph of treewidth 2 is guaranteed to exist and each bid is on at most 5 items, and whether or not the item tree we construct is required to be of treewidth 2.*

Proof: The problem is in NP because we can nondeterministically generate a graph with the items as vertices and at most k edges, and check whether it is valid item graph. To show that the problem is NP-complete, we reduce an arbitrary 3SAT instance to the following set of items and bids. For every variable $v \in V$, let there be two items i_{+v}, i_{-v}. Furthermore, let there be two more items, i_0 and i_1. Let the set of bids be as follows. For every $v \in V$, let there be bids on the following sets: $\{i_0, i_{+v}\}, \{i_0, i_{-v}\}, \{i_{+v}, i_{-v}\}, \{i_{+v}, i_{-v}, i_1\}$. Finally, for every clause $c \in C$, let there be a bid on $\{i_{+v} : +v \in c\} \cup \{i_{-v} : -v \in c\} \cup \{i_0, i_1\}$ (the set of all items corresponding to literals in the clause, plus the two extra items—we note that because we are reducing from 3SAT, these are at most 5 items). Let the target number of edges be $q = 4|V|$. We proceed to show that the two instances are equivalent.

First, suppose there exists a solution to the 3SAT instance. Then, let there be an edge between any two items which constitute a bid by themselves; additionally, let there be an edge between i_{+v} and i_1 whenever v is set to *true* in the SAT solution, and an edge between i_{-v} and i_1 whenever v is set to *false* in the SAT solution (for a total of $4|V|$ edges). We observe that all the bids of the form $\{i_{+v}, i_{-v}, i_1\}$ are now connected. Also, for any $c \in C$, because the 3SAT solution satisfied c, either i_1 is connected to some i_{+v} with $+v \in c$, or i_1 is connected to some i_{-v} with $-v \in c$. (And all the items besides i_1 in the bid corresponding to c are clearly connected.) So all the bids constitute connected subsets, and there exists a valid item graph with at most $4|V|$ edges. (Also, this is a series parallel graph, and such graphs have treewidth 2.)

Now, suppose there exists a valid item graph with at most $4|V|$ edges. Of course, there must be an edge between any two items which constitute a bid by themselves; and because of the bids on three items, for every $v \in V$, there must be an edge either between i_{+v} and i_1, or between i_{-v} and i_1. This already requires $4|V|$ edges, so there cannot be any more edges. Because each bid corresponding to a clause c must be connected, there must be either an edge between some i_{+v} with $+v \in c$ and i_1, or between some i_{-v} with $-v \in c$ and i_1. But then, it follows that if we set v to *true* if there is an edge between i_{+v} and i_1, and to *false* if there is an edge between i_{-v} and i_1, we have a solution to the SAT instance.

All that remains to show is that in these instances, a valid item graph of treewidth 2 always exists. Consider the graph that has an edge between any two items which constitute a bid by themselves;

an edge between i_{+v} and i_1 for any $v \in V$; and an edge between i_0 and i_1 (for a total of $4|V| + 1$ edges). This is a series parallel graph, and such graphs have treewidth 2. ■

3.2.5 Applications

The techniques in this section can be applied to any combinatorial auction winner determination instance whose bids happen to be consistent with some (structured) item graph. Specifically, there is no requirement that there is an item graph that makes sense for the items for sale *a priori* (before the bids arrive), based on the inherent properties of the items. Nevertheless, it is important to identify settings in which such *a priori* sensible item graphs do exist, for at least the following reasons. First, as described in the introduction, the auctioneer may wish to guarantee that the techniques described in this section can be applied by disallowing any bids that are not consistent with a prespecified item graph. This prespecified item graph should be chosen to be at least approximately consistent with bidders' likely valuations to minimize the loss of economic value due to this restriction. Second, especially in the case where the number of bidders is large, it is unlikely that the bids will be consistent with any structured item graph, unless the items are fundamentally related to each other in the manner prescribed by such an item graph.

Some settings for which *a priori* sensible item graphs exist have already been proposed. For example, Sandholm and Suri propose a web shopping scenario in which webpages describing the items for sale are structured as a tree, and bidders can, upon deciding that they wish to include the item on a webpage in their bid, continue to browse to neighboring pages [Sandholm and Suri, 2003].

In this subsection, we lay out two new settings in which the inherent relation between the items naturally suggests an item graph with the desired properties. In both of these settings, the "items" for sale do not correspond exactly to the resources under consideration. In the first setting we discuss, combinatorial renting, an item consists of the permission to use a given resource in a given time period. In the second setting, an item consists of a given set of conditions under which a given resource is allocated to the item's winner. We will make this more precise in the following subsubsections.

Combinatorial renting

In a *combinatorial renting auction*, we have a set of resources R and a number of time periods T over which to rent out these resources. In this context, an "item" for sale is a resource-time period pair $(r, t) \in R \times \{1, 2, \ldots, T\}$, and a *bid* consists of a subset of such pairs, representing at which times the bidder wants to rent which resources, together with a value offered for this subset. For example, a company renting out construction equipment may have a cement mixer, a truck, and a crane, each of which can be rented out over the course of three periods. A construction company working on a project may then bid (for example) $(\{(\text{mixer}, 1), (\text{truck}, 2), (\text{crane}, 2), (\text{truck}, 3)\}, \$15000)$, indicating that it wants to rent the mixer in the first period, the truck and the crane in the second period, and the truck again in the third period, for a total value of $15,000.

The constraint that we do not rent the same resource to multiple bidders at the same time corresponds to the constraint that we do not award the same item (r, t) to multiple bidders, and thus the winner determination problem reduces to the standard combinatorial auction winner determination problem. Moreover, the combinatorial renting auction winner determination problem is in general no easier than the standard combinatorial auction winner determination problem: for example, if $T = 1$, we effectively have a standard combinatorial auction in which items correspond directly to resources. The renting setting becomes more interesting when the multitude of the items is mostly due to the time dimension rather than the resource dimension. Thus, let us assume that the number of resources is small, *i.e.* bounded by a constant k. Additionally, suppose that the bids satisfy the following restriction: the set of time periods in which a bid demands items is connected. That is, for a bid on bundle B, for any $t_1, t_2, t_3 \in \{1, 2, \ldots, T\}, t_1 < t_3 < t_2, r_1, r_2 \in R$ with $(r_1, t_1), (r_2, t_2) \in B$, there must exist some $r_3 \in R$ such that $(r_3, t_3) \in B$. This is a sensible restriction when each bid corresponds to a project for which resources must be rented (*e.g.* the construction example above) and the project is scheduled for a particular time interval.

Under this restriction, the following is a valid item graph:

- For every $t \in \{1, 2, \ldots, T\}$, draw a subgraph G_t whose vertex set is $\{(r, t) : r \in R\}$, and make it fully connected (edges between every pair of vertices).
- Connect the subgraphs by, for every $t \in \{1, 2, \ldots, T-1\}$, drawing an edge from every vertex in G_t to every vertex in G_{t+1}.

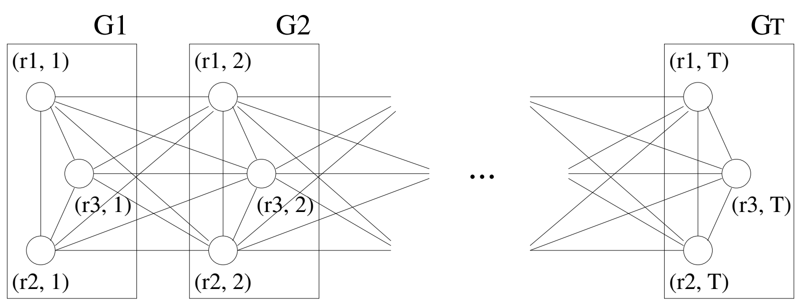


Figure 3.12: Item graph for renting three resources r_1, r_2, r_3.

Moreover, this graph has bounded treewidth as shown by the following tree decomposition (in which V_t is the vertex set of G_t):

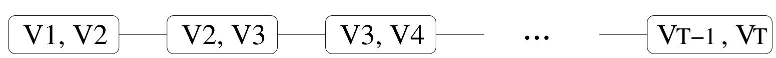


Figure 3.13: Tree decomposition in renting setting.

(In fact, because the tree decomposition is a path, this shows that the original graph has bounded pathwidth.) The width of this decomposition is $2|R| - 1$. This is an *a priori* bound, and it is possible that given the actual bids, an item graph with even smaller treewidth exists.

We now turn to a different setting in which our techniques can be applied.

Conditional awarding of the items

In a *combinatorial conditional awarding auction*, we again have a set of resources R; in addition, we have a set of possible future states of the world S. (For now, we will only concern ourselves with a single point in the future, say, the beginning of the next fiscal year.) In this context, an "item" for sale is a resource-state pair $(r, s) \in R \times S$, and a *bid* consists of a subset of such pairs, representing under which conditions the bidder wants to be awarded which resources. For example, there may be three states of the world, one in which the price of oil is below \$40 per barrel, one in which the price is between \$40 and \$60, and one in which the price is above \$60. A car dealer may have a sport utility vehicle (SUV) and a small car for sale. Then, a bidder may bid $(\{(\text{SUV}, p_o < \$40), (\text{small car}, p_o < \$40), (\text{SUV}, \$40 \leq p_o \leq \$60), (\text{small car}, p_o > \$60)\}, \$30000)$ indicating that she wants to receive both cars when the price of oil is low, only the SUV when the price of oil is in the middle range, and only the small car when the price of oil is in the high range; and that this total package is worth \$30,000 to her. (In reality, the bidder may wish to make different payments depending on which state of the world materializes; this can be incorporated into the model by using the bidder's *expected* payment.) We note that the "items" for sale are effectively securities that bidders can use to hedge against uncertain future events.

The constraint that we do not award the same resource to multiple bidders in the same state of the world corresponds to the constraint that we do not award the same item (r, s) to multiple bidders, and thus the winner determination problem reduces to the standard combinatorial auction winner determination problem. Also, in the case where $|S| = 1$, we effectively have a standard combinatorial auction in which items correspond directly to resources. Thus, as in the combinatorial renting setting, the most interesting case to look at is where R is small but S is potentially large. One interesting setting to look at is the one in which there is a linear order on the state space S. For example, the "price of oil" state space described above has such an order (higher vs. lower prices). This case is technically similar to the renting problem studied in the previous subsubsection: if the number of items is bounded by a constant k, and the states in which a bid demands items are connected (or there are only small gaps), then we can solve the problem in polynomial time. The connectedness property is likely to hold when the bidder is interested in a set of similar states (for example, the bidder wishes to hedge against high and very high oil prices).

The state space may also be the cross product of multiple linear orders. For example, a state could represent a combination of the price of oil and the exchange rate between the US dollar and the Euro. In this case, we may expect that the set of states in which a bid demands items is connected in a *grid* graph such as the following:

We can turn such a grid into a valid item graph by replacing each state with a fully connected graph (a clique) whose vertices are all items that involve that state (one for every resource), and drawing edges between any pair of items in adjacent cliques. The treewidth of this graph is at most $|R|$ times the treewidth of the grid (the graph over states), because given any tree decomposition of the grid, we can replace each state by all the items involving that state (one for every resource) to obtain a valid tree decomposition of the item graph. Unfortunately, grids do not have bounded treewidth; rather, an $m \times n$ grid has treewidth $\Theta(\min\{m, n\})$. Of course, this is still much better than the trivial bound of mn that would correspond to exhaustive search.

Combining both settings: conditional renting

As a third application, we may wish to combine the applications of the previous two subsubsections and let the bidder *rent* the items *conditionally* on the state of the world at the time of the renting. For example, the construction company described in the first subsubsection may wish to rent different

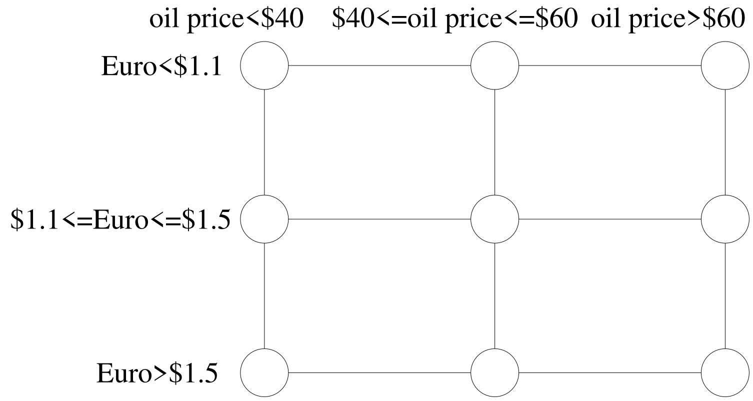


Figure 3.14: State space graph based on oil price and US$/Euro exchange rate.

resources depending on the price of oil (or the weather, or anything else) at the times the resources are to be rented. Hence, the *context* in which a resource is awarded now consists of a (time,state) pair. If there is a linear order on the state space, we may expect that the set of contexts in which a bid demands items is connected in a grid graph such as the following:

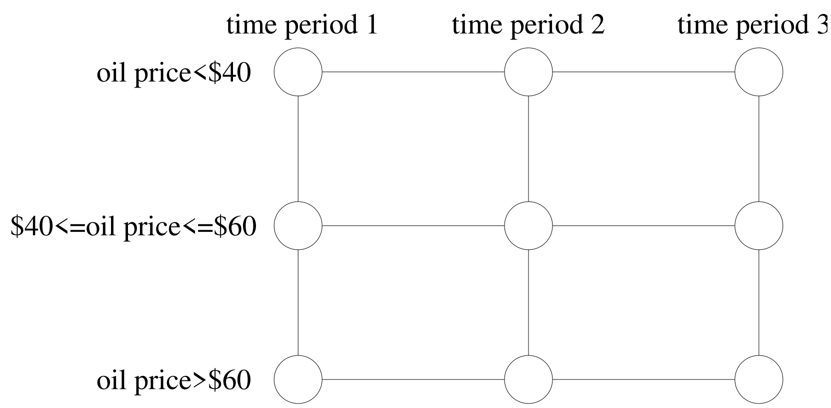


Figure 3.15: Context space graph based on time and state space (oil price).

Technically, this leads to same problem as the two-dimensional state space described in the previous subsubsection.

3.2.6 Bids on multiple connected sets

In this subsection, we investigate what happens if we reduce the requirements on item graphs somewhat. Specifically, let the requirement be that for each bid, the items form *at most* k connected components in the graph. (The case where $k = 1$ is the one we have studied up to this point.) So, to see if a bid is valid given the graph, consider how many connected components the items in the bid constitute in the graph; if (and only if) there are at most k components, the bid is valid. Figure 3.16 shows an example item tree. One bid bids on all the items encapsulated by rectangles (2 connected components); the other, on all the items encapsulated by ellipses (3 connected components). Thus, if $k = 2$, then the first bid is valid, but the second one is not. (Equivalently, the graph is not valid for the second bid.)

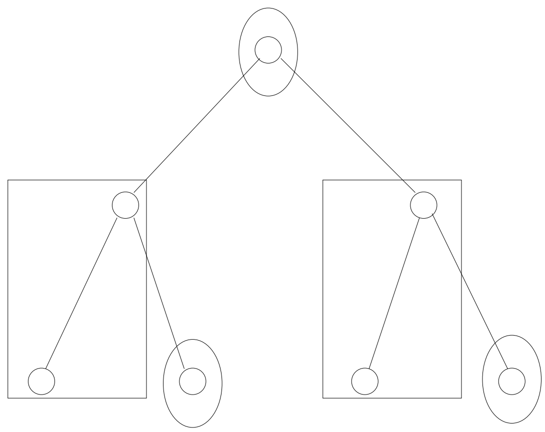


Figure 3.16: An item tree.

As we will see, both clearing when a simple graph is known, and detecting whether a simple graph exists, become hard for $k > 1$.

Clearing is hard even with 2 connected sets on a line graph

Even if the item graph is a line, it is hard to clear auctions in which bidders may bundle two intervals together. To show this, we prove the following slightly stronger theorem.

Theorem 7 *If an item graph is created that consists of two disconnected line graphs, and bidders are permitted to bundle one connected component from each line, then determining if the auction can generate revenue r is NP-complete.*

Proof: The problem is in NP because the general clearing problem is in NP. To show that it is NP-complete, an arbitrary instance of VERTEXCOVER will be reduced to a corresponding auction problem. In VERTEXCOVER, the goal is to determine whether there exists a set of vertices C of size at most k in a graph $G = (V, E)$ such that for each edge $(x, y) \in E$, either $x \in C$ or $y \in C$.

To perform the reduction, given $G = (V, E)$, create an item u_i for each edge e_i. Place all of the u_i items into the upper line. In addition, for each vertex $v_i \in V$, create the items $l_{v_i}^{e_j}$ for each edge e_j that v_i is part of. For each v_i, align all of its corresponding $l_{v_i}^{e_j}$ items into a contiguous interval on the lower line. Now, create the following bids: 1. A bid of price 2 for each "edge item pair" $(l_{v_i}^{e_j}, u_j)$. 2. A bid of price 1 for each "vertex interval bundle," $\{l_{v_i}^{e_j} |$for all e_j for which v_i is part of $e_j\}$.

We now show that there exists a vertex cover of size at most k exactly when the optimal revenue of the corresponding auction is at least $2 \cdot |E| + |V| - k$.

Suppose there is a vertex cover of size k for a graph $G = (V, E)$. Then it is possible to sell all $|E|$ of the edge items breaking up only k of the vertex interval bundles. The resulting revenue if only those k intervals are rendered unsellable by matching one or more of their members with edge items is $r = 2 \cdot |E| + |V| - k$ as required; the profit is 2 for each of the edge pairs of the form $(l_{v_i}^{e_j}, u_j)$ that are sold, plus 1 for each of the vertex interval bundles that have not had any of their items matched to edges.

Conversely, suppose there is a way to achieve revenue $r = 2 \cdot |E| + |V| - k$. Suppose all of the edge item pairs were sold in this case. Then, because $|V| - k$ vertex intervals were sold, only k were spoiled by selling some of their items with edge pairs. These k vertices can be used as a vertex cover. On the other hand, suppose not all edge pairs were sold. Then the revenue could be increased by selling the rest of the edge pairs even if it means spoiling additional vertex bundles because edge pairs carry a price of 2 while vertex interval bundles only have price 1. This implies that it is possible, by selling all of the edge item pairs, to acheive revenue $r' = 2 \cdot |E| + |V| - k' > r$ so that $k' < k$, where k' represents the size of a possible vertex cover. ■

Corollary 2 *The problem of optimally clearing bids that bundle together at most two connected components of a line item graph is NP-hard because joining the two lines of item graphs of the form discussed in Theorem 7 constitutes a trivially correct reduction.*

Constructing a line graph is hard even with 5 connected sets

We now move on to the task of constructing a simple item graph that is valid when bids are allowed to consist of multiple components. The graph construction question is perhaps less interesting here because, as we just showed, clearing remains hard even if we are given an item line graph. Nevertheless, the graph may still be helpful in reducing the clearing time: maybe the clearing time bound can be reduced to a smaller exponential function, or maybe it can reduce the clearing time in practice. In this subsection, we show that unfortunately, detecting whether a valid line graph exists with multiple (5) components is also NP-complete.

Lemma 6 *Suppose a bid is allowed to contain up to k connected components of items. Then, if there are $m \geq 2k + 5$ items, there exists a set of $O(m^k)$ bids such that there is exactly one line graph (one ordering of the items, up to symmetry) consistent with these bids.*

Proof: Label the items 1 through m. For any subset of $k + 1$ items of which at least two items are successors (i and $i + 1$), let there be a bid on that set of items. We observe that there are at most $(m - 1)\binom{m}{k-1}$ such bids (choosing the successive pair of items first, and then the remaining $k - 1$—of course we are double-counting some combinations this way, but we only want an upper bound), which is $O(m^k)$. Ordering the items $1, 2, \ldots, m$ (or equivalently $m, m - 1, \ldots, 1$), we get a valid item graph (because any two successive items are adjacent in this graph, there are at most k components in every bid). What remains to show is that if the items are ordered differently, there is at least one bid with $k + 1$ components. If the items are ordered differently, there is at least one pair of successive (according to the original labeling) items $i, i + 1$ which are not adjacent in the graph. Consider the set of these two items, plus every item that has an odd index in the ordering of this graph (besides the ones that coincide with, or are adjacent to, the first two). This set has at least $2 + (k + 3) - 4 = k + 1$ items, two of which are adjacent in the original labeling, and each of which is a separate component. It follows that there exists a subset of this set which constituted one of the bids, and now has $k+1$ components. ■

Lemma 7 *Suppose each bid is allowed to contain at most k connected components, and we have a set of bids that forces a unique ordering of the items (up to symmetry). Then suppose we replace one item r with two new items n_1 and n_2, and let every bid bidding on the original item bid on both the new items. Then the only valid orderings of the new set of items are the valid orderings for the original set, where r is replaced by n_1, n_2 (where these two can be placed in any order). This extends to replacing multiple items by pairs.*

Proof: First we show that these orderings are indeed valid. Clearly, no bid that did not include r will now have more components. Also, no bid that did include r will now have more components, because the component including r is still intact as a single component (since the bid bids on both n_1 and n_2). So the new orderings are valid. Now we will show that these are the only valid orderings. We observe that if we remove one of n_1, n_2 from a given valid ordering as well as from the bids, then we must still have a valid ordering. But because now r has been replaced by a single item, we know that the valid ordering for this is unique (up to symmetry). It follows that n_1 and n_2 had taken r's place in the unique valid ordering. The argument extends straightforwardly to replacing multiple items by item pairs. ■

Theorem 8 *Given the bids, detecting whether an ordering of the items (a line graph) exists such that each bidder bids on at most 5 connected components is NP-complete.*

Proof: The problem is in NP because we can nondeterministically generate an ordering of the items, and check whether any bid is bidding on more than 5 components. To show that the problem is NP-hard, we reduce an arbitrary 3SAT instance to the following sets of items and bids. For every variable $v \in V$, let there be four items, $i^*_v, i_v, i_{+v}, i_{-v}$. Let the set of bids be as follows. First, using Lemma 6 and Lemma 7, let there be $O(m^5)$ bids such that the only remaining valid orderings are $i^*_{v_1}, i_{v_1}, \{i_{+v_1}, i_{-v_1}\}, i^*_{v_2}, i_{v_2}, \{i_{+v_2}, i_{-v_2}\}, \ldots, i^*_{v_n}, i_{v_n}, \{i_{+v_n}, i_{-v_n}\}$. (Here, two items are in set notation if their relative order is not yet determined.) Finally, for every clause $c \in C$, let there be a bid bidding on any i_v with v occurring in c (whether it is $+v$ or $-v$), and on any i_{+v} with $+v$ occurring in c, and on any i_{-v} with $-v$ occurring in c. (So, 6 items in total.) We show the instances are equivalent.

First suppose there exists a solution to the 3SAT instance. Then, whenever a variable v is set to *true*, let i_{+v} be ordered to the left of i_{-v}; otherwise, let i_{+v} be ordered to the right of i_{-v}. Then, for every clause, for some literal $+v$ (or $-v$) occurring in that clause, i_{+v} (or i_{-v}) is adjacent to i_v, and it follows that the bid corresponding to the clause has at most 5 connected components. So, there is a valid ordering.

Now suppose there exists a valid ordering. Because of the i^*_v items, the only items in a bid corresponding to a clause that can possibly be adjacent are an i_{+v} and the corresponding i_v, or an i_{-v} and the corresponding i_v. This must happen at least once for every bid corresponding to a clause (or the bid would have 6 components. But then, if we set a variable v to *true* if i_{+v} and i_v

are adjacent, and to *false* otherwise, every clause must have at least one $+v$ in it where v is set to *true*, or at least one $-v$ in it where v is set to *false*. It follows that there is a solution to the 3SAT instance. ■

This concludes the part of this dissertation studying the complexity of the winner determination problem in combinatorial auctions; we will return to combinatorial auctions and exchanges (specifically, to the use of the VCG mechanism in such settings) in the chapter after the next chapter, Section 5.1. In the next section, we study the complexity of the outcome optimization problem in the setting of expressive preference aggregation for donations to charities.

3.3 Expressive preference aggregation for donations to charities

In this section, we study the outcome optimization problem for expressive preference aggregation for donations to charities, as defined in Section 2.3. We will refer to this problem as the *clearing* problem. The formal definition follows.

Definition 13 (DONATION-CLEARING) *We are given a set of n bids (each given by a utility function for each charity, and a payment willingness function) over charities $c_1, c_2, \ldots, c_m$ as described in Section 2.3. Additionally, we are given an objective function (*e.g.* surplus, or total amount donated). We are asked to find an objective-maximizing valid outcome.*

One aspect of the problem is not captured by this definition: if we want a decentralized solution, in which bidders donate their money to the charity directly (rather than to a center who then redistributes it), then we also need to specify which bidder donates how much to which charity. Assuming that we are given the centralized solution, any greedy algorithm that increases the cash flow from any bidder who has not yet paid enough, to any charity that has not yet received enough, until either the bidder has paid enough or the charity has received enough, will provide such a specification. Recall, however, that we may wish to allow for bidders to state that they do not wish to donate to certain charities. In general, checking whether a given centralized solution can be accomplished through decentralized payments when there are such constraints can be modeled as a MAX-FLOW problem. In the MAX-FLOW instance, there is an edge from the source node s to each bidder b_j, with a capacity of π_{b_j} (as specified in the centralized solution); an edge from each bidder b_j to each charity c_i *that the bidder is willing to donate money to*, with a capacity of ∞; and an edge from each charity c_i to the target node t with capacity π_{c_i} (as specified in the centralized solution).

In the remainder of this section, we will no longer consider the problem of decentralizing solutions; rather, we focus on the DONATION-CLEARING problem. How difficult the DONATION-CLEARING problem is depends on the types of bids used and the language in which they are expressed. This is the topic of the next subsection.

3.3.1 Hardness of clearing the market

In this subsection, we will show that the clearing problem is completely inapproximable, even when every bidder's utility function is linear (with slope 0 or 1 in each charity's payments), each bidder cares either about at most two charities or about all charities equally, and each bidder's payment willingness function is a step function. We will reduce from MAX2SAT (given a formula in conjunctive normal form (where each clause has two literals) and a target number of satisfied clauses T, does there exist an assignment of truth values to the variables that makes at least T clauses true?), which is NP-complete [Garey *et al.*, 1976].

Theorem 9 *There exists a reduction from MAX2SAT instances to DONATION-CLEARING instances such that 1. If the MAX2SAT instance has no solution, then the only valid outcome is the zero outcome (no bidder pays anything and no charity receives anything); 2. Otherwise, there exists a solution with positive surplus. Additionally, the DONATION-CLEARING instances that we reduce to have the following properties: 1. Every u_j^i is a line; that is, the utility that each bidder derives from any charity is linear; 2. All the u_j^i have slope either 0 or 1; 3. Every bidder either has at most 2 charities that affect her utility (with slope 1), or all charities affect her utility (with slope 1); 4. Every bid is a threshold bid; that is, every bidder's payment willingness function w_j is a step function.*

Proof: The problem is in NP because we can nondeterministically choose the payments to be made and received, and check the validity and objective value of this outcome.

In the following, we will represent bids as follows: $(\{(c_k, a_k)\}, s, t)$ indicates that $u_j^k(\pi_{c_k}) = a_k \pi_{c_k}$ (this function is 0 for c_k not mentioned in the bid), and $w_j(u_j) = t$ for $u_j \geq s$, $w_j(u_j) = 0$ otherwise.

To show NP-hardness, we reduce an arbitrary MAX2SAT instance, given by a set of clauses $K = \{k\} = \{(l_k^1, l_k^2)\}$ over a variable set V together with a target number of satisfied clauses T, to the following DONATION-CLEARING instance. Let the set of charities be as follows. For every literal $l \in L$, there is a charity c_l. Then, let the set of bids be as follows. For every variable v, there is a bid $b_v = (\{(c_{+v}, 1), (c_{-v}, 1)\}, 2, 1 - \frac{1}{4|V|})$. For every literal l, there is a bid $b_l = (\{(c_l, 1)\}, 2, 1)$. For every clause $k = \{l_k^1, l_k^2\} \in K$, there is a bid $b_k = (\{(c_{l_k^1}, 1), (c_{l_k^2}, 1)\}, 2, \frac{1}{8|V||K|})$. Finally, there is a single bid that values all charities equally: $b_0 = (\{(c_1, 1), (c_2, 1), \ldots, (c_m, 1)\}, 2|V| + \frac{T}{8|V||K|}, \frac{1}{4} + \frac{1}{16|V||K|})$. We show the two instances are equivalent.

First, suppose there exists a solution to the MAX2SAT instance. If in this solution, l is *true*, then let $\pi_{c_l} = 2 + \frac{T}{8|V|^2|K|}$; otherwise $\pi_{c_l} = 0$. Also, the only bids that are *not* accepted (meaning the threshold is not met) are the b_l where l is *false*, and the b_k such that both of l_k^1, l_k^2 are *false*. First we show that no bidder whose bid is accepted pays more than she is willing to. For each b_v, either c_{+v} or c_{-v} receives at least 2, so this bidder's threshold has been met. For each b_l, either l is *false* and the bid is not accepted, or l is *true*, c_l receives at least 2, and the threshold has been met. For each b_k, either both of l_k^1, l_k^2 are *false* and the bid is not accepted, or at least one of them (say l_k^i) is *true* (that is, k is satisfied) and $c_{l_k^i}$ receives at least 2, and the threshold has been met. Finally, because the total amount received by the charities is $2|V| + \frac{T}{8|V||K|}$, b_0's threshold has also been met. The total amount that can be extracted from the accepted bids is at least $|V|(1 - \frac{1}{4|V|}) + |V| + T\frac{1}{8|V||K|} + \frac{1}{4} + \frac{1}{16|V||K|}) = 2|V| + \frac{T}{8|V||K|} + \frac{1}{16|V||K|} > 2|V| + \frac{T}{8|V||K|}$, so there is positive surplus. So there exists a solution with positive surplus to the DONATION-CLEARING instance.

Now suppose there exists a nonzero outcome in the DONATION-CLEARING instance. First we show that it is not possible (for any $v \in V$) that both b_{+v} and b_{-v} are accepted. For, this would require that $\pi_{c_{+v}} + \pi_{c_{-v}} \geq 4$. The bids b_v, b_{+v}, b_{-v} cannot contribute more than 3, so we need another 1 at least. It is easily seen that for any other v', accepting any subset of $\{b_{v'}, b_{+v'}, b_{-v'}\}$ would require that at least as much is given to $c_{+v'}$ and $c_{-v'}$ as can be extracted from these bids, so this cannot help. Finally, all the other bids combined can contribute at most $|K|\frac{1}{8|V||K|} + \frac{1}{4} + \frac{1}{16|V||K|} < 1$. It follows that we can interpret the outcome in the DONATION-CLEARING instance as a partial assignment of truth values to variables: v is set to *true* if b_{+v} is accepted, and to *false* if b_{-v} is accepted. All that is left to show is that this partial assignment satisfies at least T clauses.

First we show that if a clause bid b_k is accepted, then either $b_{l_k^1}$ or $b_{l_k^2}$ is accepted (and thus either l_k^1 or l_k^2 is set to *true*, hence k is satisfied). If b_k is accepted, at least one of $c_{l_k^1}$ and $c_{l_k^2}$ must be receiving at least 1; without loss of generality, say it is $c_{l_k^1}$, and say l_k^1 corresponds to variable v_k^1 (that is, it is $+v_k^1$ or $-v_k^1$). If $c_{l_k^1}$ does not receive at least 2, $b_{l_k^1}$ is not accepted, and it is easy to check that the bids $b_{v_k^1}, b_{+v_k^1}, b_{-v_k^1}$ contribute (at least) 1 less than is paid to $c_{+v_k^1}$ and $c_{+v_k^1}$. But this is the same situation that we analyzed before, and we know it is impossible. All that remains to show is that at least T clause bids are accepted.

We now show that b_0 is accepted. Suppose it is not; then one of the b_v must be accepted. (The solution is nonzero by assumption; if only some b_k are accepted, the total payment from these bids is at most $|K|\frac{1}{8|V||K|} < 1$, which is not enough for any bid to be accepted; and if one of the b_l is accepted, then the threshold for the corresponding b_v is also reached.) For this v, $b_{v_k^1}, b_{+v_k^1}, b_{-v_k^1}$ contribute (at least) $\frac{1}{4|V|}$ less than the total payments to c_{+v} and c_{-v}. Again, the other b_v and b_l cannot (by themselves) help to close this gap; and the b_k can contribute at most $|K|\frac{1}{8|V||K|} < \frac{1}{4|V|}$. It follows that b_0 is accepted.

Now, in order for b_0 to be accepted, a total of $2|V| + \frac{T}{8|V||K|}$ must be donated. Because is not possible (for any $v \in V$) that both b_{+v} and b_{-v} are accepted, it follows that the total payment by the b_v and the b_l can be at most $2|V| - \frac{1}{4}$. Adding b_0's payment of $\frac{1}{4} + \frac{1}{16|V||K|}$ to this, we still need $\frac{T-\frac{1}{2}}{8|V||K|}$ from the b_k. But each one of them contributes at most $\frac{1}{8|V||K|}$, so at least T of them must be accepted. ■

Corollary 3 *Unless ZPP=NP, there is no polynomial-time algorithm for approximating DONATION-CLEARING (with either the surplus or the total amount donated as the objective) within any ratio $f(n)$, where f is a nonzero function of the size of the instance. This holds even if the DONATION-CLEARING structures satisfy all the properties given in Theorem 9.*

Proof: Suppose we had such a polynomial time algorithm, and applied it to the DONATION-CLEARING instances that were reduced from MAX2SAT instances in Theorem 9. It would return a nonzero solution when the MAX2SAT instance has a solution, and a zero solution otherwise. So we can decide whether arbitrary MAX2SAT instances are satisfiable this way, and it would follow that ZPP=NP. ■

This should not be interpreted to mean that our approach is infeasible. First, as we will show, there are very expressive families of bids for which the problem is solvable in polynomial time.

Second, NP-completeness is often overcome in practice (especially when the stakes are high). For instance, even though the problem of clearing combinatorial auctions is NP-complete [Rothkopf *et al.*, 1998] (even to approximate [Sandholm, 2002a]), they are typically solved to optimality in practice [Sandholm *et al.*, 2006; Sandholm, 2006].

3.3.2 Mixed integer programming formulation

In this subsection, we give a mixed integer programming (MIP) formulation for the general problem. We also discuss in which special cases this formulation reduces to a linear programming (LP) formulation. In such cases, the problem is solvable in polynomial time, because linear programs can be solved in polynomial time [Khachiyan, 1979].

The variables of the MIP defining the final outcome are the payments made to the charities, denoted by π_{c_i}, and the payments extracted from the bidders, π_{b_j}. In the case where we try to avoid direct payments and let the bidders pay the charities directly, we add variables π_{c_i,b_j} indicating how much b_j pays to c_i, with the constraints that for each c_i, $\pi_{c_i} \leq \sum\limits_{b_j} \pi_{c_i,b_j}$; and for each b_j, $\pi_{b_j} \geq \sum\limits_{c_i} \pi_{c_i,b_j}$. Additionally, there is a constraint $\pi_{c_i,b_j} = 0$ whenever bidder b_j is unwilling to pay charity c_i. The rest of the MIP can be phrased in terms of the π_{c_i} and π_{b_j}.

The objectives we have discussed earlier are both linear: surplus is given by $\sum\limits_{j=1}^{n} \pi_{b_j} - \sum\limits_{i=1}^{m} \pi_{c_i}$, and total amount donated is given by $\sum\limits_{i=1}^{m} \pi_{c_i}$ (coefficients can be added to represent different weights on the different charities in the objective).

The constraint that the outcome should be valid (no deficit) is given simply by: $\sum\limits_{j=1}^{n} \pi_{b_j} \geq \sum\limits_{i=1}^{m} \pi_{c_i}$.

For every bidder, for every charity, we define an additional utility variable u_j^i indicating the utility that this bidder derives from the payment to this charity. The bidder's total utility is given by another variable u_j, with the constraint that $u_j = \sum\limits_{i=1}^{m} u_j^i$.

Each u_j^i is given as a function of π_{c_i} by the (piecewise linear) function provided by the bidder. In order to represent this function in the MIP formulation, we will merely place upper bounding constraints on u_j^i, so that it cannot exceed the given functions. The MIP solver can then push the u_j^i variables all the way up to the constraint, in order to extract as much payment from this bidder as possible. In the case where the u_j^i are concave, this is easy: if (s_l, t_l) and (s_{l+1}, t_{l+1}) are endpoints of a finite linear segment in the function, we add the constraint that $u_j^i \leq t_l + \frac{\pi_{c_i} - s_l}{s_{l+1} - s_l}(t_{l+1} - t_l)$. If the final (infinite) segment starts at (s_k, t_k) and has slope d, we add the constraint that $u_j^i \leq t_k + d(\pi_{c_i} - s_k)$. Using the fact that the function is concave, for each value of π_{c_i}, the tightest upper bound on u_j^i is the one corresponding to the segment above that value of π_{c_i}, and therefore these constraints are sufficient to force the correct value of u_j^i.

When the function is not concave, we require (for the first time) some binary variables. First, we define another point on the function: $(s_{k+1}, t_{k+1}) = (s_k + M, t_k + dM)$, where d is the slope of the infinite segment and M is any upper bound on the π_{c_j}. This has the effect that we will never be on the infinite segment again. Now, let $x_l^{i,j}$ be an indicator variable that should be 1 if π_{c_i} is below the lth segment of the function, and 0 otherwise. To effect this, first add a constraint $\sum_{l=0}^{k} x_l^{i,j} = 1$. Now, we aim to represent π_{c_i} as a weighted average of its two neighboring $s_l^{i,j}$. For $0 \leq l \leq k+1$, let $\lambda_l^{i,j}$ be the weight on $s_l^{i,j}$. We add the constraint $\sum_{l=0}^{k+1} \lambda_l^{i,j} = 1$. Also, for $0 \leq l \leq k+1$, we add the constraint $\lambda_l^{i,j} \leq x_{l-1} + x_l$ (where x_{-1} and x_{k+1} are defined to be zero), so that indeed only the two neighboring $s_l^{i,j}$ have nonzero weight. Now we add the constraint $\pi_{c_i} = \sum_{l=0}^{k+1} s_l^{i,j} \lambda_l^{i,j}$, and now the $\lambda_l^{i,j}$ must be set correctly. Then, we can set $u_j^i = \sum_{l=0}^{k+1} t_l^{i,j} \lambda_l^{i,j}$. (This is a standard MIP technique [Nemhauser and Wolsey, 1999].)

Finally, each π_{b_j} is bounded by a function of u_j by the (piecewise linear) function provided by the bidder (w_j). Representing this function is entirely analogous to how we represented u_j^i as a function of π_{c_i}. (Again we will need binary variables only if the function is not concave.)

Because we only use binary variables when either a utility function u_j^i or a payment willingness function w_j is not concave, it follows that if all of these are concave, our MIP formulation is simply a linear program—which can be solved in polynomial time. Thus:

Theorem 10 *If all functions u_j^i and w_j are concave (and piecewise linear), the DONATION-CLEARING problem can be solved in polynomial time using linear programming.*

Even if some of these functions are not concave, we can simply replace each such function by the smallest upper bounding concave function, and use the linear programming formulation to obtain an upper bound on the objective—which may be useful in a search formulation of the general problem.

3.3.3 Why one cannot do much better than linear programming

One may wonder if, for the special cases of the DONATION-CLEARING problem that can be solved in polynomial time with linear programming, there exist special-purpose algorithms that are much faster than linear programming algorithms. In this subsection, we show that this is not the case. We give a reduction *from* (the decision variant of) the general linear programming problem to (the decision variant of) a special case of the DONATION-CLEARING problem (which can be solved in polynomial time using linear programming). (The decision variant of an optimization problem asks the binary question: "Can the objective value exceed o?") Thus, any special-purpose algorithm for solving the decision variant of this special case of the DONATION-CLEARING problem could be used to solve a decision question about an arbitrary linear program just as fast. (And thus, if we are willing to call the algorithm a logarithmic number of times, we can solve the optimization version of the linear program.)

We first observe that for linear programming, a decision question about the objective can simply be phrased as another constraint in the LP (forcing the objective to exceed the given value); then, the original decision question coincides with asking whether the resulting linear program has a feasible solution.

Theorem 11 *The question of whether an LP (given by a set of linear constraints*[9]*) has a feasible solution can be modeled as a DONATION-CLEARING instance with payment maximization as the objective, with $2v$ charities and $v + c$ bids (where v is the number of variables in the LP, and c is the number of constraints). In this model, each bid b_j has only linear u_j^i functions, and is a partially acceptable threshold bid ($w_j(u) = t_j$ for $u \geq s_j$, otherwise $w_j(u) = \frac{ut_j}{s_j}$). The v bids corresponding to the variables mention only two charities each; the c bids corresponding to the constraints mention only two times the number of variables in the corresponding constraint.*

Proof: For every variable x_i in the LP, let there be two charities, c_{+x_i} and c_{-x_i}. Let H be some number such that if there is a feasible solution to the LP, there is one in which every variable has absolute value at most H.

In the following, we will represent bids as follows: $(\{(c_k, a_k)\}, s, t)$ indicates that $u_j^k(\pi_{c_k}) = a_k\pi_{c_k}$ (this function is 0 for c_k not mentioned in the bid), and $w_j(u_j) = t$ for $u_j \geq s$, $w_j(u_j) = \frac{u_j t}{s}$ otherwise.

For every variable x_i in the LP, let there be a bid $b_{x_i} = (\{(c_{+x_i}, 1), (c_{-x_i}, 1)\}, 2H, 2H - \frac{c}{v})$. For every constraint $\sum\limits_i r_i^j x_i \leq s_j$ in the linear program, let there be a bid $b_j = (\{(c_{-x_i}, r_i^j)\}_{i:r_i^j>0} \cup \{(c_{+x_i}, -r_i^j)\}_{i:r_i^j<0}, (\sum\limits_i |r_i^j|)H - s_j, 1)$. Let the target total amount donated be $2vH$.

Suppose there is a feasible solution $(x_1^*, x_2^*, \ldots, x_v^*)$ to the LP. Without loss of generality, we can suppose that $|x_i^*| \leq H$ for all i. Then, in the DONATION-CLEARING instance, for every i, let $\pi_{c_{+x_i}} = H + x_i^*$, and let $\pi_{c_{-x_i}} = H - x_i^*$ (for a total payment of $2H$ to these two charities). This allows us to extract the maximum payment from the bids b_{x_i}—a total payment of $2vH - c$. Additionally, the utility of bidder b_j is now $\sum\limits_{i:r_i^j>0} r_i^j(H - x_i^*) + \sum\limits_{i:r_i^j<0} -r_i^j(H + x_i^*) = (\sum\limits_i |r_i^j|)H - \sum\limits_i r_i^j x_i^* \geq (\sum\limits_i |r_i^j|)H - s_j$ (where the last inequality stems from the fact that constraint j must be satisfied in the LP solution), so it follows we can extract the maximum payment from all the bidders b_j, for a total payment of c. It follows that we can extract the required $2vH$ payment from the bidders, and there exists a solution to the DONATION-CLEARING instance with a total amount donated of at least $2vH$.

Now suppose there is a solution to the DONATION-CLEARING instance with a total amount donated of at least vH. Then the maximum payment must be extracted from each bidder. From the fact that the maximum payment must be extracted from each bidder b_{x_i}, it follows that for each i, $\pi_{c_{+x_i}} + \pi_{c_{-x_i}} \geq 2H$. Because the maximum extractable total payment is $2vH$, it follows that for each i, $\pi_{c_{+x_i}} + \pi_{c_{-x_i}} = 2H$. Let $x_i^* = \pi_{c_{+x_i}} - H = H - \pi_{c_{-x_i}}$. Then, from the fact that the maximum payment must be extracted from each bidder b_j, it follows that $(\sum\limits_i |r_i^j|)H - s_j \leq \sum\limits_{i:r_i^j>0} r_i^j \pi_{c_{-x_i}} + \sum\limits_{i:r_i^j<0} -r_i^j \pi_{c_{+x_i}} = \sum\limits_{i:r_i^j>0} r_i^j(H - x_i^*) + \sum\limits_{i:r_i^j<0} -r_i^j(H + x_i^*) = (\sum\limits_i |r_i^j|)H - \sum\limits_i r_i^j x_i^*$.

[9] These constraints must include bounds on the variables (including nonnegativity bounds), if any.

Equivalently, $\sum_i r_i^j x_i^* \leq s_j$. It follows that the x_i^* constitute a feasible solution to the LP. ■

3.3.4 Quasilinear bids

Another class of bids of interest is the class of *quasilinear bids*. In a quasilinear bid, the bidder's payment willingness function is linear in utility: that is, $w_j = u_j$. (Because the units of utility are arbitrary, we may as well let them correspond exactly to units of money—so we do not need a constant multiplier.) In most cases, quasilinearity is an unreasonable assumption: for example, usually bidders have a limited budget for donations, so that the payment willingness will stop increasing in utility after some point (or at least increase slower in the case of a "softer" budget constraint). Nevertheless, quasilinearity may be a reasonable assumption in the case where the bidders are large organizations with large budgets, and the charities are a few small projects requiring relatively little money. In this setting, once a certain small amount has been donated to a charity, a bidder will derive no more utility from more money being donated from that charity. Thus, the bidders will never reach a high enough utility for their budget constraint (even when it is soft) to take effect, and thus a linear approximation of their payment willingness function is reasonable. Another reason for studying the quasilinear setting is that it is the easiest setting for mechanism design, which we will discuss shortly. In this subsection, we will see that the clearing problem is much easier in the case of quasilinear bids.

First, we address the case where we are trying to maximize surplus (which is the most natural setting for mechanism design). The key observation here is that when bids are quasilinear, the clearing problem *decomposes* across charities.

Lemma 8 *Suppose all bids are quasilinear, and surplus is the objective. Then we can clear the market optimally by clearing the market for each charity individually. That is, for each bidder b_j, let $\pi_{b_j} = \sum_{c_i} \pi_{b_j^i}$. Then, for each charity c_i, maximize $(\sum_{b_j} \pi_{b_j^i}) - \pi_{c_i}$, under the constraint that for every bidder b_j, $\pi_{b_j^i} \leq u_j^i(\pi_{c_i})$.*

Proof: The resulting solution is certainly valid: first of all, at least as much money is collected as is given away, because $\sum_{b_j} \pi_{b_j} - \sum_{c_i} \pi_{c_i} = \sum_{b_j} \sum_{c_i} \pi_{b_j^i} - \sum_{c_i} \pi_{c_i} = \sum_{c_i}((\sum_{b_j} \pi_{b_j^i}) - \pi_{c_i})$—and the terms of this summation are the objectives of the individual optimization problems, each of which can be set at least to 0 (by setting all the variables are set to 0), so it follows that the expression is nonnegative. Second, no bidder b_j pays more than she is willing to, because $u_j - \pi_{b_j} = \sum_{c_i} u_j^i(\pi_{c_i}) - \sum_{c_i} \pi_{b_j^i} = \sum_{c_i}(u_j^i(\pi_{c_i}) - \pi_{b_j^i})$—and the terms of this summation are nonnegative by the constraints we imposed on the individual optimization problems.

All that remains to show is that the solution is optimal. Because in an optimal solution, we will extract as much payment from the bidders as possible given the π_{c_i}, all we need to show is that the π_{c_i} are set optimally by this approach. Let $\pi_{c_i}^*$ be the amount paid to charity π_{c_i} in some optimal solution. If we change this amount to π'_{c_i} and leave everything else unchanged, this will only affect the payment that we can extract from the bidders because of this particular charity, and the difference in surplus will be $\sum_{b_j} u_j^i(\pi'_{c_i}) - u_j^i(\pi^*_{c_i}) - \pi'_{c_i} + \pi^*_{c_i}$. This expression is, of course, 0 if $\pi'_{c_i} = \pi^*_{c_i}$. But now notice that this expression is maximized as a function of π'_{c_i} by the decomposed solution for this charity (the terms without π'_{c_i} in them do not matter, and of course in the decomposed solution we always set $\pi_{b_j^i} = u_j^i(\pi_{c_i})$). It follows that if we change π_{c_i} to the decomposed solution, the change in surplus will be at least 0 (and the solution will still be valid). Thus, we can change the π_{c_i} one by one to the decomposed solution without ever losing any surplus. ■

Theorem 12 *When all bids are quasilinear and surplus is the objective, DONATION-CLEARING can be done in linear time.*

Proof: By Lemma 8, we can solve the problem separately for each charity. For charity c_i, this amounts to maximizing $(\sum_{b_j} u_j^i(\pi_{c_i})) - \pi_{c_i}$ as a function of π_{c_i}. Because all its terms are piecewise linear functions, this whole function is piecewise linear, and must be maximized at one of the points where it is nondifferentiable. It follows that we need only check all the points at which one of the terms is nondifferentiable. ■

Unfortunately, the decomposing lemma does not hold for payment maximization.

Proposition 1 *When the objective is payment maximization, even when bids are quasilinear, the solution obtained by decomposing the problem across charities is in general not optimal (even with concave bids).*

Proof: Consider a single bidder b_1 placing the following quasilinear bid over two charities c_1 and c_2: $u_1^1(\pi_{c_1}) = 2\pi_{c_i}$ for $0 \leq \pi_{c_i} \leq 1$, $u_1^1(\pi_{c_1}) = 2 + \frac{\pi_{c_i}-1}{4}$ otherwise; $u_1^2(\pi_{c_2}) = \frac{\pi_{c_i}}{2}$. The decomposed solution is $\pi_{c_1} = \frac{7}{3}$, $\pi_{c_2} = 0$, for a total donation of $\frac{7}{3}$. But the solution $\pi_{c_1} = 1$, $\pi_{c_2} = 2$ is also valid, for a total donation of $3 > \frac{7}{3}$. ■

In fact, when payment maximization is the objective, DONATION-CLEARING remains (weakly) NP-complete in general.

Theorem 13 *DONATION-CLEARING is (weakly) NP-complete when payment maximization is the objective, even when every bid is concerns only one charity (and has a step-function utility function for this charity), and is quasilinear.*

Proof: That the problem is in NP follows from the fact that the more general problem is in NP. To show NP-hardness, we reduce an arbitrary KNAPSACK instance (given by m pairs $(k_i, v_i)_{1 \leq i \leq m}$, a cost limit K, and a target value V), to the following DONATION-CLEARING instance. Let there be $m+1$ charities, $c_0, c_1, \ldots, c_m$. Let there be one quasilinear bidder b_0 bidding $u_0^0(\pi_{c_0}) = 0$ for $0 \leq \pi_{c_0} \leq 1$, $u_0^0(\pi_{c_0}) = K + 1$ otherwise. Additionally, for each $1 \leq j \leq m$, let there be a bidder b_j bidding $u_j^j(\pi_{c_j}) = 0$ for $0 \leq \pi_{c_j} < k_i$, $u_j^j(\pi_{c_j}) = \epsilon v_i$ otherwise (where $\epsilon \sum_{1 \leq j \leq m} v_i < 1$). Let the target total amount donated be $K + 1 + \epsilon V$. We now show the two instances are equivalent.

First, suppose there exists a solution to the KNAPSACK instance, that is, a function $f : \{1, \ldots, m\} \to \{0, 1\}$ so that $\sum_{i=1}^{m} f(i)k_i \leq K$ and $\sum_{i=1}^{m} f(i)v_i \geq V$. Then, let $\pi_{c_0} = 1 + \epsilon V + K - \sum_{i=1}^{m} f(i)k_i$, and for $i > 0$, $\pi_{c_i} = f(i)k_i$, for a total donated of $K + 1 + \epsilon V$. Because $1 + \epsilon V + K - \sum_{i=1}^{m} f(i)k_i \geq 1$, b_0's utility is $K + 1$. For $j > 0$, b_j's utility is $f(j)\epsilon v_j$, for a total utility of $\sum_{j=1}^{m} f(j)\epsilon v_j \geq \epsilon V$ for these m bidders. It follows that the total utility is at least the total amount donated, and the outcome is valid. So there exists a solution to the DONATION-CLEARING instance.

Now suppose there exists a solution to the DONATION-CLEARING instance. Let $f : \{1, \ldots, m\} \to \{0, 1\}$ be given by $f(i) = 0$ if $\pi_{c_i} < k_i$, and $f(i) = 1$ otherwise. Because the total donated is at least $K+1+\epsilon V$, and the amount that is extractable from the bidders is at most $K+1+\sum_{j=1}^{m} f(j)\epsilon v_j$, it follows that $\sum_{j=1}^{m} f(j)v_j \geq V$. Also, because the total amount donated to charities 1 through m can be at most $K + \epsilon \sum_{1 \leq j \leq m} v_i < K + 1$, it follows that $\sum_{j=1}^{m} f(j)k_i < K + 1$. Because the k_i are integers, this means $\sum_{j=1}^{m} f(j)k_i \leq K$. So there exists a solution to the KNAPSACK instance. ■

However, when the bids are also concave, a simple greedy clearing algorithm is optimal.

Theorem 14 *Given a DONATION-CLEARING instance with payment maximization as the objective where all bids are quasilinear and concave, consider the following algorithm. Start with* $\pi_{c_i} = 0$ *for all charities. Then, letting* $\gamma_{c_i} = \frac{d \sum_{b_j} u_j^i(\pi_{c_i})}{d\pi_{c_i}}$ *(at nondifferentiable points, these derivatives should be taken from the right), increase* $\pi_{c_i^*}$ *(where* $c_i^* \in \arg\max_{c_i} \gamma_{c_i}$*), until either* $\gamma_{c_i^*}$ *is no longer the highest (in which case, recompute* c_i^* *and start increasing the corresponding payment), or* $\sum_{b_j} u_j = \sum_{c_i} \pi_{c_i}$ *and* $\gamma_{c_i^*} < 1$*. Finally, let* $\pi_{b_j} = u_j$*.*

Proof: The outcome is valid because everyone pays exactly what she is willing to, and because there is no budget deficit: $\sum_{b_j} \pi_{b_j} = \sum_{b_j} u_j = \sum_{c_i} \pi_{c_i}$. To show optimality, let $\pi_{c_i}^*$ be the amount paid to charity c_i in some optimal solution, and let π_{c_i}' be the amount paid to charity i in the solution given by the greedy algorithm. We first observe that it is not possible that for any i, $\pi_{c_i}^* \geq \pi_{c_i}'$ with at least one of these inequalities being strict. This is because at the solution found by the greedy algorithm, $\gamma_{c_i^*}$ is less than 1; hence, using concavity, if $\pi_{c_i}^* > \pi_{c_i}'$, then $\int_{\pi_{c_i}'}^{\pi_{c_i}^*} \gamma_{c_i} d\pi_{c_i} < \pi_{c_i}^* - \pi_{c_i}'$. In other words, the additional payment that needs to be made to the charity is less than the additional payment that can be collected from the bidders because of this charity. Because the surplus at the greedy algorithm's solution is 0, it follows that if for any i, $\pi_{c_i}^* \geq \pi_{c_i}'$ with at least one of these inequalities being strict, the surplus at the optimal solution woud be negative, and hence the solution would not be valid. Thus, either for all i, $\pi^*_{c_i} \leq \pi'_{c_i}$ (but in this case the greedy solution has at least as large a total payment as the optimal solution, and we are done); or there exist i, j such that $\pi^*_{c_i} > \pi'_{c_i}$ but $\pi^*_{c_j} < \pi'_{c_j}$. It cannot be the case that $\gamma_{c_i}(\pi'_{c_i}) > \gamma_{c_j}(\pi^*_{c_j})$, for then the greedy algorithm would have increased π_{c_i} beyond π'_{c_i} before increasing π_{c_j} beyond $\pi^*_{c_j}$. So, $\gamma_{c_i}(\pi'_{c_i}) \leq \gamma_{c_j}(\pi^*_{c_j})$. Because $\pi^*_{c_i} > \pi'_{c_i}$, and using concavity, if we decrease $\pi^*_{c_i}$ and simultaneously increase $\pi^*_{c_j}$ by the same amount, we will not decrease the total payment we can extract—while keeping the payment to be made to the charities the same. It follows this cannot make the solution worse or invalid. We can keep doing this until there is no longer a pair i, j such that $\pi^*_{c_i} > \pi'_{c_i}$ but $\pi^*_{c_j} < \pi_{c_j}$, and by the previous we know that for all i, $\pi^*_{c_i} \leq \pi'_{c_i}$—and hence the greedy solution is optimal. ■

(A similar greedy algorithm works when the objective is surplus and the bids are quasilinear and concave, with as only difference that we stop increasing the payments as soon as $\gamma_{c^*_i} < 1$.)

This concludes the part of this dissertation studying the complexity of the outcome optimization problem for expressive preference aggregation for donations to charities; we will study mechanism design aspects of this setting in the chapter after the next chapter, Section 5.2. In the next section, we study the complexity of the outcome optimization problem in more general settings with externalities.

3.4 Expressive preference aggregation in settings with externalities

In this section, we study the optimization problem for expressive preference aggregation in settings with externalities, as defined in Section 2.4. We study the following two computational problems. (Recall that a solution is feasible if no agent prefers the default outcome (all variables set to 0) to it.)

Definition 14 (FEASIBLE-CONCESSIONS) *We are given a concessions setting (as defined in Section 2.4). We are asked whether there exists a nontrivial feasible solution.*

Definition 15 (SW-MAXIMIZING-CONCESSIONS) *We are given a concessions setting (as defined in Section 2.4). We are asked to find a feasible solution that maximizes social welfare (the sum of the agents' utilities).*

The following shows that if the first problem is hard, the second problem is hard to approximate to any ratio.

Proposition 2 *Suppose that FEASIBLE-CONCESSIONS is NP-hard even under some constraints on the instance (but no constraint that prohibits adding another agent that derives positive utility from any nontrivial setting of the variables of the other agents). Then it is NP-hard to approximate SW-MAXIMIZING-CONCESSIONS to any positive ratio, even under the same constraints.*

Proof: We reduce an arbitrary FEASIBLE-CONCESSIONS instance to a SW-MAXIMIZING-CONCESSIONS instance that is identical, except that a single additional agent has been added that derives positive utility from any nontrivial setting of the variable(s) of the other agents, and to whose variables the other agents are completely indifferent (they cannot derive any utility from the new agent's variable(s)). If the original instance has no nontrivial feasible solution, then neither does the new instance, and the maximal social welfare that can be obtained is 0. On the other hand, if the original instance has a nontrivial feasible solution, then the new instance has a feasible solution with positive social welfare: the exact same solution is still feasible, and the new agent will get positive utility (and the others, nonnegative utility). It follows that any algorithm that approximates SW-MAXIMIZING-CONCESSIONS to some positive ratio will return a social welfare of 0 if there is no solution to the FEASIBLE-CONCESSIONS problem, and positive social welfare if there is a solution—and thus the algorithm could be used to solve an NP-hard problem. ■

3.4.1 Hardness with positive and negative externalities

We first show that if we do not make the assumption of only negative externalities, then finding a feasible solution is NP-complete even when each agent controls only one variable. (In all the problems that we study, membership in NP is straightforward, so we just give the hardness proof.)

Theorem 15 *FEASIBLE-CONCESSIONS is NP-complete, even when all utility functions decompose (and all the components u_i^k are step functions), and each agent controls only one variable.*

Proof: We reduce an arbitrary SAT instance (given by variables V and clauses C) to the following FEASIBLE-CONCESSIONS instance. Let the set of agents be as follows. For each variable $v \in V$, let there be an agent a_v, controlling a single variable x_{a_v}. Also, for every clause $c \in C$, let there be an agent a_c, controlling a single variable x_{a_c}. Finally, let there be a single agent a_0 controlling x_{a_0}. Let all the utility functions decompose, as follows: For any $v \in V$, $u_{a_v}^{a_v}(x_{a_v}) = -\delta_{x_{a_v} \geq 1}$. For any $v \in V$, $u_{a_v}^{a_0}(x_{a_0}) = \delta_{x_{a_0} \geq 1}$. For any $c \in C$, $u_{a_c}^{a_c}(x_{a_c}) = (n(c) - 2|V|)\delta_{x_{a_c} \geq 1}$ where $n(c)$ is the number of variables that occur in c in negated form. For any $c \in C$, $u_{a_c}^{a_0}(x_{a_0}) = (2|V| - 1)\delta_{x_{a_0} \geq 1}$. For any $c \in C$ and $v \in V$ where $+v$ occurs in c, $u_{a_c}^{a_v}(x_{a_v}) = \delta_{x_{a_v} \geq 1}$. For any $c \in C$ and $v \in V$ where $-v$ occurs in c, $u_{a_c}^{a_v}(x_{a_v}) = -\delta_{x_{a_v} \geq 1}$. $u_{a_0}^{a_0}(x_{a_0}) = -|C|\delta_{x_{a_0} \geq 1}$. For any $c \in C$, $u_{a_0}^{a_c}(x_{a_c}) = \delta_{x_{a_c} \geq 1}$. All the other functions are 0 everywhere. We proceed to show that the instances are equivalent.

First suppose there exists a solution to the SAT instance. Then, let $x_{a_v} = 1$ if v is set to *true* in the solution, and $x_{a_v} = 0$ if v is set to *false* in the solution. Let $x_{a_c} = 1$ for all $c \in C$, and let $x_{a_0} = 1$. Then, the utility of every a_v is at least $-1+1 = 0$. Also, the utility of a_0 is $-|C|+|C| = 0$. And, the utility of every a_c is $n(c) - 2|V| + 2|V| - 1 + pt(c) - nt(c) = n(c) - 1 + pt(c) - nt(c)$, where $pt(c)$ is the number of variables that occur positively in c and are set to *true*, and $nt(c)$ is the number of variables that occur negatively in c and are set to *true*. Of course, $pt(c) \geq 0$ and $-nt(c) \geq -n(c)$; and if at least one of the variables that occur positively in c is set to *true*, or at least one of the variables that occur negatively in c is set to *false*, then $pt(c) - nt(c) \geq -n(c) + 1$, so that the utility of a_c is at least $n(c) - 1 - n(c) + 1 = 0$. But this is always the case, because the assignment satisfies the clause. So there exists a solution to the FEASIBLE-CONCESSIONS instance.

Now suppose there exists a solution to the FEASIBLE-CONCESSIONS instance. If it were the case that $x_{a_0} < 1$, then for all the a_v we would have $x_{a_v} < 1$ (or a_v would have a negative utility), and for all the a_c we would have $x_{a_c} < 1$ (because otherwise the highest utility possible for a_c is $n(c) - 2|V| < 0$, because all the x_{a_0} are below 1). So the solution would be trivial. It follows that $x_{a_0} \geq 1$. Thus, in order for a_0 to have nonnegative utility, it follows that for all $c \in C$, $x_{a_c} \geq 1$. Now, let v be set to *true* if $x_{a_v} = 1$, and to *false* if $x_{a_v} = 0$. So the utility of every a_c is $n(c) - 2|V| + 2|V| - 1 + pt(c) - nt(c) = n(c) - 1 + pt(c) - nt(c)$. In order for this to be nonnegative, we must have (for any c) that either $nt(c) < n(c)$ (at least one variable that occurs negatively in c is set to *false*) or $pt(c) > 0$ (at least one variable that occurs positively in c is set to *true*). So we have a satisfying assignment. ■

3.4.2 Hardness with only negative externalities

Next, we show that even if we do make the assumption of only negative externalities, then finding a feasible solution is still NP-complete, even when each agent controls at most two variables.

Theorem 16 *FEASIBLE-CONCESSIONS is NP-complete, even when there are only negative externalities, all utility functions decompose (and all the components are step functions), and each agent controls at most two variables.*

Proof: We reduce an arbitrary SAT instance to the following FEASIBLE-CONCESSIONS instance. Let the set of agents be as follows. For each variable $v \in V$, let there be an agent a_v, controlling variables $x^+_{a_v}$ and $x^-_{a_v}$. Also, for every clause $c \in C$, let there be an agent a_c, controlling a single variable x_{a_c}. Let all the utility functions decompose, as follows: For any $v \in V$, $u^{a_v,+}_{a_v}(x^+_{a_v}) = -|C|\delta_{x^+_{a_v} \geq 1}$, and $u^{a_v,-}_{a_v}(x^-_{a_v}) = -|C|\delta_{x^-_{a_v} \geq 1}$. For any $v \in V$ and $c \in C$, $u^{a_c}_{a_v}(x_{a_c}) = \delta_{x_{a_c} \geq 1}$. For any $c \in C$, $u^{a_c}_{a_c}(x_{a_c}) = -\delta_{x_{a_c} \geq 1}$. For any $c \in C$ and $v \in V$ where $+v$ occurs in c, $u^{a_v,+}_{a_c}(x^+_{a_v}) = \delta_{x^+_{a_v} \geq 1}$; and for any $c \in C$ and $v \in V$ where $-v$ occurs in c, $u^{a_v,-}_{a_c}(x^-_{a_v}) = \delta_{x^-_{a_v} \geq 1}$. All the other functions are 0 everywhere. We proceed to show that the instances are equivalent.

First suppose there exists a solution to the SAT instance. Then, let $x^+_{a_v} = 1$ if v is set to *true* in the solution, and $x^+_{a_v} = 0$ otherwise; and, let $x^-_{a_v} = 1$ if v is set to *false* in the solution, and $x^-_{a_v} = 0$ otherwise. Let $x_{a_c} = 1$ for all $c \in C$. Then, the utility of every a_v is $-|C| + |C| = 0$. Also, the utility of every a_c is at least $-1 + 1$ (because all clauses are satisfied in the solution, there is at least one $+v \in c$ with $x^+_{a_v} = 1$, or at least one $-v \in c$ with $x^-_{a_v} = 1$. So there exists a solution to the FEASIBLE-CONCESSIONS instance.

Now suppose there exists a solution to the FEASIBLE-CONCESSIONS instance. At least one of the $x^+_{a_v}$ or at least one of the $x^-_{a_v}$ must be set nontrivially (≥ 1), because otherwise no x_{a_c} can be set nontrivially. But this implies that for any clause $c \in C$, $x_{a_c} \geq 1$ (for otherwise the a_v with a nontrivial setting of its variables would have negative utility). So that none of the a_c have nonnegative utility, it must be the case that for any $c \in C$, either there is at least one $+v \in c$ with $x^+_{a_v} \geq 1$, or at least one $-v \in c$ with $x^-_{a_v} \geq 1$. Also, for no variable $v \in V$ can it be the case that both $x^+_{a_v} \geq 1$ and $x^-_{a_v} \geq 1$, as this would leave a_v with negative utility. But then, letting v be set to *true* if $x^+_{a_v} \geq 1$, and to *false* otherwise must satisfy every clause. So there exists a solution to the SAT instance. ■

3.4.3 An algorithm for the case of only negative externalities and one variable per agent

We have shown that with both positive and negative externalities, finding a feasible solution is hard even when each agent controls only one variable; and with only negative externalities, finding a feasible solution is hard even when each agent controls at most two variables. In this subsection we show that these results are, in a sense, tight, by giving an algorithm for the case where there are only negative externalities and each agent controls only one variable. Under some minimal assumptions, this algorithm will return (or converge to) the maximal feasible solution, that is, the solution in which the variables are set to values that are as large as possible. Although the setting for this algorithm may appear very restricted, it still allows for the solution of interesting problems. For example, consider governments negotiating over by how much to reduce their countries' carbon dioxide emissions, for the purpose of reducing global warming.

We will not require the assumption of decomposing utility functions in this subsection (except where stated). The following claim shows the sense in which the maximal solution is well-defined in the setting under discussion (there cannot be multiple maximal solutions, and under a continuity assumption, a maximal solution exists).

Theorem 17 *In a concessions setting with only negative externalities and in which each agent controls only one variable, let $x_1, x_2, \ldots, x_n$ and $x'_1, x'_2, \ldots, x'_n$ be two feasible solutions. Then $\max\{x_1, x'_1\}, \max\{x_2, x'_2\}, \ldots, \max\{x_n, x'_n\}$ is also a feasible solution. Moreover, if all the utility functions are continuous, then, letting X_i be the set of values for x_i that occur in some feasible solution, $\sup(X_1), \sup(X_2), \ldots, \sup(X_n)$ is also a feasible solution.*

Proof: For the first claim, we need to show that every agent i receives nonnegative utility in the proposed solution. Suppose without loss of generality that $x_i \geq x'_i$. Then, we have $u_i(\max\{x_1, x'_1\}, \max\{x_2, x'_2\}, \ldots, \max\{x_i, x'_i\}, \ldots, \max\{x_n, x'_n\}) = u_i(\max\{x_1, x'_1\}, \max\{x_2, x'_2\}, \ldots, x_i, \ldots, \max\{x_n, x'_n\}) \geq u_i(x_1, x_2, \ldots, x_i, \ldots, x_n)$, where the inequality stems from the fact that there are only negative externalities. But the last expression is nonnegative because the first solution is feasible.

For the second claim, we will find a sequence of feasible solutions that converges to the proposed solution. By continuity, any agent's utility at the limit point must be the limit of that agent's utility in the sequence of feasible solutions; and because these solutions are all feasible, this limit must be nonnegative. For each agent i, let $\{(x_1^{i,j}, x_2^{i,j}, \ldots, x_n^{i,j})\}_{j \in \mathbb{N}}$ be a sequence of feasible solutions with $\lim_{j \to \infty} x_i^{i,j} = \sup(X_i)$. By repeated application of the first claim, we have that (for any j) $\max_i\{x_1^{i,j}\}, \max_i\{x_2^{i,j}\}, \ldots, \max_i\{x_n^{i,j}\}$ is a feasible solution, giving us a new sequence of feasible solutions. Moreover, because this new sequence dominates every one of the original sequences, and for each agent i there is at least one original sequence where the ith element converges to $\sup(X_i)$, the sequence converges to the solution $\sup(X_1), \sup(X_2), \ldots, \sup(X_n)$. ■

We are now ready to present the algorithm. First, we give an informal description. The algorithm proceeds in stages: in each stage, for each agent, it eliminates all the values for that agent's variable that would result in a negative utility for that agent regardless of how the other agents set their variables (given that they use values that have not yet been eliminated).

ALGORITHM 1
1. **for** $i := 1$ **to** n $\{$
2. $X_i^0 := \mathbb{R}^{\geq 0}$ (alternatively, $X_i^0 := [0, M]$ where M is some upper bound) $\}$
3. $t := 0$
4. **repeat until** $((\forall i)\ X_i^t = X_i^{t-1})$ $\{$
5. $t := t + 1$
6. **for** $i := 1$ **to** n $\{$
7. $X_i^t := \{x_i \in X_i^{t-1} : \exists x_1 \in X_1^{t-1}, x_2 \in X_2^{t-1}, \ldots, x_{i-1} \in X_{i-1}^{t-1}, x_{i+1} \in X_{i+1}^{t-1}, \ldots, x_n \in X_n^{t-1} : u_i(x_1, x_2, \ldots, x_i, \ldots, x_n) \geq 0\}$ $\}$ $\}$

The set updates in step 7 of the algorithm are simple to perform, because all the X_i^t always take the form $[0, r]$, $[0, r)$, or $\mathbb{R}^{\geq 0}$ (because we are in a concessions setting), and in step 7 it never hurts to choose values for $x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$ that are as large as possible (because we have only negative externalities). Roughly, the goal of the algorithm is for $\sup(X_1^t), \sup(X_2^t), \ldots, \sup(X_n^t)$ to converge to the maximal feasible solution (that is, the feasible solution such that all of the variables are set to values at least as large as in any other feasible solution). We now show that the algorithm is sound, in the sense that it does not eliminate values of the x_i that occur in feasible solutions.

Theorem 18 *Suppose we are running Algorithm 1 in a concessions setting with only negative externalities where each agent controls only one variable. If for some t, $r \notin X_i^t$, then there is no feasible solution with x_i set to r.*

Proof: We will prove this by induction on t. For $t = 0$ the theorem is vacuously true. Now suppose we have proved it true for $t = k$; we will prove it true for $t = k + 1$. By the induction assumption, all feasible solutions lie within $X_1^k \times \ldots \times X_n^k$. But if $r \neq X_i^{k+1}$, this means exactly that there is no feasible solution in $X_1^k \times \ldots \times X_n^k$ with $x_i = r$. It follows there is no feasible solution with $x_i = r$ at all. ■

However, the algorithm is not complete, in the sense that (for some "unnatural" functions) it does not eliminate all the values of the x_i that do not occur in feasible solutions.

Proposition 3 *Suppose we are running Algorithm 1 in a concessions setting with only negative externalities where each agent controls only one variable. For some (discontinuous) utility functions (even ones that decompose), the algorithm will terminate with nontrivial X_i^t even though the only feasible solution is the zero solution.*

Proof: Consider the following symmetric example:

- $u_1^1(x_1) = -x_1$ for $x_1 < 1$, $u_1^1(x_1) = -2$ otherwise;
- $u_1^2(x_2) = (x_2)^2$ for $x_2 < 1$, $u_1^2(x_2) = 1$ otherwise;
- $u_2^1(x_1) = (x_1)^2$ for $x_1 < 1$, $u_2^1(x_1) = 1$ otherwise;
- $u_2^2(x_2) = -x_2$ for $x_2 < 1$, $u_2^2(x_2) = -2$ otherwise.

There is no solution with $x_1 \geq 1$ or $x_2 \geq 1$, because the corresponding agent's utility would definitely be negative. In order for agent 1 to have nonnegative utility we must have $(x_2)^2 \geq x_1$. Unless they are both zero, this implies $x_2 > x_1$. Similarly, in order for agent 2 to have nonnegative utility we must have $(x_1)^2 \geq x_2$, and unless they are both zero, this implies $x_1 > x_2$. It follows that the only solution is the zero solution. Unfortunately, in the algorithm, we first get $X_1^1 = X_2^1 = [0, 1)$; then also, we get $X_1^2 = X_2^2 = [0, 1)$ (for any $x_1 < 1$, we can set $x_2 = \sqrt{(}x_1) < 1$ and agent 1 will get utility 0, and similarly for agent 2). So the algorithm terminates. ■

However, if we make some reasonable assumptions on the utility functions (specifically, that they are either continuous or piecewise constant), then the algorithm is complete, in the sense that it will (eventually) remove any values of the x_i that are too large to occur in any feasible solution. Thus, the algorithm converges to the solution. We will present the case of continuous utility functions first.

Theorem 19 *Suppose we are running Algorithm 1 in a concessions setting with only negative externalities where each agent controls only one variable. Suppose that all the utility functions are continuous. Also, suppose that all the X_i^0 are initialized to $[0, M]$. Then, all the X_i^t are closed sets. Moreover, if the algorithm terminates after the tth iteration of the* **repeat** *loop, then* $\sup(X_1^t), \sup(X_2^t), \ldots, \sup(X_n^t)$ *is feasible, and it is the maximal solution. If the algorithm does not terminate, then* $\lim_{t\to\infty} \sup(X_1^t), \lim_{t\to\infty} \sup(X_2^t), \ldots, \lim_{t\to\infty} \sup(X_n^t)$ *is feasible, and it is the maximal solution.*

Proof: First we show that all the X_i^t are closed sets, by induction on t. For $t = 0$, the claim is true, because $[0, M]$ is a closed set. Now suppose they are all closed for $t = k$; we will show them to be closed for $t = k + 1$. In the step in the algorithm in which we set X_i^{k+1}, in the choice of $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$, we may as well always set each of these x_j to $\sup(X_j^k)$ (which is inside X_j^k because X_j^k is closed by the induction assumption), because this will maximize agent i's utility. It follows that $X_i^{k+1} = \{x_i : u_i(\sup(X_1^k), \ldots, \sup(X_{i-1}^k), x_i, \sup(X_{i+1}^k), \ldots, \sup(X_n^k)) \geq 0\}$. But because u_i is continuous, this set must be closed by elementary results from analysis.

Now we proceed to show the second claim. Because each X_i^t is closed, it follows that $\sup(X_i^t) \in X_i^t$. This implies that, for every agent i, there exist $x_1 \in X_1^{t-1}, x_2 \in X_2^{t-1}, \ldots, x_{i-1} \in X_{i-1}^{t-1}, x_{i+1} \in X_{i+1}^{t-1}, \ldots, x_n \in X_n^{t-1}$ such that $u_i(x_1, x_2, \ldots, \sup(X_i^t), \ldots, x_n) \geq 0$. Because for every agent i', $X_{i'}^t = X_{i'}^{t-1}$ (the algorithm terminated), this is equivalent to saying that there exist $x_1 \in X_1^t, x_2 \in X_2^t, \ldots, x_{i-1} \in X_{i-1}^t, x_{i+1} \in X_{i+1}^t, \ldots, x_n \in X_n^t$ such that $u_i(x_1, x_2, \ldots, \sup(X_i^t), \ldots, x_n) \geq 0$. Of course, for each of these $x_{i'}$, we have $x_{i'} \leq \sup(X_{i'}^t)$. Because there are only negative externalities, it follows that $u_i(\sup(X_1^t), \sup(X_2^t), \ldots, \sup(X_i^t), \ldots, \sup(X_n^t)) \geq u_i(x_1, x_2, \ldots, \sup(X_i^t), \ldots, x_n) \geq 0$. Thus, $\sup(X_1^t), \sup(X_2^t), \ldots, \sup(X_n^t)$ is feasible. It is also maximal by Theorem 18.

Finally, we prove the third claim. For any agent i, for any t, we have $u_i(\sup(X_1^{t-1}), \sup(X_2^{t-1}), \ldots, \lim_{t'\to\infty} \sup(X_i^{t'}), \ldots, \sup(X_n^{t-1})) \geq u_i(\sup(X_1^{t-1}), \sup(X_2^{t-1}), \ldots, \sup(X_i^t), \ldots, \sup(X_n^{t-1}))$ (because the X_i^t are decreasing in t, and we are in a concessions setting). The last expression evaluates to a nonnegative quantity, using the same reasoning as in the proof of the second claim with the fact that $\sup(X_i^t) \in X_i^t$. But then, by continuity, $0 \leq \lim_{t\to\infty}(u_i(\sup(X_1^{t-1}), \sup(X_2^{t-1}), \ldots, \lim_{t'\to\infty} \sup(X_i^{t'}), \ldots, \sup(X_n^{t-1}))) = u_i(\lim_{t\to\infty} \sup(X_1^{t-1}), \lim_{t\to\infty} \sup(X_2^{t-1}), \ldots, \lim_{t'\to\infty} \sup(X_i^{t'}), \ldots, \lim_{t\to\infty} \sup(X_n^{t-1})) = u_i(\lim_{t\to\infty} \sup(X_1^t),$

$\lim_{t\to\infty}\sup(X_2^t),\ldots,\lim_{t\to\infty}\sup(X_i^t),\ldots,\lim_{t\to\infty}\sup(X_n^t)$. It follows that $\lim_{t\to\infty}\sup(X_1^t)$, $\lim_{t\to\infty}\sup(X_2^t),\ldots,\lim_{t\to\infty}\sup(X_n^t))$ is feasible. It is also maximal by Theorem 18. ■

We observe that piecewise constant functions are not continuous, and thus Theorem 19 does not apply to the case where the utility functions are piecewise constant. Nevertheless, the algorithm works on such utility functions, and we can even prove that the number of iterations is linear in the number of pieces. There is one caveat: the way we have defined piecewise constant functions (as linear combinations of step functions $\delta_{x\geq a}$), the maximal solution is not well defined (the set of feasible points is never closed on the right, i.e. it does not include its least upper bound). To remedy this, call a feasible solution *quasi-maximal* if there is no feasible solution that is larger (that is, all the x_i are set to values that are at least as large) and that gives some agent a different utility (so it is maximal for all intents and purposes).

Theorem 20 *Suppose we are running Algorithm 1 in a concessions setting with only negative externalities where each agent controls only one variable. If all the utility functions decompose and all the components u_i^k are piecewise constant with finitely many steps (the range of the u_i^k is finite), then the algorithm will terminate after at most T iterations of the* **repeat** *loop, where T is the total number of steps in all the self-components u_i^i (i.e. the sum of the sizes of the ranges of these functions). Moreover, if the algorithm terminates after the tth iteration of the* **repeat** *loop, then any solution $(x_1, x_2, \ldots, x_n)$ with for all i, $x_i \in \arg\max_{x_i\in X_i^t} \sum\limits_{j\neq i} u_j^i(x_i)$, is feasible and quasi-maximal.*

Proof: If for some i and t, $X_i^t \neq X_i^{t-1}$, it must be the case that for some value r in the range of u_i^i, the preimage of this value is in $X_i^{t-1} - X_i^t$ (it has just been eliminated from consideration). Informally, one of the steps of the function u_i^i has been eliminated from consideration. Because this must occur for at least one agent in every iteration of the **repeat** loop before termination, it follows that there can be at most T iterations before termination. Now, if the algorithm terminates after the tth iteration of the **repeat** loop, and a solution $(x_1, x_2, \ldots, x_n)$ with for all i, $x_i \in \arg\max_{x_i\in X_i^t} \sum\limits_{j\neq i} u_j^i(x_i)$ is chosen, it follows that each agent derives as much utility from the other agents' variables as is possible with the sets X_i^t (because of the assumption of only negative externalities, any setting of a variable that maximizes the total utility for the other agents also maximizes the utility for each individual other agent). We know that for each agent i, there is at least some setting of the other agents' variables within the X_j^t that will give agent i enough utility to compensate for the setting of its own variable (by the definition of X_i^t and using the fact that $X_j^t = X_j^{t-1}$, as the algorithm has terminated); and thus it follows that the utility maximizing setting is also enough to make i's utility nonnegative. So the solution is feasible. It is also quasi-maximal by Theorem 18. ■

Algorithm 1 can be extended to cases where some agents control multiple variables, by interpreting x_i in the algorithm as the *vector* of agent i's variables (and initializing the X_i^0 as cross products of sets). However, the next proposition shows how this extension of Algorithm 1 fails.

Proposition 4 *Suppose we are running the extension of Algorithm 1 just described in a concessions setting with only negative externalities. When some agents control more than one variable, the algorithm may terminate with nontrivial X_i^t even though the only feasible solution is the zero solution* *(all variables set to* 0*), even when all of the utility functions decompose and all of the components* $u_i^{k,j}$ *are step functions (or continuous functions).*

Proof: Let each of three agents control two variables, with utility functions as follows:

- $u_1^{1,1}(x_1^1) = -3\delta_{x_1^1 \geq 1}$
- $u_1^{1,2}(x_1^2) = -3\delta_{x_1^2 \geq 1}$
- $u_2^{2,1}(x_2^1) = -3\delta_{x_2^1 \geq 1}$
- $u_2^{2,2}(x_2^2) = -3\delta_{x_2^2 \geq 1}$
- $u_3^{3,1}(x_3^1) = -3\delta_{x_3^1 \geq 1}$
- $u_3^{3,2}(x_3^2) = -3\delta_{x_3^2 \geq 1}$
- $u_1^{2,1}(x_2^1) = 2\delta_{x_2^1 \geq 1}$
- $u_1^{3,1}(x_3^1) = 2\delta_{x_3^1 \geq 1}$
- $u_2^{1,1}(x_1^1) = 2\delta_{x_1^1 \geq 1}$
- $u_2^{3,2}(x_3^2) = 2\delta_{x_3^2 \geq 1}$
- $u_3^{1,2}(x_1^2) = 2\delta_{x_1^2 \geq 1}$
- $u_3^{2,2}(x_2^2) = 2\delta_{x_2^2 \geq 1}$

Increasing any one of the variables to a value of at least 1 will decrease the corresponding agent's utility by 3, and will raise only one other agent's utility, by 2. It follows that there is no feasible solution besides the zero solution, because any other solution will have negative social welfare (total utility), and hence at least one agent must have negative utility.

In the algorithm, after the first iteration, it becomes clear that no agent can set both its variables to values of at least 1 (because each agent can derive at most $4 < 6$ utility from the other agents' variables). Nevertheless, for any agent, it still appears possible at this stage to set either (but not both) of its variables to a value of at least 1. Unfortunately, in the next iteration, this still appears possible (because each of the other agents could set the variable that is beneficial to this agent to a value of at least 1, leading to a utility of $4 > 3$ for the agent). It follows that the algorithm gets stuck.

These utility functions are easily made continuous, while changing neither the algorithm's behavior on them nor the set of feasible solutions—for instance, by making each function linear on the interval $[0, 1]$. ■

In the next subsection, we discuss *maximizing social welfare* under the conditions under which we showed Algorithm 1 to be successful in finding the maximal solution.

3.4.4 Maximizing social welfare remains hard

In a concessions setting with only negative externalities where each agent controls only one variable, the algorithm we provided in the previous subsection returns the *maximal* feasible solution, in a linear number of rounds for utility functions that decompose into piecewise constant functions. However, this may not be the most desirable solution. For instance, we may be interested in the feasible solution with the highest social welfare (that is, the highest sum of the agents' utilities). In this subsection we show that finding this solution remains hard, even in the setting in which Algorithm 1 finds the maximal solution fast.

Theorem 21 *The decision variant of SW-MAXIMIZING-CONCESSIONS (does there exist a feasible solution with social welfare $\geq K$?) is NP-complete, even when there are only negative externalities, all utility functions decompose (and all the components u_i^k are step functions), and each agent controls only one variable.*

Proof: We reduce an arbitrary EXACT-COVER-BY-3-SETS instance (given by a set S and subsets $S_1, S_2, \ldots, S_q$ ($|S_i| = 3$) to cover S with, without any overlap) to the following SW-MAXIMIZING-CONCESSIONS instance. Let the set of agents be as follows. For every S_i there is an agent a_{S_i}. Also, for every element $s \in S$ there is an agent a_s. Every agent a controls a single variable x_a. Let all the utility functions decompose, as follows: For any S_i, $u_{a_{S_i}}^{a_{S_i}}(x_{a_{S_i}}) = -7\delta_{x_{a_{S_i}} \geq 1}$. For any S_i and for any s, $u_{a_{S_i}}^{a_s}(x_{a_s}) = 7\delta_{x_{a_s} \geq 1}$. For any s, $u_{a_s}^{a_s}(x_{a_s}) = -\delta_{x_{a_s} \geq 1}$. For any s and for any S_i with $s \in S_i$, $u_{a_s}^{a_{S_i}}(x_{a_{S_i}}) = \frac{1}{q(s)-1}\delta_{x_{a_{S_i}} \geq 1}$, where $q(s)$ is the number of sets S_i with $s \in S_i$. Let the target social welfare be $7q(|S|-1) + 7\frac{|S|}{3}$. All the other functions are 0 everywhere. We proceed to show that the two instances are equivalent. First, suppose there exists a solution to the EXACT-COVER-BY-3-SETS instance. Then, let $x_{a_{S_i}} = 0$ if S_i is in the cover, and $x_{a_{S_i}} = 1$ otherwise. For all s, let $x_s = 1$. Then a_{S_i} receives a utility of $7|S|$ if S_i is in the cover, and $7(|S|-1)$ otherwise. Furthermore, for all $s \in S$, a_s receives a utility of $(q(s)-1)\frac{1}{q(s)-1} - 1 = 0$ (because for exactly $q(s)-1$ of the $q(s)$ subsets S_i with s in it, the corresponding agent has its variable set to 1: the only exception is the subset S_i that contains s and is in the cover). It follows that all the agents receive nonnegative utility, and the total utility (social welfare) is $7q(|S|-1) + 7\frac{|S|}{3}$. So there exists a solution to the SW-MAXIMIZING-CONCESSIONS instance. Now, suppose that there exists a solution to the SW-MAXIMIZING-CONCESSIONS instance. We first observe that if for some $s \in S$, $x_{a_s} < 1$, the total utility (social welfare) can be at most $7q(|S|-1) + 2|S| < 7q(|S|-1) + 7\frac{|S|}{3}$ (because each a_{S_i} can receive at most $7(|S|-1)$, and each a_s can receive at most $q(s)\frac{1}{q(s)-1}$, and because $q(s) \geq 2$ this can be at most 2). So it must be the case that $x_{a_s} \geq 1$ for all $s \in S$. It follows that, in order for none of these a_s to have nonnegative utility, for every $s \in S$, there are at least $q(s)-1$ subsets S_i with $x_{a_{S_i}} \geq 1$ and $s \in S_i$. In other words, for every $s \in S$, there is at most one subset S_i with $s \in S_i$ with $x_{a_{S_i}} < 1$. In other words again, the subsets S_i with $s \in S_i$ with $x_{a_{S_i}} < 1$ are disjoint (and so there are at most $\frac{|S|}{3}$ of them. However, if there were only $k \leq \frac{|S|}{3} - 1$ subsets S_i with $x_{a_{S_i}} < 1$, then the total utility (social welfare) can be at most $7q(|S|-1) + 7k + |S| - 3k$ (each a_{S_i} receives at least $7(|S|-1)$, and they receive no more unless they are among the k, in which case they receive an additional 7; and every a_s receives 0 unless it is in none of the k disjoint subsets S_i, in which case it will receive at most 1 (because $q(s) \geq 2$, so $\frac{1}{q(s)-1} \leq 1$)—but of course there can be at most $|S| - 3k$

such agents). But $7q(|S|-1)+7k+|S|-3k \leq 7q(|S|-1)+|S|+4(\frac{|S|}{3}-1) = 7q(|S|-1)+7\frac{|S|}{3}-4$, which is less than the target. It follows there are exactly $\frac{|S|}{3}$ disjoint subsets S_i with $x_{a_{S_i}} < 1$—an exact cover. So there exists a solution to the EXACT-COVER-BY-3-SETS instance. ■

3.4.5 Hardness with only two agents

So far, we have not assumed any bound on the number of agents. A natural question to ask is whether such a bound makes the problem easier to solve. In this subsection, we show that the problem of finding a feasible solution in a concessions setting with only negative externalities remains NP-complete even with only two agents (when there is no restriction on how many variables each agent controls).

Theorem 22 *FEASIBLE-CONCESSIONS is NP-complete, even when there are only two agents, there are only negative externalities, and all utility functions decompose (and all the components $u_i^{k,j}$ are step functions).*

Proof: We reduce an arbitrary KNAPSACK instance (given by r pairs (c_i, v_i), a cost constraint C and a value objective V) to the following FEASIBLE-CONCESSIONS instance with two agents. Agent 1 controls only one variable, x_1^1. Agent 2 controls r variables, $x_2^1, x_2^2, \ldots, x_2^r$. Agent 1's utility function is $u_1(x_1^1, x_2^1, x_2^2, \ldots, x_2^r) = -V\delta_{x_1^1 \geq 1} + \sum_{j=1}^{r} v_j \delta_{x_2^j \geq 1}$. Agent 2's utility function is $u_2(x_1^1, x_2^1, x_2^2, \ldots, x_2^r) = C\delta_{x_1^1 \geq 1} - \sum_{j=1}^{r} c_j \delta_{x_2^j \geq 1}$. We proceed to show that the instances are equivalent.

Suppose there is a solution to the KNAPSACK instance, that is, a subset $S \subseteq N$ such that $\sum_{j \in S} c_i \leq C$ and $\sum_{j \in S} v_i \geq V$. Then, let $x_1^1 = 1$, and for any $1 \leq j \leq r$, let $x_2^j = \delta_{j \in S}$. Then $u_1(x_1^1, x_2^1, x_2^2, \ldots, x_2^r) = -V + \sum_{j \in S} v_j \geq 0$. Also, $u_2(x_1^1, x_2^1, x_2^2, \ldots, x_2^r) = C - \sum_{j \in S} c_j \geq 0$. So there is a solution to the FEASIBLE-CONCESSIONS instance.

Now suppose there is a solution to the FEASIBLE-CONCESSIONS instance, that is, a nonzero setting of the variables $(x_1^1, x_2^1, x_2^2, \ldots, x_2^r)$ such that $u_1(x_1^1, x_2^1, x_2^2, \ldots, x_2^r) \geq 0$ and $u_2(x_1^1, x_2^1, x_2^2, \ldots, x_2^r) \geq 0$. If it were the case that $x_1^1 < 1$, then either all of agent 2's variables are set smaller than 1 (in which case x_1^1 must be nonzero and agent 1 gets negative utility), or at least one of agent 2's variables is nonzero (in which case agent 2 gets negative utility because the setting of x_1^1 is worthless to it). It follows that $x_1^1 \geq 1$. Thus, in order for agent 1 to get nonnegative utility, we must have $\sum_{j=1}^{r} v_j \delta_{x_2^j \geq 1} \geq V$. Let $S = \{j : x_2^j \geq 1\}$. Then it follows that $\sum_{j \in S} v_j \geq V$. Also, in order for agent 2 to get nonnegative utility, we must have $\sum_{j \in S} c_j \sum_{j=1}^{r} c_j \delta_{x_2^j \geq 1} \leq C$. So there is a solution to the KNAPSACK instance. ■

3.4.6 A special case that can be solved to optimality using linear programming

Finally, in this subsection, we demonstrate a special case in which we can find the feasible outcome that maximizes social welfare (or any other linear objective) in polynomial time, using linear programming. (Linear programs can be solved in polynomial time [Khachiyan, 1979].) The special case is the one in which all the utility functions decompose into piecewise linear, concave components. For this result we will need no additional assumptions (no bounds on the number of agents or variables per agent, *etc.*).

Theorem 23 *If all of the utility functions decompose, and all of the components $u_i^{k,j}$ are piecewise linear and concave, then SW-MAXIMIZING-CONCESSIONS can be solved in polynomial time using linear programming.*

Proof: Let the variables of the linear program be the x_i^j and the $u_i^{k,j}$. We use the following linear constraints: (1) For any i, we require $\sum_{k=1}^{n} \sum_{j=1}^{m_k} u_i^{k,j} \geq 0$; (2) For any i, k, j, for any linear function $l(x_k^j)$ that coincides with one of the segments of the function $u_i^{k,j}(x_k^j)$, we require $u_i^{k,j} \leq l(x_k^j)$.

The key observation is that for any value of x_k^j, the constraints allow one to set the variable $u_i^{k,j}$ to the value $u_i^{k,j}(x_k^j)$, but no larger: because the function $u_i^{k,j}(x_k^j)$ is concave, only the constraint corresponding to the segment that x_k^j is on is binding, and the constraints corresponding to other segments are not violated.

For the linear program's objective, we use $\sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{j=1}^{m_k} u_i^{k,j}$, which is the social welfare. ■

3.5 Summary

In this chapter, we studied the complexity of the outcome optimization problem for the four settings introduced in Chapter 2. While most voting rules are easy to execute, a few are not, including the Slater and Kemeny rules. In Section 3.1, we gave a powerful preprocessing technique for computing Slater rankings, showing that if a subset of the candidates consists of similar candidates, this subset can be solved recursively. We also gave an efficient algorithm for *finding* such a set of similar candidates, and provided experimental results showing the effectiveness of this preprocessing technique. Finally, we used the technique of similar sets to show that computing an optimal Slater ranking is NP-hard, even in the absence of pairwise ties.

In Section 3.2, we turned to the winner determination problem in combinatorial auctions. We studied the setting where there is a graph (with some desired property), with the items as vertices, and every bid bids on a connected set of items. Two computational problems arise: 1) clearing the auction when given the item graph, and 2) constructing an item graph (if one exists) with the desired property. We showed that given an item graph *with bounded treewidth*, the clearing problem can be solved in polynomial time (and every combinatorial auction instance has some treewidth; the complexity is exponential in only that parameter). We then gave an algorithm for constructing an item tree (treewidth 1) if such a tree exists. We showed why this algorithm does not work for treewidth greater than 1, but left open whether item graphs of (say) treewidth 2 can be constructed in polynomial time (although we did show that finding the item graph with the fewest edges is NP-complete (even when a graph of treewidth 2 exists). We showed that the problems become hard if a bid is allowed to have more than one connected component.

In Section 3.3, we studied the outcome optimization problem for the setting of expressive negotation over donations to charities. We showed that this problem is NP-complete to approximate to any ratio even in very restricted settings. Subsequently, we gave a mixed integer program formulation of the clearing problem, and show that for concave bids, the program reduces to a linear program. We then showed that the clearing problem for a subclass of concave bids is at least as hard as a linear feasibility problem. Subsequently, we showed that the clearing problem is much easier when bids are quasilinear—for surplus, the problem decomposes across charities, and for payment maximization, a greedy approach is optimal if the bids are concave (although this latter problem is weakly NP-complete when the bids are not concave).

Finally, in Section 3.4, we studied the outcome optimization problem for the setting of expressive negotation in settings with externalities. The following table gives a summary of our results in that domain.

Restriction	**Complexity**
one variable per agent	NP-complete to find nontrivial feasible solution
negative externalities; two variables per agent	NP-complete to find nontrivial feasible solution
negative externalities; one variable per agent	Algorithm 1 finds maximal feasible solution (linear time for utilities that decompose into piecewise constant functions); NP-complete to find social-welfare maximizing solution
negative externalities; two agents	NP-complete to find nontrivial feasible solution
utilities decompose; components piecewise linear, concave	linear programming finds social welfare maximizing solution

Complexity of finding solutions in concessions settings. All of the hardness results hold even if the utility functions decompose into step functions.

One issue that we have not yet considered is that the agents will report their preferences *strategically*, that is, they will report them truthfully if and only if it is in their best interest to do so. This will be addressed in the deeper levels of the hierarchy, which will be the focus of the remainder of this dissertation. To prepare us for this, the next chapter reviews some basic concepts and results from *mechanism design*, which is the study of creating preference aggregation methods that are robust to this strategic behavior.

Chapter 4

Mechanism Design

Honesty is the best policy - when there is money in it.

Mark Twain

In order for a preference aggregator to choose a good outcome, she needs to be provided with the agents' (relevant) preferences. Usually, the only way of learning these preferences is by having the agents report them. Unfortunately, in settings where the agents are self-interested, they will report these preferences truthfully if and only if it is in their best interest to do so. Thus, the preference aggregator has the difficult task of not only choosing good outcomes for the given preferences, but also choosing outcomes in such a way that agents will not have any incentive to misreport their preferences. This is the topic of *mechanism design*, and the resulting outcome selection functions are called *mechanisms*.

This chapter gives an introduction to some basic concepts and results in mechanism design. In Section 4.1, we review basic concepts in mechanism design (although discussions of the game-theoretic justifications for this particular framework, in particular the *revelation principle*, will be postponed to Chapter 7). In Section 4.2, we review the famous and widely-studied *Vickrey-Clarke-Groves* mechanisms and their properties. In Section 4.3, we briefly review some other positive results (mechanisms that achieve particular properties), while in Section 4.4, we briefly review some key impossibility results (combinations of properties that no mechanism can achieve).

4.1 Basic concepts

If all of the agents' preferences were public knowledge, there would be no need for mechanism design—all that would need to be done is solve the outcome optimization problem. Techniques from mechanism design are useful and necessary only in settings in which agents' have *private information* about their preferences. Formally, we say that each agent i has a privately known *type* θ_i that corresponds to that agent's private information, and we denote by Θ_i the space of all of agent i's possible types. In general, it is possible to have private information that has implications for how other agents value outcomes—for example, one agent may privately know that the painting that is being auctioned is a forgery, which would be relevant to other agents that may not know this [Ito *et al.*, 2002, 2003, 2004]. In this dissertation, as is most commonly done in the mechanism design literature, we will only consider private information about the agent's own preferences (which is the most common type of private information). We model these preferences by saying that each agent i has a utility function $u_i : \Theta_i \times O \rightarrow \mathbb{R}$, where $u_i(\theta_i, o)$ gives the agent's utility for outcome o when the agent has type θ_i. The utility function u_i is common knowledge, but it is still impossible for other agents to precisely assess agent i's utility for a given outcome o without knowing agent i's type. For example, in an auction for a single item, an agent's type θ_i could be simply that agent's valuation for the item. Then, the agent's utility for an outcome in which he receives the item will be θ_i (not counting any payments to be made by the agent), and the utility is 0 otherwise. Hence, the utility function is common knowledge, but one still needs to know the agent's type to assess the agent's utility for (some) outcomes.

A *direct-revelation mechanism* asks each agent to report its private information, and chooses an outcome based on this (and potentially some random bits). It will generally be convenient not to consider payments imposed by the mechanism as part of the outcome, so that the mechanism also needs to specify payments to be made by/to agents. Formally:

Definition 16

- *A* deterministic direct-revelation mechanism without payments *consists of an outcome selection function* $o : \Theta_1 \times \ldots \times \Theta_n \rightarrow O$*.*

- *A* randomized direct-revelation mechanism without payments *consists of a distribution selection function* $p : \Theta_1 \times \ldots \times \Theta_n \rightarrow \Delta(O)$*, where* $\Delta(O)$ *is the set of probability distributions over* O*.*

- *A* deterministic direct-revelation mechanism with payments *consists of an outcome selection function* $o : \Theta_1 \times \ldots \times \Theta_n \rightarrow O$ *and for each agent* i*, a payment selection function* $\pi_i : \Theta_1 \times \ldots \times \Theta_n \rightarrow \mathbb{R}$*, where* $\pi_i(\theta_1, \ldots, \theta_n)$ *gives the payment made by agent* i *when the reported types are* $\theta_1, \ldots, \theta_n$*.*

- *A* randomized direct-revelation mechanism with payments *consists of a distribution selection function* $p : \Theta_1 \times \ldots \times \Theta_n \rightarrow \Delta(O)$*, and for each agent* i*, a payment selection function* $\pi_i : \Theta_1 \times \ldots \times \Theta_n \rightarrow \mathbb{R}$*.*

In some settings, it makes sense to think of an agent's type θ_i as being drawn from a (commonly known) prior distribution over Θ_i. In this case, while each agent still only knows its own type, each agent can use the commonly known prior to make probabilistic assessments of what the others will report.

So, what makes for a good mechanism? Typically, there is an *objective function* that the designer wants to maximize. One common objective is social welfare (the sum of the agents' utilities with respect to their *true*, not reported, types), but there are many others—for example, the designer may wish to maximize revenue (the sum of the agents' payments). However, there are certain constraints on what the designer can do. For example, it would not be reasonable for the designer to specify that a losing bidder in an auction should pay the designer a large sum: if so, the bidder would simply not participate in the auction. We next present constraints, called *participation* or *individual rationality (IR)* constraints, that prevent this. Before we do so, we note that we will assume *quasilinear preferences* when payments are involved.

Definition 17 *An agent i has* quasilinear preferences *if the agent's utility function can be written as $u_i(\theta_i, o) - \pi_i$.*

We are now ready to present the IR constraints.

Definition 18 Individual rationality (IR) *is defined as follows.*

- *A deterministic mechanism is* ex post IR *if for any agent i, and any type vector $(\theta_1, \ldots, \theta_n) \in \Theta_1 \times \ldots \times \Theta_n$, we have $u_i(\theta_i, o(\theta_1, \ldots, \theta_n)) - \pi_i(\theta_1, \ldots, \theta_n) \geq 0$.*

  A randomized mechanism is ex post IR *if for any agent i, and any type vector $(\theta_1, \ldots, \theta_n) \in \Theta_1 \times \ldots \times \Theta_n$, we have $E_{o|\theta_1,..,\theta_n}[u_i(\theta_i, o) - \pi_i(\theta_1, .., \theta_n)] \geq 0$.*

- *A deterministic mechanism is* ex interim IR *if for any agent i, and any type $\theta_i \in \Theta_i$, we have $E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_n)|\theta_i}[u_i(\theta_i, o(\theta_1, .., \theta_n)) - \pi_i(\theta_1, .., \theta_n)] \geq 0$.*

  A randomized mechanism is ex interim IR *if for any agent i, and any type $\theta_i \in \Theta_i$, we have $E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_n)|\theta_i} E_{o|\theta_1,..,\theta_n}[u_i(\theta_i, o) - \pi_i(\theta_1, .., \theta_n)] \geq 0$.*

The terms involving payments are left out if payments are not possible.

Thus, participating in an *ex post* individually rational mechanism never makes an agent worse off; participating in an *ex interim* individually rational mechanism may make an agent worse off in the end, but not in expectation (assuming that the agent's belief over the other agents' reported types matches the common prior).

Still, as long as these are the only constraints, all that the designer needs to do is solve the outcome optimization problem (perhaps charging the agents' their entire utility as payment, in case revenue maximization is the objective). But we have not yet considered the agents' *incentives*. Agents will only report their preferences truthfully if they have an incentive to do so. We will impose *incentive compatibility (IC)* constraints that ensure that this is indeed the case. Again, there is an *ex post* and an *ex interim* variant; in this context, these variants are usually called *dominant-strategies incentive compatible* and *Bayes-Nash equilibrium (BNE) incentive compatible*, respectively. Given the (potential) difference between true and reported types, we will use the standard notation $\hat{\theta}_i$ to refer to agent i's reported type.

Definition 19 *A mechanism is* dominant-strategies incentive compatible *(or* strategy-proof*) if telling the truth is always optimal, even when the types reported by the other agents are already known. Formally, for any agent i, any type vector $(\theta_1, \ldots, \theta_i, \ldots, \theta_n) \in \Theta_1 \times \ldots \times \Theta_i \times \ldots \times \Theta_n$, and any alternative type report $\hat{\theta}_i \in \Theta_i$, in the case of deterministic mechanisms we require $u_i(\theta_i, o(\theta_1, \ldots, \theta_i, \ldots, \theta_n)) - \pi_i(\theta_1, \ldots, \theta_i, \ldots, \theta_n) \geq u_i(\theta_i, o(\theta_1, \ldots, \hat{\theta}_i, \ldots, \theta_n)) - \pi_i(\theta_1, \ldots, \hat{\theta}_i, \ldots, \theta_n)$. In the case of randomized mechanisms we have $E_{o|\theta_1,..,\theta_i,..,\theta_n}[u_i(\theta_i, o) - \pi_i(\theta_1, \ldots, \theta_i, \ldots, \theta_n)] \geq E_{o|\theta_1,..,\hat{\theta}_i,..,\theta_n}[u_i(\theta_i, o) - \pi_i(\theta_1, \ldots, \hat{\theta}_i, \ldots, \theta_n)]$.*
The terms involving payments are left out if payments are not possible.

Definition 20 *A mechanism is* Bayes-Nash equilibrium (BNE) incentive compatible *if telling the truth is always optimal to an agent when that agent does not yet know anything about the other* *agents' types, and the other agents are telling the truth. Formally, for any agent* i*, any type* $\theta_i \in \Theta_i$*, and any alternative type report* $\hat{\theta}_i \in \Theta_i$*, in the case of deterministic mechanisms we have* $E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_n)|\theta_i}[u_i(\theta_i, o(\theta_1,\ldots,\theta_i,\ldots,\theta_n)) - \pi_i(\theta_1,\ldots,\theta_i,\ldots,\theta_n)] \geq E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_n)|\theta_i}[u_i(\theta_i, o(\theta_1,\ldots,\hat{\theta}_i,\ldots,\theta_n)) - \pi_i(\theta_1,\ldots,\hat{\theta}_i,\ldots,\theta_n)]$*. In the case of randomized mechanisms we have* $E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_n)|\theta_i} E_{o|\theta_1,..,\theta_i,..,\theta_n}[u_i(\theta_i, o) - \pi_i(\theta_1,\ldots,\theta_i,\ldots,\theta_n)] \geq E_{(\theta_1,..,\theta_{i-1},\theta_{i+1},..,\theta_n)|\theta_i} E_{o|\theta_1,..,\hat{\theta}_i,..,\theta_n}[u_i(\theta_i, o) - \pi_i(\theta_1,\ldots,\hat{\theta}_i,\ldots,\theta_n)]$*.*

The terms involving payments are left out if payments are not possible.

One may wonder whether it is possible to obtain better outcomes by using a direct-revelation mechanism that is not truthful—perhaps the cost of the resulting strategic misreporting is not as great as the cost of having to honor the incentive compatibility constraints. Or, perhaps we could do even better using a mechanism that is not a direct-revelation mechanism—that is, a mechanism under which agents have other actions to take besides merely reporting their preferences. A famous result called the *revelation principle* [Gibbard, 1973; Green and Laffont, 1977; Myerson, 1979, 1981] shows that, when agents are perfectly strategic (unboundedly rational), the answer to both of these questions is "no": there is no loss in restricting attention to truthful, direct-revelation mechanisms.[1] For now, we do not yet have a definition of strategic behavior in non-truthful or indirect mechanisms, so we will postpone detailed discussion of the revelation principle to Chapter 7 (and we will question the assumption of unbounded rationality in Chapters 8 and 9). However, the intuition behind the revelation principle is simple: suppose we envelop a non-truthful mechanism with an *interface layer*, to which agents input their preferences. Then, the interface layer interacts with the original mechanism on behalf of each agent, *playing strategically in the agent's best interest* based on the reported preferences. (Compare, for example, proxy agents on eBay [eBay UK, 2004].) The resulting mechanism is truthful: an agent has no incentive to misreport to the interface layer, because the layer will play the agent's part in the original mechanism in the agent's best interest. Moreover, the final outcome of the new, truthful mechanism will be the same, because the layer will play strategically optimally—just as the agent would have.

In the next section, we will define the famous Vickrey-Clarke-Groves mechanisms.

4.2 Vickrey-Clarke-Groves mechanisms

The most straightforward direct-revelation mechanism for selling a single item is the *first-price sealed-bid* auction, in which each bidder submits a bid for the item in (say) a sealed envelope, and the highest bidder wins and pays the value that he bid. This is certainly not an incentive-compatible mechanism: in fact, bidding one's true valuation guarantees a utility of 0 (even if the bid wins, the bidder will pay his entire valuation). Rather, to obtain positive utility, a bidder needs to reduce (or *shave*) his bid, ideally to the point where it is only slightly higher than the next highest bid. Another direct-revelation mechanism is the *Vickrey* [Vickrey, 1961] or *second-price sealed-bid* auction, in which the highest bidder still wins, but pays the value of the *second* highest bid. The Vickrey auction is strategy-proof. To see why, imagine a bidder that knows the other bids. This bidder has only two

[1] The result requires that we can use randomized truthful mechanisms. Moreover, if there are multiple strategic equilibria in the original non-truthful mechanism, then we can choose any one of them to be preserved in the truthful mechanism, but not *all* the equilibria are necessarily preserved.

choices: bid higher than the highest other bid, to win and pay the value of that other bid; or bid lower, and do not win the item. The bidder will prefer to do the former if his valuation is higher than the highest other bid, and the latter otherwise. But in fact, bidding truthfully accomplishes exactly this! Hence, bidding truthfully guarantees one the same utility that an omniscient bidder would receive, and therefore the mechanism is strategy-proof.

It turns out that the Vickrey mechanism is a special case of a general mechanism called the *Clarke* mechanism (or Clarke tax) [Clarke, 1971], which can be applied to combinatorial auctions and exchanges, as well as other preference aggregation settings. The Clarke mechanism works as follows. First, choose the optimal outcome based on the bidders' reported preferences; call this outcome o^*. Then, to determine agent i's payment, remove agent i from the preference aggregation problem, and solve this problem again to obtain o^*_{-i}. Agent i will be required to pay $\sum\limits_{j \neq i} u_j(\hat{\theta}_j, o^*_{-i}) - \sum\limits_{j \neq i} u_j(\hat{\theta}_j, o^*)$. Informally, agent i's payment is exactly the amount by which the other agents are worse off due to agent i's presence—the *externality* that i imposes on the other agents. The Clarke mechanism is strategy-proof, for the following reason. Agent i seeks to maximize $u_i(\theta_i, o^*) + \sum\limits_{j \neq i} u_j(\hat{\theta}_j, o^*) - \sum\limits_{j \neq i} u_j(\hat{\theta}_j, o^*_{-i})$. Since o^*_{-i} does not depend on agent i's report, agent i cannot affect the term $\sum\limits_{j \neq i} u_j(\hat{\theta}_j, o^*_{-i})$, so equivalently, agent i seeks to maximize $u_i(\theta_i, o^*) + \sum\limits_{j \neq i} u_j(\hat{\theta}_j, o^*)$. Agent i can only affect this expression by influencing the choice of o^*, and the mechanism will select o^* to maximize $\sum\limits_{j=1}^{n} u_j(\hat{\theta}_j, o^*)$. But then, if the agent reports truthfully, that is, $\hat{\theta}_i = \theta_i$, then the mechanism will choose o^* precisely to maximize $u_i(\theta_i, o^*) + \sum\limits_{j \neq i} u_j(\hat{\theta}_j, o^*)$, thereby maximizing agent i's utility.

The Clarke mechanism is also *ex post* individually rational, *if* 1) the presence of an agent never makes it impossible to choose some outcome that could have been chosen without that agent, and 2) no agent ever has a negative utility for an outcome that would be selected if that agent were not present. Note that if either 1) or 2) does not hold, then the Clarke mechanism may require a payment from an agent that receives a utility of 0 for the chosen outcome, and is therefore not individually rational. Both 1) and 2) will hold in the remainder of this dissertation.

Additionally, the Clarke mechanism is *weak budget balanced*, that is, the sum of the payments from the agents is always nonnegative, *if* the following condition holds: when an agent is removed from the system, the new optimal (welfare-maximizing) outcome is at least as good for the remaining agents as the optimal outcome before the first agent was removed. That is, if an agent leaves, that does not make the other agents worse off in terms of the chosen outcome (not considering payments). This condition does not hold in, for example, task allocation settings: if an agent leaves, the tasks allocated to that agent must be re-allocated to the other agents, who will therefore be worse off. Indeed, in task allocation settings, the agents must be compensated for taking on tasks, so we do not expect weak budget balance. Green and Laffont [1977] show that it is not possible to obtain *strong* budget balance—the sum of the payoffs always being zero—in addition to choosing optimal outcomes and having dominant-strategies incentive compatibility.

Finally, the Clarke mechanism is just one mechanism among the class of *Groves* mechanisms [Groves, 1973]. To introduce this class of mechanisms, we note that in the Clarke mechanism, agent

i's type report $\hat{\theta}_i$ does not affect the terms $\sum_{j \neq i} u_j(\hat{\theta}_j, o^*_{-i})$ in agent i's payment; short of colluding with the other agents, there is nothing that agent i can do about paying these terms. Hence, if we removed these terms from the payment function, the mechanism would still be strategy-proof. Moreover, *any* term that we add to agent i's payment *that does not depend on* $\hat{\theta}_i$ will not compromise strategy-proofness. The class of Groves mechanisms consists precisely of all mechanisms that can be obtained in this manner. Additionally, Groves mechanisms are in fact the *only* mechanisms that are efficient (*i.e.* the mechanism chooses the optimal outcome) and dominant-strategies incentive-compatible, given that there is no restriction on what the agents' types can be [Green and Laffont, 1977] or even only given that agents' type spaces are smoothly connected [Holmström, 1979]. It should be noted that the Clarke mechanism is often referred to as "the" VCG mechanism, and we will follow this convention.

In the next section, I survey some other positive results in mechanism design (without presenting them in full detail).

4.3 Other possibility results

Interestingly, in some settings, there are Groves mechanisms that require a smaller total payment from the agents than the Clarke mechanism, while maintaining individual rationality and never incurring a deficit. The idea here is to *redistribute* some of the Clarke surplus back to the agents. To maintain incentive compatibility, how much is redistributed to an agent cannot depend on that agent's type. Nevertheless, if the other agents' reported types are such that a certain amount of Clarke surplus will be obtained *regardless* of the given agent's report, then we can redistribute a share of that guaranteed surplus (most naturally, $1/n$) to the agent. For example, in a single-item auction, each agent receives $1/n$ of the second-highest bid among the other bids [Cavallo, 2006].

It turns out that if we are willing to use Bayes-Nash incentive compatibility rather than dominant-strategies incentive compatibility, then we can obtain (strong) budget balance, using the dAGVA [d'Aspremont and Gérard-Varet, 1979; Arrow, 1979] mechanism. This mechanism is similar to a Groves mechanism, except that, instead of being paid the sum of other agents' utilities according to their reported types, an agent is paid the *expected* sum of other agent's utilities given only the agent's own report. In addition, payment terms that do not depend on the agent's own report can be set in such a way as to obtain budget balance.

As noted before, maximizing social welfare is not always the objective. Another common objective is to maximize revenue. In the context of auctions, this is often referred to as the problem of designing an "optimal" auction. The Myerson auction [Myerson, 1981] is a general mechanism for maximizing the expected revenue of an auctioneer selling a single item. The Maskin-Riley auction [Maskin and Riley, 1989] generalizes this to the case of multiple units of the same item. Only very limited characterizations of revenue-maximizing combinatorial auctions (with more than one item) are known [Avery and Hendershott, 2000; Armstrong, 2000].

Another positive result exists in the context of voting: if preferences are single-peaked, then choosing the median voter's peak as the winner (as we did in Chapter 2) is a strategy-proof mechanism.

4.4 Impossibility results

In the previous sections, we saw mechanisms that achieve certain sets of desirable properties. In this section, we discuss a few negative results, that state that certain sets of desirable properties cannot be obtained by a single mechanism.

Possibly the best-known impossibility result in mechanism design is the Gibbard-Satterthwaite theorem [Gibbard, 1973; Satterthwaite, 1975]. This result shows a very strong impossibility in very general preference aggregation settings (voting settings). Specifically, it shows that when there are three or more possible outcomes (candidates), two or more agents (voters), and there is no restriction on the preferences that can be submitted (such as single-peakedness), then a (deterministic) mechanism (voting rule) cannot have the following properties simultaneously:

- For every outcome, there exist preference reports by the agents that will make this outcome win.
- The mechanism is non-dictatorial, that is, the rule does not simply always choose a single, fixed voter's most-preferred candidate.
- The mechanism is strategy-proof.

Gibbard [1977] later extended this impossibility result to encompass randomized voting rules as well: a randomized voting rule is strategy-proof only if it is a probability mixture of *unilateral* and *duple* rules. (A rule is unilateral if only one voter affects the outcome, and duple if only two candidates can win.) It is not difficult to see that this result implies the Gibbard-Satterthwaite impossibility result.

As we have seen in the previous section, this impossibility result does not apply in settings where the agents' preferences are more restricted—*e.g.* single-peaked, or quasilinear in settings where payments are possible (in which case VCG can be used). Nevertheless, impossibility results exist in these more restricted settings as well. For example, the Myerson-Satterthwaite impossibility theorem [Myerson and Satterthwaite, 1983] states that even in simple bilateral trade settings with quasilinear utility functions, where we have a single seller with a single item (and a privately held valuation for this item), and a single buyer who may procure the item (and has a privately held valuation for the item), it is impossible to have a mechanism that achieves the following properties simultaneously:

- efficiency (trade takes place if and only if the buyer's valuation for the item is greater than the seller's);
- budget-balance (money may flow between the buyer and the seller, but not from/to other places);
- Bayes-Nash incentive compatibility;
- ex-interim individual rationality.

We will show another, similar impossibility result in Chapter 5.

4.5 Summary

This chapter reviewed basic concepts and results from mechanism design. We first reviewed various types of mechanisms, as well as individual-rationality and incentive-compatibility concepts. We then reviewed Vickrey-Clarke-Groves mechanisms and their properties in detail, and we briefly reviewed some other positive results (that is, mechanisms that achieve certain sets of properties), including Cavallo's redistribution mechanism, the dAGVA mechanism, Myerson and Maskin-Riley auctions, and single-peaked preferences. Finally, we briefly reviewed the Gibbard-Satterthwaite and Myerson-Satterthwaite impossibility results.

Armed with a basic understanding of mechanism design, we are now ready to move on to deeper levels of the hierarchy. The next chapter studies difficulties for classical mechanism design in expressive preference aggregation settings.

Chapter 5

Difficulties for Classical Mechanism Design

The classical study of mechanism design has not directly concerned itself with the more computational questions of the process, such as the representations of the outcome and preference spaces, or the complexity of choosing the optimal outcome. Some of the computational implications of the classical mechanisms are clear and direct. For example, to execute the VCG mechanism, we typically need to solve the optimization problem instance with all agents included, as well as, for each agent, the instance where that agent is removed. (However, in some cases, the structure of the domain can be used to solve all these instances simultaneously with an asymptotic time complexity that is the same as that of a single optimization [Hershberger and Suri, 2001].) But the issues run deeper than that. When the setting is complex and computation is limited, the optimizations must be approximated. This effectively results in a different mechanism, which may no longer be truthful—and its strategic equilibria (*e.g.*, Nash equilibria) may be terrible even when the approximation algorithm *per se* is very good. The resulting challenge is to design special approximation algorithms that do motivate the agents to report their preferences truthfully. Viewed differently, the challenge is to design special truthful mechanisms whose outcomes are at least reasonably good, and can be computed efficiently. This line of research, which has been called *algorithmic mechanism design* [Nisan and Ronen, 2001], has produced a number of interesting results [Nisan and Ronen, 2001, 2000; Feigenbaum *et al.*, 2001; Lehmann *et al.*, 2002; Mu'alem and Nisan, 2002; Archer *et al.*, 2003; Bartal *et al.*, 2003].

There have been various other directions in the study of mechanism design from a computer science perspective. One direction close to algorithmic mechanism design is the design of *anytime* mechanisms, which produce better outcomes as they are given more time to compute, but nevertheless maintain good incentive properties [Parkes and Schoenebeck, 2004]. Another goal that has been pursued is *distributing* the mechanism's computation across the agents [Parkes and Shneidman, 2004; Brandt and Sandholm, 2004b,a, 2005b,c,a; Izmalkov *et al.*, 2005; Petcu *et al.*, 2006]. A different direction is the design of mechanisms in task-allocation settings where the agents may fail to accomplish the task, and the failure probabilities need to be elicited from the agents, as well as the costs [Porter *et al.*, 2002; Dash *et al.*, 2004]. Rather than specifying a complete mechanism, another approach that has been considered is to provide the agents with limited data from a centralized optimization, and let the agents work out the remainder of the transactions. Specifically, the optimal allocation is given, as well as some bounds on reasonable prices, in such a way that the agents do not have an incentive to misreport [Bartal *et al.*, 2004]. (This has the advantage of circumventing, or at least delaying until later, results such as the Myerson-Satterthwaite impossibility theorem mentioned previously.)

This chapter provides some results on mechanism design in expressive preference aggregation settings. Unlike some the results mentioned above, these results are not inherently computational: rather, they are pure mechanism design results that are driven by the expressive nature of the preference aggregation problems under study. (Of course, computational advances are what has made running mechanisms in such expressive domains possible.) In Section 5.1, we study two related vulnerabilities of the VCG (Clarke) mechanism in combinatorial auctions and exchanges: low revenue/high cost, and collusion. Specifically, it will show how much worse these vulnerabilities are in these settings than in single-item settings [Conitzer and Sandholm, 2006d]. In Section 5.2, we study mechanism design for expressive preference aggregation for donations to (charitable) causes [Conitzer and Sandholm, 2004e].

5.1 VCG failures in combinatorial auctions and exchanges

The VCG mechanism is the canonical payment scheme for motivating the bidders to bid truthfully in combinatorial auctions and exchanges; if the setting is general enough, under some requirements, it is the only one [Green and Laffont, 1977; Lavi *et al.*, 2003; Yokoo, 2003]. Unfortunately, there are also many problems with the VCG mechanism [Rothkopf *et al.*, 1990; Sandholm, 2000; Ausubel and Milgrom, 2006]. In this section, we discuss two related problems: the VCG mechanism is vulnerable to collusion, and may lead to low revenue/high payment for the auctioneer. It is well-known that these problems occur even in single-item auctions (where the VCG mechanism specializes to the Vickrey or second-price sealed-bid auction). However, in the single-item setting, these problems are not as severe. For example, in a Vickrey auction, it is not possible for colluders to obtain the item at a price less than the bid of any other bidder. Additionally, in a Vickrey auction, various types of revenue equivalence with (for example) first-price sealed-bid auctions hold. As we will show, in the multi-item setting these properties do not hold and can be violated to an arbitrary extent. Some isolated examples of such problems with the VCG mechanism in multi-item settings have already been noted in the literature [Ausubel and Milgrom, 2006; Yokoo *et al.*, 2004; Archer and Tardos, 2002] (these will be discussed later in the section). In contrast, our goal in this section is to give a *comprehensive* characterization of how severe these problems can be and when these severe problems can occur. For the various variants of combinatorial auctions and exchanges, we study the following single problem that relates both issues under consideration: *Given some of the bids, how bad can the remaining bidders make the outcome?* Informally, "bad" here means that the remaining bidders are paid an inordinately large amount, or pay an inordinately small amount, relative to the goods they receive and/or provide. This is closely related to the problem of making revenue guarantees to the auctioneer. But it is also the collusion problem, if we conceive of the remaining bidders as colluders. (The collusion problem can become more difficult if the collusion is required to be *self-enforcing*. A collusion is self-enforcing when none of the colluders have an incentive to unilaterally deviate from the collusion. We will also study how this extra requirement affects our results.)

As it turns out, the fundamental problem of deciding how bad the remaining bidders can make the outcome is often computationally hard. Computational hardness here is a double-edged sword. On the one hand, if the problem is hard, collusion may not occur (or to a lesser extent) because the colluders cannot find a beneficial collusion. On the other hand, if the problem is hard, it is difficult to make strong revenue guarantees to the auctioneer. Of course, in either case, the computational hardness may be overcome in practice if the stakes are high enough.

All the results in this section hold even when all bidders are *single-minded*, that is, they bid only on a single bundle of items. Hence, we do not need to discuss bidding languages.

5.1.1 Combinatorial (forward) auctions

We recall that in a *combinatorial auction*, there is a set of items $I = \{s_1, s_2, \dots, s_m\}$ for sale. A bid takes the form $b = (B, v)$, where $B \subseteq I$ and $v \in \mathbb{R}$. The winner determination problem is to label bids as accepted or rejected, to maximize the sum of the values of the accepted bids, under the constraint that no item occurs in more than one accepted bid. (This is assuming *free disposal*: items do not have to be allocated to anyone.)

Motivating example

(A similar example to the one described in this subsubsection has been given before [Ausubel and Milgrom, 2006], and examples of vulnerability to false-name bidding in combinatorial auctions [Yokoo *et al.*, 2004] can in fact also be used to demonstrate the basic point. We include this subsubsection for completeness.) Consider an auction with two items, s_1 and s_2. Suppose we have collected two bids (from different bidders), both $(\{s_1, s_2\}, N)$. If these are the only two bids, one of the bidders will be awarded both the items and, under the VCG mechanism, will have to pay N. However, suppose two more bids (by different bidders) come in: $(\{s_1\}, N+1)$ and $(\{s_2\}, N+1)$. Then these bids will win. Moreover, neither winning bidder will have to pay anything! (This is because a winning bidder's item would simply be thrown away if that winning bidder were removed.)

This example demonstrates a number of issues. First, the addition of more bidders can actually decrease the auctioneer's revenue from an arbitrary amount to 0. Second, the VCG mechanism is not revenue-equivalent to the sealed-bid first-price mechanism in combinatorial auctions, even when all bidders' true valuations are common knowledge[1]—unlike in the single-item case. Third, even when the other bidders by themselves would generate nonnegative revenue for the auctioneer under the VCG mechanism, it is possible that two colluders can bid so as to receive all the items without paying anything.

[1]Consider the above example with $N \geq 9$ and suppose that the four bids reflect the bidders' true valuations—since bidding truthfully is a weakly dominant strategy in the VCG mechanism. Running a first-price sealed bid auction in this setting, when all bidders' valuations are common knowledge, will not generate expected revenue less than $\frac{N}{8}$. For suppose the expected revenue is less than this. Then the probability that the revenue is at least $\frac{N}{4}$ must be less than $\frac{1}{2}$ by Markov's inequality. So, bidding $(\{A, B\}, \frac{N}{4})$ will win any bidder both items with probability at least $\frac{1}{2}$, leading to an expected utility of at least $\frac{1}{2}(N - \frac{N}{4}) = \frac{3N}{8}$. Because at most one of the three bidders with valuations $(\{A\}, N+1)$ or $(\{A, B\}, N)$ can win its desired bundle, it follows that at least one of these bidders has a probability of at most $\frac{1}{3}$ of winning its desired bundle, and thus has an expected utility of at most $\frac{N+1}{3}$. Because $N \geq 9$, $\frac{3N}{8} > \frac{N+1}{3}$, so this bidder would be better off bidding $(\{A, B\}, \frac{N}{4})$—contradicting the assumption that we are in equilibrium.

The following sums up the properties of this example.

Proposition 5 *In a forward auction (even with only 2 items), the following can hold simultaneously: 1. The winning bidders pay nothing under the VCG mechanism; 2. If the winning bids are removed, the remaining bids generate revenue N under the VCG mechanism; 3. If these bids were truthful (as we would expect under VCG), then if we had run a first-price sealed-bid auction instead (and the bidders' valuations were common knowledge), any equilibrium would have generated revenue* $\Theta(N)$.

Characterization

We now characterize the settings where, given the noncolluders' bids, the colluders can receive all the items for free.

Lemma 9 *If the colluders receive all the items at cost 0, then for any positive bid on a bundle B of items by a noncolluder, at least two of the colluders receive an item from B.*

Proof: Suppose that for some positive bid b on a bundle B by a noncolluder i, one of the colluders c receives all the items in B (and possibly others). Then, in the auction where we remove that colluder's bids, one possible allocation gives every remaining bidder all the goods that bidder received in the original auction; additionally, it gives i all the items in bundle B; and it disposes of all the other items c received in the original auction. With this allocation, the total value of the accepted bids by bidders other than c is at least $v(b)$ more than in the original auction. Because the total value obtained in the new auction is at least the value of this particular allocation, it follows that c imposes a negative externality of at least $v(b)$ on the other bidders, and will pay at least $v(b)$. But this contradicts the fact that no colluder pays anything; and hence it follows that for any positive bid b on a bundle B by a noncolluder i, at least two of the colluders receive an item from B. ■

Lemma 10 *Suppose all the items in the auction can be divided among the colluders in such a way that, for any positive bid on a bundle of items B by a noncolluder, at least two of the colluders receive an item from B. Then the colluders can receive all the items at cost 0.*

Proof: For the given partition of items among the noncolluders, let each colluder place a bid with an extremely large value on the bundle consisting of the items assigned to him in the partition. (For instance, twice the sum of the values of all noncolluders' bids.) Then, the auction will clear awarding each colluder the items assigned to him by the partition. Moreover, if we remove the bids of one of the colluders, all the remaining colluders' bids will still win—and thus none of the noncollu ders' bids will win, because each such bid requires items assigned to at least two colluders by the partition (and at least one of them is still in the auction and wins th ese items). Thus, each colluder (individually) imposes no externality on the other bidders. ■

Combining these two lemmas, we get:

Theorem 24 *The colluders can receive all the items at cost* 0 *if and only if it is possible to divide the items among the colluders in such a way that, for any positive bid B by a noncolluder, at least two colluders receive an item from B.*

Self-enforcing collusion

It turns out that requiring that the collusion is self-enforcing (i.e., no colluder has an incentive to unilaterally deviate) is no harder for the colluders:

Theorem 25 *Whenever the colluders can receive all the items for free, they can also receive them all for free in a self-enforcing way.*

Proof: Let each colluder bid on the same bundle as before; but, increase the bid value of each colluder by an amount that exceeds the utility that any colluder can get from any bundle of items. The colluders will continue to receive all the items at a cost of 0. Now, the only reason that a colluder may wish to deviate from this is that the colluder wishes to obtain items outside of the colluder's assigned bundle. However, doing so would prevent one of the other bundles from being awarded to its designated colluder. This would cause a decrease in the total value of bids awarded to bidders other than the deviating colluder that exceeds the utility of the deviating colluder for any bundle, and the deviating colluder would have to pay for this decrease under the VCG mechanism. Therefore, there is no incentive for the colluder to deviate. ■

Complexity

In order to collude in the manner described above, the n colluders must solve the following computational problem.

Definition 21 (DIVIDE-SUBSETS) *Suppose we are given a set I, as well as a collection $R = \{S_1, \ldots, S_q\}$ of subsets of it. We are asked whether I can be partitioned into n parts $T_1, T_2, \ldots, T_n$ so that no subset $S_i \in R$ is contained in one of these parts.*

Theorem 26 *DIVIDE-SUBSETS is NP-complete, even when $n = 2$.*

Proof: The problem is technically identical to HYPERGRAPH-2-COLORABILITY, which is NP-complete [Garey and Johnson, 1979]. ■

This hardness result only states that it is hard to identify the *most* beneficial collusion, and one may wonder whether it is perhaps easier to find *some* beneficial collusion. It turns out that the hardness of the former problem implies the hardness of the latter problem: the utility functions of the colluders can always be such that only the most beneficial collusion actually benefits them, in which case the two problems are the same. This observation can also be applied to hardness results presented later in this section.

5.1.2 Combinatorial reverse auctions

We recall that in a *combinatorial reverse auction*, there is a set of items $I = \{s_1, s_2, \ldots, s_m\}$ to be procured. A bid takes the form $b = (B, v)$, where $B \subseteq I$ and $v \in \mathbb{R}$. (Here, v represents the value that the bidder must be compensated by in order to provide the goods B.) The winner determination problem is to label bids as accepted or rejected, to minimize the sum of the values of the accepted bids, under the constraint that each item occurs in at least one accepted bid. (This is assuming free disposal.)

Motivating example

Consider a reverse auction with m items, $s_1, s_2, \ldots, s_m$. Suppose we have collected two bids (from different bidders), both $(\{s_1, s_2, \ldots, s_m\}, N)$. If these are the only two bids, one of the bidders will be chosen to provide all the goods, and, under the VCG mechanism, will be paid N. However, suppose m more bids (by different bidders) come in: $(\{s_1\}, 0), (\{s_2\}, 0), \ldots, (\{s_m\}, 0)$. Then, these m bids will win. Moreover, each bidder will be paid N under the VCG mechanism. (This is because without this bidder, we would have had to accept one of the original bids.) Thus, the total payment that needs to be made is mN.[2]

Again, this example demonstrates a number of issues. First, the addition of more bidders may actually increase the total amount that the auctioneer needs to pay. Second, the VCG mechanism requires much larger payments than a first-price auction in the case where all bidders' valuations are common knowledge. (The first-price mechanism will not require a total payment of more than N for these valuations in any pure-strategy equilibrium.[3]) Third, even when the other bidders by themselves would allow the auctioneer to procure the items at a low cost under the VCG mechanism, it is possible for m colluders to get paid m times as much for all the items.

The following sums up the properties of this example.

Proposition 6 *In a reverse auction, the following can hold simultaneously: 1. The winning bidders are paid mN under the VCG mechanism; 2. If the winning bids are removed, the remaining bids allow the auctioneer to procure everything at a cost of only N under the VCG mechanism; 3. If these bids were truthful (as we would expect under VCG), then if we had run a first-price sealed-bid reverse auction instead (and the bidders' valuations were common knowledge), any equilibrium in pure strategies would have required total payment of at most N. (However, there are also mixed-strategy equilibria with arbitrarily large expected tot al payment.)*

---

[2]Similar examples have been discovered in the context of purchasing paths in a graph [Archer and Tardos, 2002]. However, in that setting, the buyer does not seek to procure all of the items, and hence the examples cannot be applied directly to combinatorial reverse auctions.

[3]Consider the above example and suppose that the $n + 2$ bids reflect the bidders' true valuations—since bidding truthfully is a weakly dominant strategy in the VCG mechanism. Supposing that a pure-strategy equilibrium is being played, let the total payment to be made in this equilibrium be π. (We observe that the final allocation can still be uncertain, e.g. if there is a random tie-breaking rule.) Suppose $\pi > N$. Then, the expected utility for either one of the bidders interested in providing the whole bundle can never exceed $\pi - N$ (because the bidder will be paid 0 whenever none of its bids are accepted, and providing any items at all will cost it N). Moreover, it is not possible for both of these bidders to simultaneously have an expected utility of $\pi - N$ (as this would mean that both are paid π with certainty, contrary to the fact that the total payment is π). It follows at least one has an expected utility of $\pi - N - \epsilon$ for some $\epsilon > 0$. But then this bidder would be better off bidding $\pi - \frac{\epsilon}{2}$ for the whole bundle, which would be accepted with certainty and give an expected utility of $\pi - N - \frac{\epsilon}{2}$. It follows that the total payment in a pure-strategy equilibrium cannot exceed N. Perhaps surprisingly, the first-price combinatorial reverse auction for this example (with commonly known true valuations corresponding to the given bids) actually has mixed-strategy equilibria with arbitrarily high expected payments.

Characterization

Letting N be the sum of the values of the accepted bids when all the colluders' bids are taken out,[4] it is clear that no colluder can be paid more than N. (With the colluder's bid, the sum of the values of others' accepted bids is still at least 0; without it, it can be at most N, because in the worst case the auctioneer can accept the bids that would be accepted if none of the colluders are present.) In this subsubsection, we will identify a necessary and sufficient condition for the colluders to be able to each receive N.

Lemma 11 *If a colluder receives N, then the items that it has to provide cannot be covered by a subset of the noncolluders' bids with cost less than N.*

Proof: If they could be covered by such a set, we could simply accept this set of bids (including those that were accepted already) rather than the colluder's bid, and increase the total cost by less than N. Thus, the colluder's VCG payment is less than N. ∎

Thus, in order for each of the n colluders to be able to receive N, it is necessary that there exist n disjoint subsets of the items, each of which cannot be covered with a subset of the noncolluders' bids with total value less than N. The next lemma shows that this condition is also sufficient.

Lemma 12 *If there are n disjoint sets of items $R_1, \ldots, R_n$, each of which cannot be covered by a subset of the noncolluders' bids with cost less than N, then n colluders can be paid N each.*

Proof: Let colluder i (for $i < n$) bid $(R_i, 0)$, and let colluder n bid $(R_n \cup (S - \bigcup_i R_i), 0)$. Then the total cost of all accepted bids with all the colluders is 0; but when one colluder is omitted, the items it won cannot be covered at a cost less than N (because its bid contained one of the R_i). Thus, each colluder's VCG payment is N. ∎

The next lemma shows that the necessary and sufficient condition above is equivalent to being able to *partition* all the items into n sets, so that no element of the partition can be covered by a subset of the noncolluders' bids with total value less than N. That is, we can restrict our attention to the case where the subsets exhaust all the items.

Lemma 13 *The condition of Lemma 12 is satisfied if and only if it is possible to partition the items into $T_1, \ldots, T_n$ such that no T_i can be covered by a subset of the noncolluders' bids with cost less than N.*

Proof: The "if" part is trivial: given T_i that satisfy the condition of this lemma, simply let $R_i = T_i$. For the "only if" part, given R_i that satisfy the condition of Lemma 12, let $T_i = R_i$ for $i < n$, and $T_n = R_n \cup (S - \bigcup_i R_i)$. We observe that this last set can also not be covered at a cost of less than N because it contains R_n. ∎

Combining all the lemmas, we get:

[4] We assume, as is commonly done in settings such as these, that a feasible solution still exists when all the colluders' bids are removed.

Theorem 27 *The n colluders can receive a payment of N each (simultaneously), where N is the sum of the values of the accepted bids when all the colluders' bids are removed, if and only if it is possible to partition the items into $T_1, \ldots, T_n$ such that no T_i can be covered by a subset of the noncolluders' bids with cost less than N.*

Self-enforcing collusion

Unlike the case of combinatorial forward auctions, in reverse auctions, a stronger condition is required if the collusion is also required to be self-enforcing.

Theorem 28 *The n colluders can receive a payment of N each (simultaneously), where N is the sum of the values of the accepted bids when all the colluders' bids are removed, if and only if it is possible to partition the items into $T_1, \ldots, T_n$ such that 1) no T_i can be covered by a subset of the noncolluders' bids with cost less than N, 2) for no colluder i, the following holds: there exists a subset $T_i' \subseteq T_i$ such that T_i' can be covered by a set of noncolluders' bids with total cost less than $v_i(T_i) - v_i(T_i - T_i')$ (the marginal savings to colluder i of not having to provide T_i').*

Proof: For the "if" part, each colluder i can bid on T_i with a value of 0. As in the above, this will give each colluder a payment of N. Moreover, no colluder i has an incentive to deviate, for the following reasons. Under the VCG mechanism, it is not possible to change a bidder i's bid in such a way that the allocation to i remains the same, but the payment to i changes. Therefore, we only need to consider what happens if colluder i bids on a different bundle. Bidding on items outside T_i cannot increase the payment to i because the other colluders are bidding on these items with a value of 0. Therefore, the only deviation that can possibly be advantageous is to bid on a subset T_i'' of T_i. Let $T_i' = T_i - T_i''$. If the colluder bids on T_i'' (with, say, value 0), then the payment to colluder i will decrease by the total cost of covering T_i' with noncolluder bids. By the assumption in the theorem, this total cost is at least $v_i(T_i) - v_i(T_i'')$, the marginal savings to colluder i of not having to provide T_i'. It follows that the bid does not make the colluder better off.

For the "only if" part, we already know by Theorem 27 that in order for the n colluders to receive a payment of N each (simultaneously), it must be possible to partition the items into $T_1, \ldots, T_n$ such that no T_i can be covered by a subset of the noncolluders' bids with cost less than N (so that colluder i can bid on T_i with a value of 0 to achieve the desired outcome). But if for some colluder i, there exists a subset $T_i' \subseteq T_i$ such that T_i' can be covered by a set of noncolluders' bids with total cost less than $v_i(T_i) - v_i(T_i - T_i')$, then this colluder would be better off bidding a value of 0 for $T_i - T_i'$ instead, because this would decrease the payment to colluder i by less than the marginal savings to colluder i of not having to provide T_i'. Hence the collusion would not be self-enforcing. ∎

Complexity

In order to collude in the manner described above, the n colluders must solve the following computational problem.

Definition 22 (CRITICAL-PARTITION) *We are given a set of items I, a collection of bids (S_i, v_i) where $S_i \subseteq I$ and $v_i \in \mathbb{R}$, and a number n. Say that the* cost *of a subset of these bids is the sum of* *their v_i; and that the cost $c(T)$ of a subset $T \subseteq I$ is the lowest cost of any subset of the bids whose S_i cover T. We are asked whether there exists a partition of I into n disjoint subsets $T_1, T_2, \ldots, T_n$, such that for any $1 \leq i \leq n$, $c(T_i) = c(I)$.*

Theorem 29 *Even when the bids are so that a partition $T_1, \ldots, T_n$ is a solution if and only if no set $I - T_i$ covers all items in a bid, CRITICAL-PARTITION is NP-complete (even with $n = 2$).*

Proof: The problem is in NP in this case because given a partition $T_1, \ldots, T_n$, it is easy to check if any set $I - T_i$ covers all items in a bid.

To show NP-hardness, we reduce an arbitrary NAESAT[5] instance (given by a set of clauses C over a set of variables V, with each variable occurring at most once in any clause) to the following CRITICAL-PARTITION instance with $n = 2$ (where we are trying to partition into T_1 and T_2). Let I be as follows. For every variable $v \in V$, there are two items labeled s_{+v} and s_{-v}. Let the bids be as follows. For every variable $v \in V$, there is a bid $(\{s_{+v}, s_{-v}\}, 2)$. For every clause $c \in C$, there are two bids $(\{s_l : l \in c\}, 2m_c - 1)$ and $(\{s_l : -l \in c\}, 2m_c - 1)$ where m_c is the number of literals occurring in c.

First we show that this instance satisfies the condition that a partition $T_1, \ldots, T_n$ is a solution if and only if no set $I - T_i$ covers all items in a bid. First, we observe that $c(I) = |I|$ (we can use all the bids of the form $(\{s_{+v}, s_{-v}\}, 2)$, getting a per-item cost of 1; no other bid gives a lower per-item cost).

Now, if some set $I - T_i$ covers all the items in a bid of the form $(\{s_{+v}, s_{-v}\}, 2)$, then $c(T_i) \leq 2|I| - 2$ (because we can simply omit this bid from the solution for all the items). If some set $I - T_i$ covers all the items in a bid of the form $(\{s_l : l \in c\}, 2m_c - 1)$, then $c(T_i) = |I| - 1$. (This is because we can now accept the "complement" bid $(\{s_l : -l \in c\}, 2m_c - 1)$, and we will have covered all the items s_{+v} and s_{-v} in T_i such that v occurs in c (precisely $2m_c$ items, because variables do not reoccur within a clause); for any other item s_{+v} or s_{-v}, we can accept the bid $(\{s_{+v}, s_{-v}\}, 2)$, and we need to accept at most $|V| - m_c$ such bids, leading to a total cost of $2m_c - 1 + 2(|V| - m_c) = |I| - 1$.)

On the other hand, suppose there is no set $I - T_i$ that covers all the items in a bid. Then, either T_i must include at precisely one of s_v and s_{-v}. (Otherwise one T_i would include neither and $I - T_i$ would cover all items in the bid $(\{s_{+v}, s_{-v}\}, 2)$.) Thus, when we are trying to cover T_i, covering items in it with bids of the form $(\{s_{+v}, s_{-v}\}, 2)$ would result in a per-item cost of 2. On the other hand, covering items in it with bids of the form $(\{s_l : l \in c\}, 2m_c - 1)$ or $(\{s_l : -l \in c\}, 2m_c - 1)$ would result in a per-item cost of at least $\frac{2m_c - 1}{m_c - 1} > 2$ (because at most $m_c - 1$ of the m_c items in the bid can be in T_i, otherwise T_i would cover all the items in the bid; but $T_i = I - T_{3-i}$ which by assumption does not cover all the items in any bid). It follows that $C(T_i) = 2|V| = |I| = c(I)$.

Now we show that the two instances are equivalent. First suppose there exists a solution to the NAESAT instance. Then partition the elements as $T_1 = \{s_l : l = true\}$ and $T_2 = \{s_l : l = false\}$, according to this solution. Clearly neither of $I - T_i = T_{3-i}$ covers a bid of the form $(\{s_{+v}, s_{-v}\}, 2)$. Also, because no clause has all its literals set to the same value (we have a NAESAT solution), the items in a corresponding bid $(\{s_l : l \in c\}, 2m_c - 1)$ or $(\{s_l : -l \in c\}, 2m_c - 1)$ are not all in

[5]The goal in NAESAT is to assign truth values to all variables in such a way that there is no clause with all its literals set to *true*, and no clause with all its literals set to *false*.

the same set. By the previously proved property, it follows that this partition is a solution to the CRITICAL-PARTITION instance.

On the other hand, suppose that there exists a solution to the CRITICAL-PARTITION instance. Then label a literal *true* if $s_l \in T_1$, and *false* otherwise. By the previously proved property, because $(\{s_{+v}, s_{-v}\}, 2)$ is a bid, only one of s_{+v} and s_{-v} can be in $T_1 = I - T_2$, so this provides a consistent setting of the literals. Additionally, because $(\{s_l : l \in c\}, 2m_c - 1)$ is a bid, not all the s_l in that bid can be in $T_1 = I - T_2$. It follows that some of the literals $l \in c$ are set to *false*. Similarly, not all the s_l in that bid can be in $T_2 = I - T_1$, so some of the literals $l \in c$ are set to *true*. It follows that this assignment of truth values to variables is a solution to the NAESAT instance. ■

5.1.3 Combinatorial forward (or reverse) auctions without free disposal

We recall that a combinatorial forward auction *without free disposal* is exactly the same as one with free disposal, with the exception that every item must be allocated to some bidder. Recall from Section 2.2 that since we are looking for an exact cover of the items, and negative bids may be of use, combinatorial forward auctions are technically identical to combinatorial reverse auctions.

Motivating example

Consider a forward auction with two nondisposable items, s_1 and s_2. Suppose we have collected two bids (from different bidders), both $(\{s_1, s_2\}, N)$. If these are the only two bids, one of the bidders will be awarded both the items and, under the VCG mechanism, will have to pay N. However, suppose two more bids (by different bidders) come in: $(\{s_1\}, N + M)$ and $(\{s_2\}, N + M)$, with $M > 0$. Then these bids will win. Moreover, because without free disposal, we cannot accept either of these bids without the other, each of these bidders will be *paid* M under the VCG mechanism!

Again, this example demonstrates a number of issues. First, additional bidders may change the auctioneer's revenue from an arbitrarily large positive amount to an arbitrarily large negative amount (an arbitrarily large cost). Second, the VCG mechanism may require arbitrarily large payments from the auctioneer even in cases where a first-price auction would actually generate revenue for the auctioneer, in the case where all bidders' valuations are common knowledge. (The first-price mechanism will generate a revenue of at least N for these valuations in any pure-strategy equilibrium.[6]) Third, even when the other bidders by themselves would generate positive revenue for the

[6]Consider the above example and suppose that the four bids reflect the bidders' true valuations—since bidding truthfully is a weakly dominant strategy in the VCG mechanism. Supposing that a pure-strategy equilibrium is being played, let the total revenue to the auctioneer be ρ, where ρ is possibly negative. (We observe that the final allocation can still be uncertain, e.g. if there is a random tie-breaking rule.) Suppose $\rho < N$. Then the expected utility for either of the bidders interested in providing the whole bundle is at most $N - \rho$. (If the bidder receives a singleton item, its utility is $-\infty$; if it receives nothing, its utility is 0; if it receives both items, its utility is $N - \rho$.) Moreover, it is not possible for both of these bidders to both have an expected utility of $N - \rho$, as this would mean they both receive both items with probability 1. It follows that at least one of them has an expected utility of $N - \rho - \epsilon$ where $\epsilon > 0$. But then this bidder would be better off bidding $\rho + \frac{\epsilon}{2}$, as this bid would be accepted with certainty and give an expected utility of $N - \rho - \frac{\epsilon}{2}$. It follows that the expected revenue in a pure-strategy equilibrium cannot be less than N. Similarly to the case of the combinatorial reverse auction with free disposal, there are mixed-strategy equilibria in the first-price auction where the auctioneer is forced to make arbitrarily large payments.

auctioneer under the VCG mechanism, it is possible that two colluders can make the auctioneer pay each of them an arbitrarily large amount.

The following sums up the properties of this example.

Proposition 7 *In a forward auction without free disposal (even with only two items), the following can hold simultaneously: 1. Each winning bidder is paid an arbitrary amount M under the VCG mechanism (where M depends only on the winners' bids); 2. If the winning bids are removed, the remaining bids actually generate revenue N to the auctioneer under the VCG mechanism; 3. If these bids were truthful (as we would expect under VCG), then if we had run a first-price sealed-bid auction instead (and the bidders' valuations were common knowledge), any equilibrium in pure strategies would have generated revenue N. (However, there are mixed-strategy equilibria with arbitrarily large cost to the auctioneer.)*

Characterization

In this subsubsection, we will identify a necessary and sufficient condition for the colluders to be able to each receive an arbitrary amount. Let $v(b)$ denote the value of bid b.

Lemma 14 *If each colluder receives a payment of more than $2\sum_d |v(b_d)|$ (where d ranges over the noncolluders), then for each colluder c, the set of all items awarded to either that colluder or a noncolluder (that is, $s_c \cup \bigcup_d s_d$, where s_b is the set of items awarded to bidder b and d ranges over the noncolluders) cannot be covered exactly with bids from the noncolluders.*

Proof: Say that the sum of the values of accepted noncolluder bids is D (which may be negative). Suppose that for one colluder c, the set of all items awarded to either her or a noncolluder (that is, $s_c \cup \bigcup_d s_d$) can be covered by a set of noncolluder bids of combined value C (which may be negative). Then removing colluder c can make the allocation at most $D - C$ worse to the other bidders (relative to their reported valuations), because we could simply accept the bids of combined value C and no longer accept the bids of combined value D, and keep the rest of the allocation the same. Thus, under VCG, that colluder should be rewarded at most $D - C \leq 2\sum_d |v(b_d)|$. ■

Thus, in order for each colluder to be able to receive an arbitrarily large payment, it is necessary that there are n disjoint subsets of the items such that no such subset taken together with the remaining items can be covered exactly by the noncolluders' bids. Also, the set of remaining items must be exactly coverable by the noncolluders' bids (otherwise we cannot accept all the colluders' bids). The next lemma shows that this condition is also sufficient.

Lemma 15 *If it is possible to partition the items into $R_1, \ldots, R_n, R_{n+1}$ such that for no $1 \leq i \leq n$, $R_i \cup R_{n+1}$ can be covered exactly with bids from the noncolluders; and such that R_{n+1} can be covered exactly with bids from the noncolluders; then for any $M > 0$, n colluders can place additional bids such that each of them receives at least M.*

Proof: Let colluder i place a bid $(R_i, M + 3\sum_d |v(b_d)|)$ (where d ranges over the noncolluders). All these bids will be accepted, because it is possible to do so by also accepting the noncolluder bids that cover R_{n+1} exactly; and these noncolluder bids will have a combined value of at least $-\sum_d |v(b_d)|$, so that the sum of the values of all accepted bids is at least $(3n-1)\sum_d |v(b_d)| + nM$. (We observe that if we do not accept all of the colluder bids, the sum of the values of all accepted bids is at most $(3(n-1)+1)\sum_d |v(b_d)| + (n-1)M = (3n-2)\sum_d |v(b_d)| + (n-1)M$, which is less.) Now, if the bid of colluder i is removed, it is no longer possible to accept all the remaining $n-1$ colluder bids, because $R_i \cup R_{n+1}$ cannot be covered exactly with noncolluder bids. It follows that the total value of all accepted bids when i's bid is removed can be at most $(3(n-2)+1)\sum_d |v(b_d)| + (n-2)M$. When i's bid is not omitted, the sum of the values of all accepted bids other than i's is at least $(3(n-1)-1)\sum_d |v(b_d)| + (n-1)M$. Subtracting the former quantity from this, we get that the VCG payment to i is at least $\sum_d |v(b_d)| + M$. ■

The next lemma shows that the necessary and sufficient condition above is equivalent to being able to partition all the items into n sets, so that no element of the partition can be covered exactly by a subset of the noncolluders' bids. That is, we can restrict our attention to the case where $R_{n+1} = \emptyset$.

Lemma 16 *The condition of Lemma 15 is satisfied if and only if the items can be partitioned into $T_1, \ldots, T_n$ such that no T_i can be covered exactly with bids from the noncolluders.*

Proof: For the "if" part: given T_i that satisfy the condition of this lemma, let $R_i = T_i$ for $i \leq n$, and $R_{n+1} = \emptyset$. Then no $R_i \cup R_{n+1} = T_i$ can be covered exactly with bids from the noncolluders, and $R_{n+1} = \emptyset$ can trivially be covered exactly with noncolluder bids. For the "only if" part: given R_i that satisfy the condition of Lemma 15, let $T_i = R_i$ for $i < n$, and let $T_n = R_n \cup R_{n+1}$. That T_n cannot be covered exactly by noncolluder bids now follows directly from the conditions of Lemma 15. But also, no T_i with $i < n$ can be covered exactly: because if it could, then we could cover $R_i \cup R_{n+1} = T_i \cup R_{n+1}$ using the bids that cover T_i exactly together with the bids that cover R_{n+1} exactly (which exist by the conditions of Lemma 15). ■

Combining all the lemmas, we get:

Theorem 30 *The n colluders can receive a payment of at least M each (simultaneously), where M is an arbitrarily large number, if and only if it is possible to partition the items into $T_1, \ldots, T_n$ such that no T_i can be covered exactly with bids from the noncolluders.*

Self-enforcing collusion

Again, a stronger condition is required if the collusion is also required to be self-enforcing.

Theorem 31 *The n colluders can receive a payment of at least M each (simultaneously), where M is an arbitrarily large number, if and only if it is possible to partition the items into $T_1, \ldots, T_n$ such that 1) no T_i can be covered exactly with noncolluder bids, 2) for no colluder i, the following holds: there exists a subset $T_i' \subseteq T_i$ such that T_i' can be covered exactly by a set of noncolluders' bids with total value greater than $v_i(T_i) - v_i(T_i - T_i')$ (the marginal value to colluder i of receiving T_i').*

Proof: For the "if" part, each colluder i can bid on T_i with a sufficiently large value. As in the above, this will give each colluder a payment of at least M. Moreover, no colluder i has an incentive to deviate, for the following reasons. Under the VCG mechanism, it is not possible to change a bidder i's bid in such a way that the allocation to i remains the same, but the payment to i changes. Therefore, we only need to consider what happens if colluder i bids on a different bundle. Bidding on items outside T_i will prevent one of the other colluders' bids from being accepted, leading to a severe reduction in the total value of the allocation, and therefore to a severe reduction in the payment to colluder i. Therefore, the only deviation that can possibly be advantageous is to bid on a subset T_i'' of T_i. Let $T_i' = T_i - T_i''$. If the colluder bids on T_i'' (with a sufficiently large value), one of two things may happen. First, it can be the case that it is not possible to exactly cover T_i' with noncolluder bids. If so, then it must be the case that one of the other colluders' bids cannot be accepted, leading again to a severe reduction in the payment to colluder i. Second, it can be the case that it is possible to exactly cover T_i' with noncolluder bids. In this case, the payment to colluder i will increase by the total value of this cover of T_i'. By the assumption in the theorem, this total value is at most $v_i(T_i) - v_i(T_i'')$, the marginal value to colluder i of receiving T_i'. It follows that the bid does not make the colluder better off.

For the "only if" part, we already know by Theorem 30 that in order for the n colluders to receive an arbitrarily large payment of at least M each (simultaneously), it must be possible to partition the items into $T_1, \ldots, T_n$ such that no T_i can be covered exactly with noncolluder bids (so that colluder i can bid on T_i with a sufficiently large value to achieve the desired outcome). But if for some colluder i, there exists a subset $T_i' \subseteq T_i$ such that T_i' can be covered exactly by a set of noncolluders' bids with total value greater than $v_i(T_i) - v_i(T_i - T_i')$, then this colluder would be better off bidding a sufficiently large value for $T_i - T_i'$ instead, because this would increase the payment to colluder i by more than the marginal value to colluder i of receiving T_i' as well. Hence the collusion would not be self-enforcing. ■

Complexity

In order to collude in the manner described above, the n colluders must solve the following computational problem.

Definition 23 (COVERLESS-PARTITION) *We are given a set I and a collection of subsets $S_1, S_2, \ldots, S_q \subseteq I$. We are asked whether there is a partition of I into subsets $T_1, T_2, \ldots T_n \subseteq I$ such that no T_i can be covered exactly by some of the S_i.*

Theorem 32 *Even if there is a singleton S_i for all but two elements a and b, and $n = 2$, COVERLESS-PARTITION is NP-complete.*

Proof: The problem is in NP in this case because given a partition T_1, T_2, either one of the T_i contains both a and b, in which case the other can be covered exactly with singleton sets; or they each contain one of a and b (say T_a contains a and T_b contains b). In the latter case, there is a cover of T_s if and only if it contains a subset containing s (the other elements in T_s can be covered with singleton sets), which can be checked in polynomial time.

To show that the problem is NP-hard, we reduce from SAT. Given an arbitrary SAT instance (given by a set of clauses C over variables V), let S be as follows. It contains a and b; for each variable $v \in V$, it contains an element s_v; and for each clause $c \in C$, it contains an element s_c. Let the collection of subsets be as follows. For every s_v, there is a subset $S_v = \{s_v\}$. For every s_c, there is a subset $S_c = \{s_c\}$. Finally, for every clause $c \in C$, there are two more subsets: one consisting of b, s_c, and all the variables that occur positively in c ($S_{c+} = \{b, s_c\} \cup \{s_v : +v \in c\}$), and one consisting of a, s_c, and all the variables that occur negatively in c ($S_{c-} = \{a, s_c\} \cup \{s_v : -v \in c\}$). We now show that the instances are equivalent.

First suppose there is a solution to the SAT instance, given by a labeling $t : V \to \{$*true, false*$\}$. Then let $\{a\} \cup \{s_v : t(v) =$*true*$\} \subseteq T_1$ and $\{b\} \cup \{s_v : t(v) =$*false*$\} \subseteq T_2$. Furthermore, if one of the variables occurring positively in c is set to *true*, let $s_c \in T_2$; otherwise, let $s_c \in T_1$. First, we claim that no subset S_{c+} is contained in some T_i. It is not contained in T_1 because it has b in it. If c is satisfied because of one of the variables v occurring positively in c is set to *true*, then $s_v \in T_1$, and because $s_v \in S_{c+}$, S_{c+} is not contained in T_2. Otherwise, $s_c \in T_1$, and again S_{c+} is not contained in T_2. Next, we claim that no subset S_{c-} is contained in some T_i. It is not contained in T_2 because it has a in it. If c is satisfied because of one of the variables v occurring positively in c is set to *true*, $s_c \in T_2$, and S_{c-} is not contained in T_1. Otherwise, one of the variables v occurring negatively in c must be set to *false*, so $v \in T_2$, and because $s_v \in S_{c-}$, again S_{c-} is not contained in T_1. Because only bids of the form S_{c+} or S_{c-} contain a or b, it follows that there is no exact cover of either T_1 or T_2, and we have a solution to the COVERLESS-PARTITION instance.

Now suppose there is a solution to the COVERLESS-PARTITION instance, given by a partition T_1, T_2. Because a and b cannot occur in the same T_i, suppose without loss of generality that $a \in T_1$ and $b \in T_2$. Then, set v to *true* if $s_v \in T_1$, and to *false* otherwise. Suppose that a given clause c is not satisfied with this assignment. This means that for all variables v that occur positively in c, $s_v \in T_2$, and for all variables v that occur negatively in c, $s_v \in T_1$. If $s_c \in T_1$, then S_{c-} is contained in T_1; and thus we can cover T_1 exactly with this set and singleton sets for the remaining elements. On the other hand, if $s_c \in T_2$, then S_{c+} is contained in T_2; and thus we can cover T_2 exactly with this set and singleton sets for the remaining elements. It follows that all clauses are satisfied with this assignment, and we have a solution to the SAT instance. ∎

An easier collusion problem

So far in this subsection, we have formulated the collusion problem so that *each* colluder should receive M, where M is an arbitrary amount. An easier problem for the colluders is to make sure that *together*, they receive M, where M is an arbitrary amount. Such a collusion may be less stable (because some of the colluders may be receiving very little). Nevertheless, as we will show, this type of collusion is possible whenever a weak (and easily verified, given the noncolluders' bids) condition holds: at least one item has no singleton bid on it. (A singleton bid is a bid on only one item.) We first show that this condition is necessary.

Lemma 17 *If at least one colluder receives a payment of more than* $\sum_d |v(b_d)|$ *(where* d *ranges over the noncolluders), then there is at least one item* s *on which no noncolluder places a singleton bid.*

Proof: If each item has a singleton noncolluder bid placed on it, then when we remove a colluder's bid, we can simply cover all the items in it with singleton bids (with a combined value of at least $-\sum_d |v(b_d)|$), and leave the rest of the allocation unchanged. It follows that the VCG payment to the colluder can be at most $\sum_d |v(b_d)|$). ■

We now show that the condition is sufficient.

Lemma 18 *If there is at least one item s on which no noncolluder places a singleton bid, then if one colluder bids $(\{s\}, 0)$, and the other colluder bids $(I - \{s\}, M + 2\sum_d |v(b_d)|)$ (for $M > 0$), then the total payment to the colluders is at least M.*

Proof: The colluders' bids will be the only accepted ones (because colluder 2's bid has a greater value than all other bids combined). If we removed colluder 2's bid, the total value of the accepted bids would be at most $\sum_d |v(b_d)|$), so colluder 2 will pay at most this much under the VCG mechanism. If we removed colluder 1's bid, colluder 2's bid could no longer be accepted (because $\{s\}$ cannot be covered by itself), and thus the total value of the accepted bids could be at most $\sum_d |v(b_d)|)$. It follows that colluder 1 is paid at least $M + \sum_d |v(b_d)|)$. So the total payment to the colluders is at least M ■

Combining the two lemmas, we get the desired result:

Theorem 33 *Two (or more) colluders can receive a total payment of M, where M is an arbitrarily large number, if and only if there is at least one item that has no singleton bid placed on it by a noncolluder.*

5.1.4 Combinatorial exchanges

We recall that in a *combinatorial exchange*, there is a set of items $I = \{s_1, s_2, \ldots, s_m\}$ that can be traded. A bid takes the form $b = (\lambda_1, \ldots, \lambda_m, v)$, where $\lambda_1, \ldots, \lambda_m, v \in \mathbb{R}$ (possibly negative). (Each λ_i is the number of units of the ith item that the bidder seeks to procure, and v is how much the bidder is wi lling to pay.) The winner determination problem is to label bids as accepted or rejected, under the constraint that the sum of the accepted vectors has its first m entries ≤ 0, to maximize the last entry of the sum of the accepted vectors. (This is assuming free disposal.) We will also use the notation $(\{(s_{i_1}, \lambda_{i_1}), (s_{i_2}, \lambda_{i_2}), \ldots, (s_{i_k}, \lambda_{i_k})\}, v)$ for representing a bid in which λ_{i_j} units of item s_{i_j} are demanded (and 0 units of each item that is not mentioned).

Characterization

In a combinatorial exchange with at least two items s_1 and s_2, let q_1 (respectively, q_2) be the total number of units of s_1 (respectively, s_2) offered for sale in bids so far (by noncolluders). Now consider the following two bids (by colluders): $(\{(s_1, q_1 + 1), (s_2, -q_2 - 1)\}, M + \sum_d |v(b_d)|)$ and $(\{(s_1, -q_1 - 1), (s_2, q_2 + 1)\}, M + \sum_d |v(b_d)|)$, where $M > 0$ and d ranges over the original (noncolluding) bids. Both these bids will be accepted (for otherwise, the total value of the accepted bids could be at most $M + 2\sum_d |v(b_d)| < 2(M + \sum_d |v(b_d)|)$). Moreover, if we remove one of these two bids, the other cannot be accepted (because its demand cannot be met), so the total value of the accepted bids can be at most $\sum_d |v(b_d)|$). It follows that the VCG payment to each of these two bidders is at least M. This proves the following:

Theorem 34 *In a combinatorial exchange with at least two items, for any set of bids by noncolluders, two colluders can place bids so that each of them will receive at least M, where M is an arbitrary amount. Moreover, each one receives exactly the items that the other provides, so that their net contribution in terms of items is nothing.*

This concludes the part of this dissertation analyzing problematic outcomes of the VCG mechanism in combinatorial auctions and exchanges. We will return to combinatorial auctions briefly in the next chapter, Section 6.5. In the next section, we consider mechanism design for negotiating over donations to charities.

5.2 Mechanism design for donations to charities

In this section, we study mechanism design for the setting of expressive preference aggregation for donations to charities described in Section 2.3. The rules that we described in that section for deciding on outcomes turn out not to be strategy-proof, as we will see shortly. This is not too surprising, because the mechanism described so far is, in a sense, a *first-price* mechanism, where the mechanism will extract as much payment from a bidder as her bid allows; and such mechanisms are typically not strategy-proof. In this section, we consider changing the rules to make bidding truthfully strategically optimal.

5.2.1 Strategic bids under the first-price mechanism

We first point out some reasons for bidders to misreport their preferences under the first-price mechanism described up to this point. First of all, even when there is only one charity, it may make sense to underbid one's true valuation for the charity. For example, suppose a bidder would like a charity to receive a certain amount x, but does not care if the charity receives more than that. Additionally, suppose that the other bids guarantee that the charity will receive at least x no matter what bid the bidder submits (and the bidder knows this). Then the bidder is best off not bidding at all (or submitting a utility for the charity of 0), to avoid having to make any payment. (This is an instance of the *free rider* problem [Mas-Colell *et al.*, 1995].)

With multiple charities, another kind of manipulation may occur, where the bidder attempts to steer others' payments towards her preferred charity. Suppose that there are two charities, and three bidders. The first bidder bids $u_1^1(\pi_{c_1}) = 1$ if $\pi_{c_1} \geq 1$, $u_1^1(\pi_{c_1}) = 0$ otherwise; $u_1^2(\pi_{c_2}) = 1$ if $\pi_{c_2} \geq 1$, $u_1^2(\pi_{c_2}) = 0$ otherwise; and $w_1(u_1) = u_1$ if $u_1 \leq 1$, $w_1(u_1) = 1 + \frac{1}{100}(u_1 - 1)$ otherwise. The second bidder bids $u_2^1(\pi_{c_1}) = 1$ if $\pi_{c_1} \geq 1$, $u_1^1(\pi_{c_1}) = 0$ otherwise; $u_2^2(\pi_{c_2}) = 0$ (always); $w_2(u_2) = \frac{1}{4}u_2$ if $u_2 \leq 1$, $w_2(u_2) = \frac{1}{4} + \frac{1}{100}(u_2 - 1)$ otherwise. Now, the third bidder's *true* preferences are accurately represented[7] by the bid $u_3^1(\pi_{c_1}) = 1$ if $\pi_{c_1} \geq 1$, $u_3^1(\pi_{c_1}) = 0$ otherwise; $u_3^2(\pi_{c_2}) = 3$ if $\pi_{c_2} \geq 1$, $u_3^2(\pi_{c_1}) = 0$ otherwise; and $w_3(u_3) = \frac{1}{3}u_3$ if $u_3 \leq 1$, $w_3(u_3) = \frac{1}{3} + \frac{1}{100}(u_3 - 1)$ otherwise. Now, it is straightforward to check that, if the third bidder bids truthfully, regardless of whether the objective is surplus maximization or total donated, charity 1 will receive at least 1, and charity 2 will receive less than 1. The same is true if bidder 3 does not place a bid at all (as in the previous type of manipulation); hence bidder 2's utility will be 1 in this case. But now, if bidder 3 reports $u_3^1(\pi_{c_1}) = 0$ everywhere; $u_3^2(\pi_{c_2}) = 3$ if $\pi_{c_2} \geq 1$, $u_3^2(\pi_{c_2}) = 0$ otherwise (this part of the bid is truthful); and $w_3(u_3) = \frac{1}{3}u_3$ if $u_3 \leq 1$, $w_3(u_3) = \frac{1}{3}$ otherwise; then charity 2 will receive at least 1, and bidder 3 will have to pay at most $\frac{1}{3}$. Because up to this amount of payment, one unit of money corresponds to three units of utility to bidder 3, it follows his utility is now at least $3 - 1 = 2 > 1$. We observe that in this case, the strategic bidder is not only affecting how much the bidders pay, but also how much the charities receive.

5.2.2 Mechanism design in the quasilinear setting

In the remainder of this section, we restrict our attention to bidders with quasilinear preferences. There are at least four reasons why the mechanism design approach is likely to be most successful in the setting of quasilinear preferences. First, historically, mechanism design has been been most successful when the quasilinear assumption could be made. Second, because of this success, some very general mechanisms have been discovered for the quasilinear setting (for instance, the VCG and dAGVA mechanisms) which we could apply directly to the expressive charity donation problem (although they are not fully satisfactory, as VCG is not budget-balanced, and dAGVA is not individually rational). Third, as we saw in Section 3.3.4, the clearing problem is much easier in this setting, and thus we are less likely to run into computational trouble for the mechanism design problem. Fourth, as we will show shortly, the quasilinearity assumption in some cases allows for decomposing the mechanism design problem over the charities (as it did for the simple clearing problem).

Moreover, in the quasilinear setting (unlike in the general setting), it makes sense to pursue social welfare (the sum of the utilities) as the objective, because now 1) units of utility correspond directly to units of money, so that we do not have the problem of the bidders arbitrarily scaling their utilities; and 2) it is no longer possible to give a payment willingness function of 0 while still affecting the donations through a utility function.

We are now ready to present the result that shows that we can sometimes decompose the problem over the charities.

Theorem 35 *Suppose all agents' preferences are quasilinear (and, as we have been assuming throughout, that the utility that an agent derives from one charity is independent of how much other charities receive). Furthermore, suppose that there exists a single-charity mechanism M that, for a certain subclass P of (quasilinear) preferences, under a given solution concept S (either implementation in dominant strategies or Bayes-Nash equilibrium) and a given notion of individual*

[7] Formally, this means that if the bidder is forced to pay the full amount that his bid allows for a particular vector of payments to charities, the bidder is indifferent between this and not participating in the mechanism at all. (Compare this to bidding truthfully in a first-price auction.)

rationality R (either ex post, ex interim, or none), satisfies a certain notion of budget balance (either ex post, ex ante, or none), and is ex-post efficient. Then, there exists a mechanism with the same properties for any number of charities—namely, the mechanism that runs the single-charity mechanism separately for each individual charity.

Proof: As stated in the theorem, the mechanism is simply the following: for each charity, run the single-charity mechanism on the agents' preferences for that charity, and let the agents make the corresponding payments to that charity. (So, each agent's total payment will be the sum of her payments to the individual charities.) Because the agents are assumed to be maximizing expected utility, and the utilities that they derive from different charities are independent, it follows by linearity of expectation that they can separate their truthfulness and participation decisions across the charities. Thus, the desired properties follow from the fact that the single-charity mechanism has these properties. ■

Two mechanisms that satisfy efficiency (and can in fact be applied directly to the multiple-charity problem without use of the previous theorem) are the VCG (which is incentive compatible in dominant strategies) and dAGVA (which is incentive compatible only in Bayes-Nash equilibrium) mechanisms. Each of them, however, has a drawback that would probably make it impractical in the setting of donations to charities. The VCG mechanism is not budget balanced. The dAGVA mechanism does not satisfy ex-post individual rationality. In the next subsection, we will investigate whether we can do better in the setting of donations to charities.

5.2.3 Impossibility of efficiency

In this subsection, we show that even in a very restricted setting, and with minimal requirements on incentive-compatibility and individual-rationality constraints, it is impossible to create a mechanism that is efficient.

Theorem 36 *There is no mechanism which is ex-post budget balanced, ex-post efficient, and ex-interim individually rational with Bayes-Nash equilibrium as the solution concept (even with only one charity, only two quasilinear bidders, with identical type distributions (uniform over two types, with either both utility functions being step functions or both utility functions being concave piece-wise linear functions)).*

Proof: Suppose the two bidders both have the following distribution over types. With probability $\frac{1}{2}$, the bidder does not care for the charity at all (u is zero everywhere); otherwise, the bidder derives utility $\frac{5}{4}$ from the charity getting at least 1, and utility 0 otherwise. (Alternatively, for the second type, the bidder can get $\min\{\frac{5}{4}, \frac{5\pi_c}{4}\}$—a concave piecewise linear function.) Call the first type the low type (L), the second one the high type (H).

Suppose a mechanism with the desired properties does exist. By the revelation principle, we can assume that revealing preferences truthfully is a Bayes-Nash equilibrium in this mechanism. Because the mechanism is ex-post efficient, the charity should receive exactly 1 when either bidder has the high type, and 0 otherwise. Let $\pi_1(\theta_1, \theta_2)$ be bidder 1's (expected) payment when she reports θ_1 and the other bidder reports θ_2. By ex-interim IR, $\pi_1(L, H) + \pi_1(L, L) \leq 0$. Because bidder one cannot have an incentive to report falsely when her true type is high, we have $\frac{5}{4} - \pi_1(L,H) - \pi_1(L,L) \leq \frac{5}{4} - \pi_1(H,H) + \frac{5}{4} - \pi_1(H,L)$, or equivalently $\pi_1(H,H) + \pi_1(H,L) \leq \frac{5}{4} + \pi_1(L,L) + \pi_1(L,H) \leq \frac{5}{4}$. Because the example is completely symmetric between bidders, we can similarly conclude for bidder 2's payments that $\pi_2(H,H) + \pi_2(L,H) \leq \frac{5}{4}$. Of course, in order to pay the charity the necessary amount of 1 whenever one of the bidders has her high type, we need to have $\pi_1(H,H) + \pi_1(H,L) + \pi_2(H,H) + \pi_2(L,H) + \pi_1(L,H) + \pi_2(H,L) = 3$, and thus we can conclude that $\pi_1(L,H) + \pi_2(H,L) \geq 3 - \frac{10}{4} = \frac{1}{2}$. Because the charity receives 0 when both report low, $\pi_1(L,L) + \pi_2(L,L) = 0$ and thus we can conclude that $\pi_1(L,H) + \pi_1(L,L) + \pi_2(H,L) + \pi_2(L,L) \geq \frac{1}{2}$. But by the individual rationality constraints, $\pi_1(L,H) + \pi_1(L,L) \leq 0$ and $\pi_2(H,L) + \pi_2(L,L) \leq 0$. (Contradiction.)[8] ■

The case of step-functions in this theorem corresponds exactly to the case of a single, fixed-size, nonexcludable public good (the "public good" being that the charity receives the desired amount)—for which such an impossibility result is already known [Mas-Colell *et al.*, 1995]. Many similar results are known, probably the most famous of which is the Myerson-Satterthwaite impossibility result, which proves the impossibility of efficient bilateral trade under the same requirements [Myerson and Satterthwaite, 1983].

Theorem 35 indicates that there is no reason to decide on donations to multiple charities under a single mechanism (rather than a separate one for each charity), when an efficient mechanism with the desired properties exists for the single-charity case. However, because under the requirements of Theorem 36, no such mechanism exists, there may be a benefit to bringing the charities under the same umbrella. The next proposition shows that this is indeed the case.

Proposition 8 *There exist settings with two charities where there exists no ex-post budget balanced, ex-post efficient, and ex-interim individually rational mechanism with Bayes-Nash equilibrium as the solution concept for either charity alone; but there exists an ex-post budget balanced, ex-post efficient, and ex-post individually rational mechanism with dominant strategies as the solution concept for both charities together. (Even when the conditions are the same as in Theorem 36, apart from the fact that there are now two charities.)*

Proof: Suppose that each bidder has two types, With probability $\frac{1}{2}$ each: for the first type, her preferences for the first charity correspond to the high type in the proof of Theorem 36, and her preferences for the second charity correspond to the low type in the proof of Theorem 36. For the second type, her preferences for the first charity correspond to the low type, and her preferences for the second charity correspond to the high type. Now, if we wish to create a mechanism for either charity individually, we are in exactly the same setting as in the proof of Theorem 36, where

[8]As an alternative proof technique (a proof by computer), we let our automated mechanism design software (described in Chapter 6) create a mechanism for the (step-function) instance described in the proof, which was restricted to be implementable in dominant strategies, ex-interim individually rational, and (weak) budget balanced, with social welfare (counting the payments made) as the objective. The mechanism did not burn any money (did not pay unnecessarily much to the charity), but did not always give money to the charity when it was beneficial to do so. (It randomized uniformly between giving 1 and giving 0 when player one's type was low, and player 2's high.) Since an ex-post budget balanced, ex-post efficient mechanism would have had a higher expected objective value, and automated mechanism design always finds the mechanism that maximizes the expected objective value under the constraints it is given, we can conclude that no ex-post budget balanced, ex-post efficient mechanism exists under the given constraints.

we know that it is impossible to get all of ex-post budget balance, ex-post efficiency, and ex-interim individually rationality in Bayes-Nash equilibrium. On the other hand, consider the following mechanism for the joint problem. If both bidders report preferring the same charity, each bidder pays $\frac{1}{2}$, and the preferred charity receives 1 (the other 0). Otherwise, each bidder pays 1, and each charity receives 1. It is straightforward to check that the mechanism is ex-post budget balanced, ex-post efficient, and ex-post individually rational. To see that truthtelling is a dominant strategy, we need to check two cases. First, if one bidder reports a high type for the charity that the other bidder does not prefer, this latter bidder is better off reporting truthfully: reporting falsely will give her utility $-\frac{1}{2}$ (nothing will be donated to her preferred utility), which is less than reporting truthfully by ex-post IR. Second, if one bidder reports a high type for the charity that the other bidder prefers, this latter bidder is better off reporting truthfully as well: her preferred charity will receive the same amount regardless of her report, but her required payment is only $\frac{1}{2}$ if she reports truthfully, as opposed to 1 if she reports falsely. ■

This concludes the part of this dissertation studying expressive preference aggregation for donations to charities.

5.3 Summary

In this chapter, we studied problems that classical mechanism design faces in some expressive preference aggregation settings. In Section 5.1, we studied two related problems concerning the VCG mechanism: the problem of revenue guarantees, and that of collusion. We studied four settings: combinatorial forward auctions with free disposal, combinatorial reverse auctions with free disposal, combinatorial forward (or reverse) auctions without free disposal, and combinatorial exchanges. In each setting, we gave an example of how additional bidders (colluders) can make the outcome much worse (less revenue or higher cost) under the VCG mechanism (but not under a first price mechanism); derived necessary and sufficient conditions for such an effective collusion to be possible under the VCG mechanism; and (when nontrivial) studied the computational complexity of deciding whether these conditions hold.

In Section 5.2, we studied mechanism design for expressive preference aggregation for donations to (charitable) causes. We showed that even with only a single charity, a fundamental impossibility result similar to the Myerson-Satterthwaite impossibility theorem holds; but we also showsed some positive results, including how mechanisms that are successful in single-charity settings can be extended to settings with multiple charities, and how combining the aggregation of preferences over donations to multiple individual charities into a single mechanism can improve efficiency.

The work in this chapter provides some reasons why simply taking a standard mechanism “off the shelf” is not always satisfactory, especially in domains with complex preferences. Rather, it may be preferable to design a custom mechanism. The next chapter takes this idea to its extreme: we will study how an optimal mechanism can be automatically designed (computed) *for the specific instance at hand only*.

Chapter 6

Automated Mechanism Design

Mechanism design has traditionally been a manual endeavor. The designer uses experience and intuition to hypothesize that a certain rule set is desirable in some ways, and then tries to prove that this is the case. Alternatively, the designer formulates the mechanism design problem mathematically and characterizes desirable mechanisms analytically in that framework. These approaches have yielded a small number of canonical mechanisms over the last 40 years, the most significant of which we discussed in Chapter 4. Each of these mechanisms is designed for a class of settings and a specific objective. The upside of these mechanisms is that they do not rely on (even probabilistic) information about the agents' preferences (*e.g.* Vickrey-Clarke-Groves mechanisms), or they can be easily applied to any probability distribution over the preferences (*e.g.* the dAGVA mechanism, the Myerson auction, and the Maskin-Riley multi-unit auction). However, these general mechanisms also have significant downsides:

- The most famous and most broadly applicable general mechanisms, VCG and dAGVA, only maximize social welfare. If the designer is self-interested, as is the case in many electronic commerce settings, these mechanisms do not maximize the designer's objective.

- The general mechanisms that do focus on a self-interested designer are only applicable in very restricted settings. For example, Myerson's expected revenue maximizing auction is for selling a single item, and Maskin and Riley's expected revenue maximizing auction is for selling multiple identical units of an item.

- Even in the restricted settings in which these mechanisms apply, the mechanisms only allow for payment maximization. In practice, the designer may also be interested in the outcome *per se*. For example, an auctioneer may care which bidder receives the item.

- It is often assumed that side payments can be used to tailor the agents' incentives, but this is not always practical. For example, in barter-based electronic marketplaces—such as mybarterclub.com, Recipco, and National Trade Banc—side payments are not allowed. Furthermore, among software agents, it might be more desirable to construct mechanisms that do not rely on the ability to make payments, because many software agents do not have the infrastructure to make payments.

- The most common mechanisms (*e.g.*, VCG, dAGVA, the Myerson auction, and the Maskin-Riley auction) assume that the agents have quasilinear preferences—that is, they assume that the utility function of each agent $i \in \{1, \ldots, n\}$ can be written as $u_i(o, \pi_1, \ldots, \pi_n) = v_i(o) - \pi_i$, where o is the outcome and π_i is the amount that agent i has to pay. So, very restrictively, it is assumed that 1) the agent's valuation, v_i, of outcomes is independent of money, 2) the agent does not care about other agents' payments, and 3) the agent is risk neutral.

In sharp contrast to manual mechanism design, in this chapter we introduce a systematic approach—called *automated mechanism design (AMD)*—where the mechanism is automatically created for the setting and objective at hand [Conitzer and Sandholm, 2002b].[1] This has at least four important advantages:

- It can be used in settings beyond the classes of problems that have been successfully studied in (manual) mechanism design to date.
- It can allow one to circumvent the impossibility results: when the mechanism is designed for the setting (instance) at hand, it does not matter that it would not work on preferences beyond those in that setting (*e.g.*, for a class of settings). Even when the optimal mechanism—created automatically—does not circumvent the impossibility, it always minimizes the pain entailed by impossibility.
- It can yield better mechanisms (in terms of better outcomes and/or stronger nonmanipulability guarantees[2]) than the canonical mechanisms because the mechanism capitalizes on the particulars of the setting (the probabilistic (or other) information that the mechanism designer has about the agents' preferences). Given the vast amount of information that parties have about each other today, it is astonishing that the canonical mechanisms (such as first-price reverse auctions), which ignore that information, have prevailed thus far. It seems likely that future mechanisms will be created automatically. For example, imagine a Fortune 1000 company automatically creating its procurement mechanism based on its statistical knowledge about its suppliers (and potentially also the public prices of the suppliers' inputs, *etc.*). Initial work like this is already being conducted at CombineNet, Inc.
- It shifts the burden of mechanism design from humans to a machine.

The rest of this chapter is layed out as follows. In Section 6.1, we define the basic computational problem of automated mechanism design [Conitzer and Sandholm, 2002b]. In Section 6.2, we illustrate the types of mechanism that automated mechanism design can create, using divorce settlement as an example [Conitzer and Sandholm, 2003a]. In Section 6.3, we show that several variants of the problem of designing an optimal deterministic mechanism are hard [Conitzer and Sandholm, 2003b,

[1] Automated mechanism design is completely different from *algorithmic mechanism design* [Nisan and Ronen, 2001]. In the latter, the mechanism is designed manually with the goal that *executing* the mechanism is computationally tractable. On the other hand, in automated mechanism design, the mechanism itself is designed automatically. Some work on automatically choosing the mechanism to use that preceded our work was done by Cliff [2001], Byde [2003], and Phelps *et al.* [2002]. These works focused on setting a parameter of the mechanism (rather than searching through the space of all possible mechanisms, as we do here), and evaluated the resulting mechanism based on agents that they evolved with the mechanism (rather than requiring truthfulness).

[2] For example, satisfaction of *ex post* IC and/or IR constraints rather than their *ex interim* variants.

2004f]. In Section 6.4, we show that optimal randomized mechanisms can be designed in polynomial time using a linear programming formulation, and that a mixed integer programming version of this formulation can be used to design optimal deterministic mechanisms [Conitzer and Sandholm, 2002b, 2003b, 2004f]. In Section 6.5, we demonstrate some initial applications of automated mechanism design [Conitzer and Sandholm, 2003a]. In Section 6.6, we give experimental scalability results for the linear/mixed integer programming techniques Conitzer and Sandholm [2003a]. In Section 6.7, we give and study a special-purpose algorithm for the special case of designing a deterministic mechanism for a single agent that does not use payments [Conitzer and Sandholm, 2004a]. In Section 6.8, we introduce a representation that can be more concise than the straightforward representation of automated mechanism design problem instances, and study how using this representation affects the complexity of the problem [Conitzer and Sandholm, 2003c].

6.1 The computational problem

In this section, we define the computational problem of automated mechanism design. First, we define an instance of the problem as follows.

Definition 24 *In an* automated mechanism design setting*, we are given*
A finite set of outcomes O;
A finite set of n agents;
For each agent i,

- *a finite*[3] *set of* types *set of types Θ_i,*
- *a probability distribution γ_i over Θ_i (in the case of correlated types, there is a single joint distribution γ over $\Theta_1 \times \ldots \times \Theta_n$),*
- *a utility function $u_i : \Theta_i \times O \to \mathbb{R}$;*

An objective function whose expectation the designer wishes to maximize.

There are many possible objective functions the designer might have, for example, social welfare (where the designer seeks to maximize the sum of the agents' utilities, $\sum\limits_{i=1}^{n} u_i(\theta, o)$, or $\sum\limits_{i=1}^{n} u_i(\theta, o) - \pi_i$ if payments are taken into account), or the minimum utility of any agent (where the designer seeks to maximize the worst utility had by any agent, $\min_i u_i(\theta, o)$, or $\min_i u_i(\theta, o) - \pi_i$ if payments are taken into account). In both of these cases, the designer is *benevolent*, because the designer, in some sense, is pursuing the agents' collective happiness. On the other hand, a *self-interested* designer cares only about the outcome chosen (that is, the designer does not care how the outcome relates to the agents' preferences, but rather has a fixed preference over the outcomes), and about the net payments made by the agents, which flow to the designer. Specifically, a self-interested designer

[3] It should be noted that in mechanism design, the type space is often continuous. However, the techniques described in this chapter require a finite number of types. One can approximate a continuous type space with a discretized type space, but perhaps future research will discover more elegant and better methods for dealing with continuous type spaces.

has an objective function $g(o) + \sum_{i=1}^{n} \pi_i$, where $g : O \to \mathbb{R}$ indicates the designer's own preference over the outcomes, and π_i is the payment made by agent i. In the case where $g = 0$ everywhere, the designer is said to be *payment maximizing*. In the case where payments are not possible, g constitutes the objective function by itself.

We can now define the computational problem of automated mechanism design.

Definition 25 *(AUTOMATED-MECHANISM-DESIGN (AMD)) We are given an automated mechanism design setting, an IR notion (*ex interim*,* ex post*, or none), and a solution concept (dominant strategies or Bayes-Nash equilibrium). Also, we are told whether payments are possible, and whether randomization is possible. Finally, we are given a target value G. We are asked whether there exists a mechanism of the specified type that satisfies both the IR notion and the solution concept, and gives an expected value of at least G for the objective.*[4]

6.2 A tiny example: Divorce settlement

To get some intuition about the types of mechanism that AMD generates, in this section, we apply AMD to divorce settlement. We study several variants of the mechanism design problem, and the optimal solutions (mechanisms) to those variants generated by our AMD implementation (described later). We first study a benevolent arbitrator, then a benevolent arbitrator that uses payments to structure the agents' incentives, and finally a greedy arbitrator that wants to maximize the sum of side payments from the agents—while still motivating the agents to come to the arbitration.

6.2.1 A benevolent arbitrator

A couple is getting a divorce. They jointly own a painting and the arbitrator has to decide what happens to the painting. There are 4 options to decide among: (1) the husband gets the painting, (2) the wife gets the painting, (3) the painting remains in joint ownership and is hung in a museum, and (4) the painting is burned. The husband and wife each have two possible types: one that implies not caring for the painting too much (low), and one that implies being strongly attached to the painting (high). (low) is had with probability .8, (high) with .2, by each party. To maximize social welfare, the arbitrator would like to give the painting to whoever cares for it more, but even someone who does not care much for it would prefer having it over not having it, making the arbitrator's job in ascertaining the preferences nontrivial. Specifically, the utility function is (for either party)

```
u(low,get the painting)=2
u(low,other gets the painting)=0
u(low,joint ownership)=1
u(low,burn the painting)=-10 (both parties feel
 that burning the painting would be a terrible
 thing from an art history perspective)
u(high,get the painting)=100
```

[4] For studying computational complexity, we phrase AMD as a decision problem, but the corresponding optimization problem is clear.

```
u(high,other gets the painting)=0
u(high,joint ownership)=50
u(high,burn the painting)=-10
```

Let us assume (for now) that side payments are not possible, randomization is not possible, and that implementation in dominant strategies is required. Now we have a well-specified AMD instance. Our solver generated the following optimal mechanism for this setting:

```
           husband_low            husband_high
wife_low   husband gets painting  husband gets painting
wife_high  husband gets painting  husband gets painting
```

That is, we cannot do better than always giving the painting to the husband (or always giving it to the wife). (The solver does not look for the "fairest" mechanism because fairness is not part of the objective we specified.) Now let us change the problem slightly, by requiring only implementation in BNE. For this instance, our solver generated the following optimal mechanism:

```
           husband_low          husband_high
wife_low   joint ownership      husband gets painting
wife_high  wife gets painting   painting is burned
```

Thus, when we relax the incentive compatibility constraint to BNE, we can do better by sometimes burning the painting! The burning of the painting (with which nobody is happy) is sufficiently helpful in tailoring the incentives that it becomes a key part of the mechanism. (This is somewhat similar to the item not being sold in an optimal (*i.e.*, revenue-maximizing) auction—more on optimal auctions later.) Now let us see whether we can do better by also allowing for randomization in the mechanism. It turns out that we can, and the optimal mechanism generated by the solver is the following:

```
           husband_low                husband_high
wife_low   .57: husband, .43: wife    1: husband
wife_high  1: wife                    .45: burn; .55: husband
```

The randomization helps us because the threat of burning the painting *with some probability* when both report high is enough to obtain the incentive effect that allows us to give the painting to the right party in other settings. Interestingly, the mechanism now chooses to randomize over the party that receives the painting rather than awarding joint ownership in the setting where both report low.

6.2.2 A benevolent arbitrator that uses payments

Now imagine that we can force the parties to pay money, depending on the types reported—that is, side payments are possible. The arbitrator (for now) is still only concerned with the parties' welfare—taking into account how much money they lose because of the payment rule, as well as the allocation of the painting.[5] Thus, it does not matter to the arbitrator whether the agents' net payment goes to the arbitrator, a charity, or is burned, but other things being equal the arbitrator would like to minimize the payments that the agents make. Now the optimal deterministic mechanism in dominant strategies generated by the solver has the following allocation rule:

```
           husband_low            husband_high
wife_low   husband gets painting  husband gets painting
wife_high  wife gets painting     wife gets painting
```

The payment function is (wife's payment listed first):

```
           husband_a  husband_high
wife_low   0,0        0,0
wife_high  2,0        2,0
```

In this mechanism, the allocation of the painting is always optimal. However, the price (in terms of social welfare) that is paid for this is that the wife must sometimes pay money; the fact that she has to pay 2 whenever she reports her high type removes her incentive to falsely report her high type.

6.2.3 An arbitrator that attempts to maximize the payments extracted

Now we imagine a non-benevolent arbitrator, who is running an arbitration business. The agents' net payments now go to the arbitrator, who is seeking to maximize these payments. Of course, the arbitrator cannot extract arbitrary amounts from the parties; rather, the parties should overall still be happy with their decision to go to the arbitrator. Thus, we need an IR constraint. If we require ex post IR and dominant strategies, the optimal deterministic mechanism generated by the solver has the following allocation rule:

```
           husband_low         husband_high
wife_low   painting is burned  husband gets painting
wife_high  wife gets painting  wife gets painting
```

Now the painting is burned when both parties report their low types! (This is even more similar to an item not being sold in an optimal combinatorial auction.) As for the mechanism's payment function: in this setting, the arbitrator is always able to extract *all* of each agent's utility from the allocation as her payment (but note that the allocation is not always optimal: the painting is burned sometimes, in which case the arbitrator obtains no revenue, but rather has to compensate the parties involved for the loss of the painting).

Many other specifications of the problem are possible, but we will not study them here.

[5] Classical mechanism design often does not count the payments in the social welfare calculation (*e.g.*, the VCG mechanism), allowing for easier analysis; one of the benefits of automated mechanism design is that the payments made can easily be integrated into the social welfare calculation in designing the mechanisms.

6.3 Complexity of designing deterministic mechanisms

This section characterizes the computational complexity of automated mechanism design for the case where the designed mechanism is required to be deterministic. An interesting special case is the setting where there is only one agent (or, more generally, only one *type-reporting* agent). In this case, the agent always knows everything there is to know about the other agents' types—because there is nothing to know about their types. Since *ex post* and *ex interim* IR only differ on what an agent is assumed to know about other agents' types, the two IR concepts coincide here. Also, because implementation in dominant strategies and implementation in Bayes-Nash equilibrium only differ on what an agent is assumed to know about other agents' types, the two solution concepts coincide here. This observation is a useful tool in proving hardness results: if we prove computational hardness in the single-agent setting, this immediately implies hardness for both IR concepts, for both solution concepts, and for any constant number of agents.

In this section, we will show that most variants of the automated mechanism design problem are hard (NP-complete) even in the single-agent setting, *if* the mechanism is required to be deterministic. (In contrast, we will show in Section 6.4 that allowing for randomized mechanisms makes the problem solvable in polynomial time.) Most of the reductions are from the MINSAT problem:

Definition 26 (MINSAT) *We are given a formula ϕ in conjunctive normal form, represented by a set of Boolean variables V and a set of clauses C, and an integer K ($K < |C|$). We are asked whether there exists an assignment to the variables in V such that at most K clauses in ϕ are satisfied.*

MINSAT was recently shown to be NP-complete [Kohli *et al.*, 1994].

We first show that the problem of designing a welfare-maximizing deterministic mechanism that does not use payments is NP-complete. Of course, this problem is easy if there is only a single agent: in this case, the welfare-maximizing mechanism is to always give the agent one of its most preferred outcomes. However, we show that if, in addition to the type-reporting agent, there is an additional agent that does not report a type (for example, because its type is common knowledge), then the problem becomes NP-complete.

Theorem 37 *The AMD problem for designing deterministic mechanisms without payments is NP-complete, even when the objective is social welfare, there is only a single type-reporting agent (in addition to an agent that does not report a type), and the probability distribution over Θ is uniform. (Membership in NP is guaranteed only if the number of agents is constant.)*

Proof: The problem is in NP when the number of agents is constant because we can nondeterministically generate an outcome selection function, and subsequently verify in polynomial time whether it is nonmanipulable, and whether the expectation of the objective function achieves the threshold. (We note that if we do not restrict the number of agents, then the outcome selection function will have exponential size.) To show that the problem is NP-hard, we reduce an arbitrary MINSAT instance to an automated mechanism design instance as follows.

Let the outcomes O be as follows. For every clause $c \in C$, there is an outcome o_c. For every variable $v \in V$, there is an outcome o_v and an outcome o_{-v}. Finally, there is a single additional outcome o_b.

Let L be the set of literals, that is, $L = \{v : v \in V\} \cup \{-v : v \in V\}$. Then, let the type space Θ be as follows. For every clause $c \in C$, there is a type θ_c. For every variable $v \in V$, there is a type θ_v. The probability distribution over Θ is uniform.

Let the utility function be as follows:

- $u(\theta_v, o_v) = u(\theta_v, o_{-v}) = |C| + 3$ for all $v \in V$;
- $u(\theta_c, o_l) = 1$ for all $c \in C$ and $l \in c$ (that is, l is a literal that occurs in c);
- $u(\theta_c, o_c) = 1$ for all $c \in C$;
- u is 0 everywhere else.

Let $g(\theta, o) = u(\theta, o) + v(o)$, where $v(o_b) = 2$ and v is 0 everywhere else. (Here, v represents the utility of the agent that does not report a type.) Finally, let $G = \frac{|V|(|C|+3)+2|C|-k}{|V|+|C|}$ (k is the threshold of the MINSAT instance). We claim that the automated mechanism design instance has a solution if and only if the MINSAT instance has a solution.

First suppose that the MINSAT instance has a solution, that is, an assignment to the variables that satisfies at most k clauses. Then consider the following mechanism. If $v \in V$ is set to *true* in the assignment, then set $o(\theta_v) = o_v$; if it is set to *false*, then set $o(\theta_v) = o_{-v}$. If $c \in C$ is satisfied by the assignment, then set $o(\theta_c) = o_c$; if it is not satisfied, then set $o(\theta_c) = o_b$. First we show that this mechanism is nonmanipulable. If the agent's type is either any one of the θ_v or one of the θ_c corresponding to a satisfied clause c, then the mechanism gives the agent the maximum utility it can possibly get with that type, so there is no incentive for the agent to misreport. On the other hand, if the agent's type is one of the θ_c corresponding to a nonsatisfied clause c, then any outcome o_l corresponding to a literal l in c, or o_c, would give utility 1, as opposed to o_b (which the mechanism actually chooses for θ_c) which gives the agent utility 0. It follows that the mechanism is nonmanipulable if and only if there is no other θ such that $o(\theta)$ is any outcome o_l corresponding to a literal l in c, or o_c. It is easy to see that there is indeed no θ such that $o(\theta) = o_c$. There is also no θ such that $o(\theta)$ is any outcome o_l corresponding to a literal l in c: this is because the only type that could possibly give the outcome o_l is θ_v, where v is the variable corresponding to l; but because c is not satisfied in the assignment to the variables, we know that actually, $o(\theta_v) = o_{-l}$ (that is, the outcome corresponding to the opposite literal is chosen). It follows that the mechanism is indeed nonmanipulable. All that is left to show is that the expected value of $g(\theta, o(\theta))$ reaches G. For any θ_v we have $g(\theta_v, o(\theta_v)) = |C| + 3$. For any θ_c where c is a satisfied clause, we have $g(\theta_c, o(\theta_c)) = 1$. Finally, for any θ_c where c is an unsatisfied clause, we have $g(\theta_c, o(\theta_c)) = 2$. If s is the number of satisfied clauses, then, using the facts that the probability distribution over Θ is uniform and that $s \leq k$, we have $E[g(\theta, o(\theta))] = \frac{|V|(|C|+3)+s+2(|C|-s)}{|V|+|C|} \geq \frac{|V|(|C|+3)+2|C|-k}{|V|+|C|} = G$. So there is a solution to the automated mechanism design instance.

Now suppose there is a solution to the automated mechanism design instance, that is, a nonmanipulable mechanism given by an outcome function $o : \Theta \to O$, which leads to an expected value of $g(\theta, o(\theta))$ of at least G. We observe that the maximum value that we can get for $g(\theta, o(\theta))$ is $|C| + 3$ when θ is one of the θ_v, and 2 otherwise. Thus, if for some v it were the case that $o(\theta_v) \notin \{o_v, o_{-v}\}$ and hence $g(\theta, o(\theta)) \leq 2$, it would follow that $E[g(\theta, o(\theta))]$ can be at most

$\frac{(|V|-1)(|C|+3)+2(|C|+1)}{|V|+|C|} < \frac{(|V|)(|C|+3)+|C|}{|V|+|C|} < \frac{|V|(|C|+3)+2|C|-k}{|V|+|C|} = G$ (because $k < |C|$). (Contradiction.) It follows that for all v, $o(\theta_v) \in \{o_v, o_{-v}\}$. From this we can derive an assignment to the variables: set v to *true* if $o(\theta_v) = o_v$, and to *false* if $o(\theta_v) = o_{-v}$. We claim this assignment is a solution to the MINSAT instance for the following reason. If a clause c is satisfied by this assignment, there is some literal l such that $l \in c$ and $o(\theta_v) = o_l$ for the corresponding variable v. But then $o(\theta_c)$ cannot be o_b, because if it were, the agent would be motivated to report θ_v when its true type is θ_c, to get a utility of 1 as opposed to the 0 it would get for reporting truthfully. Hence $g(\theta_c, o(\theta_c))$ can be at most 1 for a satisfied clause c. It follows that $E[g(\theta, o(\theta))]$ can be at most $\frac{|V|(|C|+3)+s+2(|C|-s)}{|V|+|C|}$ where s is the number of satisfied clauses. But because $E[g(\theta, o(\theta))] \geq G$, we can conclude $\frac{|V|(|C|+3)+s+2(|C|-s)}{|V|+|C|} \geq G = \frac{|V|(|C|+3)+2|C|-k}{|V|+|C|}$, which is equivalent to $s \leq k$. So there is a solution to the MINSAT instance. ■

We note that the previous result is in contrast to the case where payments are allowed: in that case, the VCG mechanism constitutes an optimal mechanism, and it can be computed in polynomial time. We may wonder if the ability to use payments makes the automated mechanism design problem easy in all cases. The following theorem shows that this is not the case: there are objective functions (that do not depend on the payments made) such that designing the optimal deterministic mechanism is hard even when the mechanism is allowed to use payments.

Theorem 38 *The AMD problem for designing deterministic mechanisms with payments is NP-complete, even when the objective does not depend on the payments made, there is only a single agent, and the probability distribution over Θ is uniform. (Membership in NP is guaranteed only if the number of agents is constant.)*

Proof: First we show that the problem is in NP. When the number of agents is constant, we can nondeterministically generate an outcome function o. We then check whether the payment function π can be set so as to make the mechanism nonmanipulable. Because we have already generated o, we can phrase this problem as a linear program with the following constraints: for all $\theta, \hat{\theta} \in \Theta$, $u(\theta, o(\theta)) + \pi(\theta) \geq u(\theta, o(\hat{\theta})) + \pi(\hat{\theta})$. If the linear program has a solution, we subsequently check if the corresponding mechanism achieves the threshold G for $E[g(\theta, o(\theta))]$.

To show that the problem is NP-hard, we reduce an arbitrary INDEPENDENT-SET instance to an automated mechanism design instance as follows. For every vertex $v \in V$, let there be outcomes o_v^1 and o_v^2, and a type θ_v. The probability distribution over Θ is uniform. Let the utility function be as follows:

- $u(\theta_v, o_w^1) = 1$ for all $v, w \in V$ with $(v, w) \in E$;
- $u(\theta_v, o_w^1) = 0$ for all $v, w \in V$ with $(v, w) \notin E$ (this includes all cases where $v = w$ as there are no self-loops in the graph);
- $u(\theta_v, o_v^2) = 1$ for all $v \in V$;
- $u(\theta_v, o_w^2) = 0$ for all $w \in V$ with $v \neq w$.

Let the objective function be $g(\theta_v, o_v^1) = 1$ for all $v \in V$, and $g() = 0$ everywhere else. Finally, let $G = \frac{k}{|V|}$ (where k is the threshold of the INDEPENDENT-SET instance). We claim that the automated mechanism design instance has a solution if and only if the INDEPENDENT-SET instance has a solution.

First suppose that the INDEPENDENT-SET instance has a solution, that is, some $I \subseteq V$ of size at least k such that no two elements of I have an edge between them. Then consider the following mechanism. For all $v \in I$, let $o(\theta_v) = o_v^1$. For all $v \notin V$, let $o(\theta_v) = o_v^2$. Let π be zero everywhere (no payments are made). First we show that this mechanism is indeed nonmanipulable. If $v \in I$ and $w \in I$, then (because I is an independent set) $(v, w) \notin I$, and thus $u(\theta_v, o(\theta_v)) + \pi(\theta_v) = u(\theta_v, o_v^1) = 0 = u(\theta_v, o_w^1) = u(\theta_v, o(\theta_w)) + \pi(\theta_w)$. If $v \in I$ and $w \notin I$, then $u(\theta_v, o(\theta_v)) + \pi(\theta_v) = u(\theta_v, o_v^1) = 0 = u(\theta_v, o_w^2) = u(\theta_v, o(\theta_w)) + \pi(\theta_w)$. Finally, if $v \notin I$, then $u(\theta_v, o(\theta_v)) + \pi(\theta_v) = u(\theta_v, o_v^2) = 1$, which is the highest possible value the agent can attain. So there is no incentive for the agent to misreport anywhere. All that is left to show is that the expected value of $g(\theta, o(\theta))$ reaches G. For $v \in I$, $g(\theta, o(\theta)) = g(\theta, o_v^1) = 1$, and for $v \notin I$, $g(\theta, o(\theta)) = g(\theta, o_v^2) = 0$. Because the distribution over Θ is uniform, it follows that $E[g(\theta, o(\theta))] = \frac{|I|}{|V|} \geq \frac{k}{|V|} = G$. So there is a solution to the automated mechanism design instance.

Now suppose there is a solution to the automated mechanism design instance, that is, a nonmanipulable mechanism given by an outcome function $o : \Theta \to O$ and a payment function $\pi : \Theta \to \mathbb{R}$, which leads to an expected value of $g(\theta, o(\theta))$ of at least G. Let $I = \{v : o(\theta) = o_v^1\}$. We claim I is a solution to the INDEPENDENT-SET instance. First, because $g(\theta_v, o(\theta_v))$ is 1 only for $v \in I$, we know that $\frac{k}{|V|} = G \leq E[g(\theta, o(\theta))] = \frac{|I|}{|V|}$, or equivalently, $|I| \geq k$. All that is left to show is that there are no edges between elements of I. Suppose there were an edge between $v, w \in I$. Without loss of generality, say $\pi(\theta_v) \leq \pi(\theta_w)$. Then, $u(\theta_v, o(\theta_v)) + \pi(\theta_v) = u(\theta_v, o_v^1) + \pi(\theta_v) = \pi(\theta_v) \leq \pi(\theta_w) < 1 + \pi(\theta_w) = u(\theta_v, o_w^1) + \pi(\theta_w) = u(\theta_v, o(\theta_w)) + \pi(\theta_w)$. So the agent has an incentive to misreport when its type is θ_v, which contradicts the nonmanipulability of the mechanism. It follows that there are no edges between elements of I. So there is a solution to the INDEPENDENT-SET instance. ■

The objective functions studied up to this point depended on the agents' types. However, this is not the case for so-called *self-interested* designers, who are concerned only with how the chosen outcome fits their own goals and the payments collected. Formally, we say that the designer is self-interested if the objective function takes the form $g(o) + \sum_{i=1}^{n} \pi_i$, where $g : O \to \mathbb{R}$ indicates the designer's own preference over the outcomes, and π_i is the payment made by agent i. While the previous complexity results did not depend on the presence of an IR constraint, the automated mechanism design problem is trivial for a self-interested designer without an IR constraint: the designer can simply choose the outcome that it likes best, and force the agents to pay an unbounded amount. Hence, the following hardness results depend on the presence of an IR constraint. We first study the case where the objective is $\sum_{i=1}^{n} \pi_i$, that is, the designer is only interested in maximizing the total payment, and show that the problem of designing an optimal mechanism in this case is NP-complete.

Theorem 39 *The AMD problem for designing deterministic mechanisms is NP-complete, even* *when the objective is to maximize total payments made (under an IR constraint), there is only a single agent, and the probability distribution over Θ is uniform. (Membership in NP is guaranteed only if the number of agents is constant.)*

Proof: The proof of membership in NP for a constant number of agents is similar to previous proofs. To show NP-hardness, we reduce an arbitrary MINSAT instance to the following automated mechanism design instance. Let the agent's type set be $\Theta = \{\theta_c : c \in C\} \cup \{\theta_v : v \in V\}$, where C is the set of clauses in the MINSAT instance, and V is the set of variables. Let the probability distribution over these types be uniform. Let the outcome set be $O = \{o_0\} \cup \{o_c : c \in C\} \cup \{o_l : l \in L\}$, where L is the set of literals, that is, $L = \{+v : v \in V\} \cup \{-v : v \in V\}$. Let the notation $v(l) = v$ denote that v is the variable corresponding to the literal l, that is, $l \in \{+v, -v\}$. Let $l \in c$ denote that the literal l occurs in clause c. Then, let the agent's utility function be given by $u(\theta_c, o_l) = |\Theta| + 1$ for all $l \in L$ with $l \in c$; $u(\theta_c, o_l) = 0$ for all $l \in L$ with $l \notin c$; $u(\theta_c, o_c) = |\Theta| + 1$; $u(\theta_c, o_{c'}) = 0$ for all $c' \in C$ with $c \neq c'$; $u(\theta_v, o_l) = |\Theta|$ for all $l \in L$ with $v(l) = v$; $u(\theta_v, o_l) = 0$ for all $l \in L$ with $v(l) \neq v$; $u(\theta_v, o_c) = 0$ for all $c \in C$. The goal of the automated mechanism design instance is $G = |\Theta| + \frac{|C|-K}{|\Theta|}$, where K is the goal of the MINSAT instance. We show the instances are equivalent. First, suppose there is a solution to the MINSAT instance. Let the assignment of truth values to the variables in this solution be given by the function $f : V \to L$ (where $v(f(v)) = v$ for all $v \in V$). Then, for every $v \in V$, let $o(\theta_v) = o_{f(v)}$ and $\pi(\theta_v) = |\Theta|$. For every $c \in C$, let $o(\theta_c) = o_c$; let $\pi(\theta_c) = |\Theta|+1$ if c is not satisfied in the MINSAT solution, and $\pi(\theta_c) = |\Theta|$ if c is satisfied. It is straightforward to check that the IR constraint is satisfied. We now check that the agent has no incentive to misreport. If the agent's type is some θ_v, then any other report will give it an outcome that is no better, for a payment that is no less, so it has no incentive to misreport. If the agent's type is some θ_c where c is a satisfied clause, again, any other report will give it an outcome that is no better, for a payment that is no less, so it has no incentive to misreport. The final case to check is where the agent's type is some θ_c where c is an unsatisfied clause. In this case, we observe that for none of the types, reporting it leads to an outcome o_l for a literal $l \in c$, precisely because the clause is not satisfied in the MINSAT instance. Because also, no type besides θ_c leads to the outcome o_c, reporting any other type will give an outcome with utility 0, while still forcing a payment of at least $|\Theta|$ from the agent. Clearly the agent is better off reporting truthfully, for a total utility of 0. This establishes that the agent never has an incentive to misreport. Finally, we show that the goal is reached. If s is the number of satisfied clauses in the MINSAT solution (so that $s \leq K$), the expected payment from this mechanism is $\frac{|V||\Theta|+s|\Theta|+(|C|-s)(|\Theta|+1)}{|\Theta|} \geq \frac{|V||\Theta|+K|\Theta|+(|C|-K)(|\Theta|+1)}{|\Theta|} = |\Theta| + \frac{|C|-K}{|\Theta|} = G$. So there is a solution to the automated mechanism design instance.

Now suppose there is a solution to the automated mechanism design instance, given by an outcome function o and a payment function π. First, suppose there is some $v \in V$ such that $o(\theta_v) \notin \{o_{+v}, o_{-v}\}$. Then the utility that the agent derives from the given outcome for this type is 0, and hence, by IR, no payment can be extracted from the agent for this type. Because, again by IR, the maximum payment that can be extracted for any other type is $|\Theta| + 1$, it follows that the maximum expected payment that could be obtained is at most $\frac{(|\Theta|-1)(|\Theta|+1)}{|\Theta|} < |\Theta| < G$, contradicting that this is a solution to the automated mechanism design instance. It follows that in the solution to the automated mechanism design instance, for every $v \in V$, $o(\theta_v) \in \{o_{+v}, o_{-v}\}$. We can interpret this as an assignment of truth values to the variables: v is set to *true* if $o(\theta_v) = o_{+v}$, and to *false* if $o(\theta_v) = o_{-v}$. We claim this assignment is a solution to the MINSAT instance. By the IR constraint, the maximum payment we can extract from any type θ_v is $|\Theta|$. Because there can be no incentives for the agent to report falsely, for any clause c satisfied by the given assignment, the maximum payment we can extract for the corresponding type θ_c is $|\Theta|$. (For if we extracted more from this type, the agent's utility in this case would be less than 1; and if v is the variable satisfying c in the assignment, so that $o(\theta_v) = o_l$ where l occurs in c, then the agent would be better off reporting θ_v instead of the truthful report θ_c, to get an outcome worth $|\Theta| + 1$ to it while having to pay at most $|\Theta|$.) Finally, for any unsatisfied clause c, by the IR constraint, the maximum payment we can extract for the corresponding type θ_c is $|\Theta|+1$. It follows that the expected payment from our mechanism is at most $\frac{V|\Theta|+s|\Theta|+(|C|-s)(|\Theta|+1)}{\Theta}$, where s is the number of satisfied clauses. Because our mechanism achieves the goal, it follows that $\frac{V|\Theta|+s|\Theta|+(|C|-s)(|\Theta|+1)}{\Theta} \geq G$, which by simple algebraic manipulations is equivalent to $s \leq K$. So there is a solution to the MINSAT instance. ■

Finally, we study the case where the designer is self-interested and is not interested in payments made (that is, the objective is some function $g : O \to \mathbb{R}$). In this case, if the designer is allowed to use payments, then the designer can always choose her most preferred outcome by giving the agents an amount large enough to compensate them for the choice of this outcome, thereby not breaking the IR constraint. However, the case where the designer is not allowed to use payments is more complex, as the following theorem shows:

Theorem 40 *The AMD problem for designing deterministic mechanisms without payments is NP-complete, even when the designer is self-interested (but faces an IR constraint), there is only a single agent, and the probability distribution over Θ is uniform. (Membership in NP is guaranteed only if the number of agents is constant.)*

Proof: The proof of membership in NP for a constant number of agents is similar to previous proofs. To show NP-hardness, we reduce an arbitrary MINSAT instance to the following automated mechanism design instance. Let the agent's type set be $\Theta = \{\theta_c : c \in C\} \cup \{\theta_v : v \in V\}$, where C is the set of clauses in the MINSAT instance, and V is the set of variables. Let the probability distribution over these types be uniform. Let the outcome set be $O = \{o_0\} \cup \{o_c : c \in C\} \cup \{o_l : l \in L\} \cup \{o^*\}$, where L is the set of literals, that is, $L = \{+v : v \in V\} \cup \{-v : v \in V\}$. Let the notation $v(l) = v$ denote that v is the variable corresponding to the literal l, that is, $l \in \{+v, -v\}$. Let $l \in c$ denote that the literal l occurs in clause c. Then, let the agent's utility function be given by $u(\theta_c, o_l) = 2$ for all $l \in L$ with $l \in c$; $u(\theta_c, o_l) = -1$ for all $l \in L$ with $l \notin c$; $u(\theta_c, o_c) = 2$; $u(\theta_c, o_{c'}) = -1$ for all $c' \in C$ with $c \neq c'$; $u(\theta_c, o^*) = 1$; $u(\theta_v, o_l) = 1$ for all $l \in L$ with $v(l) = v$; $u(\theta_v, o_l) = -1$ for all $l \in L$ with $v(l) \neq v$; $u(\theta_v, o_c) = -1$ for all $c \in C$; $u(\theta_v, o^*) = -1$. Let the designer's objective function be given by $g(o^*) = |\Theta| + 1$; $g(o_l) = |\Theta|$ for all $l \in L$; $g(o_c) = |\Theta|$ for all $c \in C$. The goal of the automated mechanism design instance is $G = |\Theta| + \frac{|C|-K}{|\Theta|}$, where K is the goal of the MINSAT instance. We show the instances are equivalent. First, suppose there is a solution to the MINSAT instance. Let the assignment of truth values to the variables in this solution be given by the function $f : V \to L$ (where $v(f(v)) = v$ for all $v \in V$). Then, for every $v \in V$, let $o(\theta_v) = o_{f(v)}$. For every $c \in C$ that is satisfied in the MINSAT solution, let

$o(\theta_c) = o_c$; for every unsatisfied $c \in C$, let $o(\theta_c) = o^*$. It is straightforward to check that the IR constraint is satisfied. We now check that the agent has no incentive to misreport. If the agent's type is some θ_v, it is getting the maximum utility for that type, so it has no incentive to misreport. If the agent's type is some θ_c where c is a satisfied clause, again, it is getting the maximum utility for that type, so it has no incentive to misreport. The final case to check is where the agent's type is some θ_c where c is an unsatisfied clause. In this case, we observe that for none of the types, reporting it leads to an outcome o_l for a literal $l \in c$, precisely because the clause is not satisfied in the MINSAT instance. Because also, no type leads to the outcome o_c, there is no outcome that the mechanism ever selects that would give the agent utility greater than 1 for type θ_c, and hence the agent has no incentive to report falsely. This establishes that the agent never has an incentive to misreport. Finally, we show that the goal is reached. If s is the number of satisfied clauses in the MINSAT solution (so that $s \leq K$), then the expected value of the designer's objective function is $\frac{|V||\Theta|+s|\Theta|+(|C|-s)(|\Theta|+1)}{|\Theta|} \geq \frac{|V||\Theta|+K|\Theta|+(|C|-K)(|\Theta|+1)}{|\Theta|} = |\Theta| + \frac{|C|-K}{|\Theta|} = G$. So there is a solution to the automated mechanism design instance.

Now suppose there is a solution to the automated mechanism design instance, given by an outcome function o. First, suppose there is some $v \in V$ such that $o(\theta_v) \notin \{o_{+v}, o_{-v}\}$. The only other outcome that the mechanism is allowed to choose under the IR constraint is o_0. This has an objective value of 0, and because the highest value the objective function ever takes is $|\Theta| + 1$, it follows that the maximum expected value of the objective function that could be obtained is at most $\frac{(|\Theta|-1)(|\Theta|+1)}{|\Theta|} < |\Theta| < G$, contradicting that this is a solution to the automated mechanism design instance. It follows that in the solution to the automated mechanism design instance, for every $v \in V$, $o(\theta_v) \in \{o_{+v}, o_{-v}\}$. We can interpret this as an assignment of truth values to the variables: v is set to *true* if $o(\theta_v) = o_{+v}$, and to *false* if $o(\theta_v) = o_{-v}$. We claim this assignment is a solution to the MINSAT instance. By the above, for any type θ_v, the value of the objective function in this mechanism will be $|\Theta|$. For any clause c satisfied by the given assignment, the value of the objective function in the case where the agent reports type θ_c will be at most $|\Theta|$. (This is because we cannot choose the outcome o^* for such a type, as in this case the agent would have an incentive to report θ_v instead, where v is the variable satisfying c in the assignment (so that $o(\theta_v) = o_l$ where l occurs in c).) Finally, for any unsatisfied clause c, the maximum value the objective function can take in the case where the agent reports type θ_c is $|\Theta| + 1$, simply because this is the largest value the function ever takes. It follows that the expected value of the objective function for our mechanism is at most $\frac{V|\Theta|+s|\Theta|+(|C|-s)(|\Theta|+1)}{\Theta}$, where s is the number of satisfied clauses. Because our mechanism achieves the goal, it follows that $\frac{V|\Theta|+s|\Theta|+(|C|-s)(|\Theta|+1)}{\Theta} \geq G$, which by simple algebraic manipulations is equivalent to $s \leq K$. So there is a solution to the MINSAT instance. ■

6.4 Linear and mixed integer programming approaches

In this section, we describe how the problem of designing an optimal *randomized* mechanism can be cast as a linear programming problem. As we will show, the size of the linear program is exponential only in the number of agents, and because linear programs can be solved in polynomial time [Khachiyan, 1979], this implies that the problem of designing an optimal[6] randomized mechanism is in P if the number of agents is a constant.

Theorem 41 *With a constant number of agents, the optimal randomized mechanism can be found in polynomial time using linear programming, both with and without payments, both for* ex post *and* ex interim *IR, and both for implementation in dominant strategies and for implementation in Bayes-Nash equilibrium—even if the types are correlated (that is, an agent's type tells him something about the other agents' types).*

Proof: Because linear programs can be solved in polynomial time, all we need to show is that the number of variables and equations in our program is polynomial for any constant number of agents—that is, exponential only in n. Throughout, for purposes of determining the size of the linear program, let $T = \max_i\{|\Theta_i|\}$. The variables of our linear program will be the probabilities $(p(\theta_1, \theta_2, \ldots, \theta_n))(o)$ (at most $T^n|O|$ variables) and the payments $\pi_i(\theta_1, \theta_2, \ldots, \theta_n)$ (at most nT^n variables). (We show the linear program for the case where payments are possible; the case without payments is easily obtained from this by simply omitting all the payment variables in the program, or by adding additional constraints forcing the payments to be 0.)

First, we show the IR constraints. For *ex post* IR, we add the following (at most nT^n) constraints to the LP:

- For every $i \in \{1, 2, \ldots, n\}$, and for every $(\theta_1, \theta_2, \ldots, \theta_n) \in \Theta_1 \times \Theta_2 \times \ldots \times \Theta_n$, we add

$$(\sum_{o \in O} (p(\theta_1, \theta_2, \ldots, \theta_n))(o)u(\theta_i, o)) - \pi_i(\theta_1, \theta_2, \ldots, \theta_n) \geq 0.$$

For *ex interim* IR, we add the following (at most nT) constraints to the LP:

- For every $i \in \{1, 2, \ldots, n\}$, for every $\theta_i \in \Theta_i$, we add

$$\sum_{\theta_1, \ldots, \theta_n} \gamma(\theta_1, \ldots, \theta_n|\theta_i)((\sum_{o \in O} (p(\theta_1, \theta_2, \ldots, \theta_n))(o)u(\theta_i, o)) - \pi_i(\theta_1, \theta_2, \ldots, \theta_n)) \geq 0.$$

Now, we show the solution concept constraints. For implementation in dominant strategies, we add the following (at most nT^{n+1}) constraints to the LP:

- For every $i \in \{1, 2, \ldots, n\}$, for every $(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n) \in \Theta_1 \times \Theta_2 \times \ldots \times \Theta_n$, and for every alternative type report $\hat{\theta}_i \in \Theta_i$, we add the constraint

$$(\sum_{o \in O} (p(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n))(o)u(\theta_i, o)) - \pi_i(\theta_1, \theta_2, \ldots, \theta_i, \ldots, \theta_n) \geq$$
$$(\sum_{o \in O} (p(\theta_1, \theta_2, \ldots, \hat{\theta}_i, \ldots, \theta_n))(o)u(\theta_i, o)) - \pi_i(\theta_1, \theta_2, \ldots, \hat{\theta}_i, \ldots, \theta_n).$$

Finally, for implementation in Bayes-Nash equilibrium, we add the following (at most nT^2) constraints to the LP:

[6] Since linear programs allow for an objective, we can search for the optimal mechanism rather than only solve the decision variant (does a mechanism with objective value at least G exist?) of the problem.

• For every $i \in \{1, 2, ..., n\}$, for every $\theta_i \in \Theta_i$, and for every alternative type report $\hat{\theta}_i \in \Theta_i$, we add the constraint

$$\sum_{\theta_1,...,\theta_n} \gamma(\theta_1, ..., \theta_n|\theta_i)((\sum_{o \in O} (p(\theta_1, \theta_2, ..., \theta_i, ..., \theta_n))(o)u(\theta_i, o)) - \pi_i(\theta_1, \theta_2, ..., \theta_i, ..., \theta_n)) \geq$$
$$\sum_{\theta_1,...,\theta_n} \gamma(\theta_1, ..., \theta_n|\theta_i)((\sum_{o \in O} (p(\theta_1, \theta_2, ..., \hat{\theta}_i, ..., \theta_n))(o)u(\theta_i, o)) - \pi_i(\theta_1, \theta_2, ..., \hat{\theta}_i, ..., \theta_n)).$$

All that is left to do is to give the expression the designer is seeking to maximize, which is:

• $\sum_{\theta_1,...,\theta_n} \gamma(\theta_1, ..., \theta_n)((\sum_{o \in O} (p(\theta_1, \theta_2, ..., \theta_i, ..., \theta_n))(o)g(o)) + \sum_{i=1}^{n} \pi_i(\theta_1, \theta_2, ..., \theta_n)).$

As we indicated, the number of variables and constraints is exponential only in n, and hence the linear program is of polynomial size for constant numbers of agents. Thus the problem is solvable in polynomial time. ■

By forcing all the probability variables to be either 0 or 1, thereby changing the linear program into a mixed integer program, we can solve for the optimal deterministic mechanism. (We note that solving a mixed integer program is NP-complete.) Our general-purpose automated mechanism design solver consists of running CPLEX (a commercial solver) on these linear or mixed integer programs.

6.5 Initial applications

The only example application that we have seen so far is the divorce settlement setting from Section 6.2. In this section, we apply AMD to some domains that are more commonly studied in mechanism design: optimal auctions and mechanisms for public goods.

6.5.1 Optimal auctions

In this subsection we show how AMD can be used to design auctions that maximize the seller's expected revenue (so-called *optimal* auctions). In many auction settings, the seller would like to design the rules of the auction to accomplish this. However, as we briefly mentioned in Chapter 4, in general settings this is a known difficult mechanism design problem; for one, it is much more difficult than designing a mechanism that allocates the goods efficiently (among bidders with quasi-linear preferences, *ex post* efficiency and IR can be accomplished in dominant strategies using the VCG mechanism).

We first study auctioning off a single good, and show that AMD reinvents a known landmark optimal auction mechanism, the Myerson auction, for the specific instance that we study. (Of course, it does not derive the general form of the Myerson auction, which can be applied to any single-item instance: AMD necessarily only solves the instance at hand.) We then move to multi-item (combinatorial) auctions, where the optimal auction has been unknown in the literature to date. We show that AMD can design optimal auctions for this setting as well.

An optimal 2-bidder, 1-item auction

We first show how automated mechanism design can rederive known results in optimal single-item auction design. Say there is one item for sale. The auctioneer can award it to any bidder, or not award it (and say the auctioneer's valuation for the good is 0). There are two bidders, 1 and 2. For each of them, their distribution of valuations is uniform over $\{0, 0.25, 0.5, 0.75, 1\}$.

In designing the auction automatically, we required ex-interim IR and implementation in Bayes-Nash equilibrium. Randomization was allowed (although in this setting, it turned out that the probabilities were all 0 or 1). The allocation rule of the mechanism generated by the solver is as follows. If both bid below 0.5, do not award the item; otherwise, give the item to the highest bidder (a specific one of them in the case of a tie). This is effectively[7] the celebrated *Myerson auction* [Myerson, 1981] (although the Myerson auction was originally derived for a continuous valuation space). So, AMD quickly reinvented a landmark mechanism from 1981. (Although it should be noted that it invented it for a special case, and did not derive the general characterization. Also, it did not invent the *question* of optimal auction design.)

Multi-item (combinatorial) auctions

We now move to combinatorial auctions where there are multiple goods for sale. The design of a mechanism for this setting that maximizes the seller's expected revenue is a recognized open research problem [Avery and Hendershott, 2000; Armstrong, 2000; Vohra, 2001]. The problem is open even if there are only two goods for sale. (The two-good case with a very special form of complementarity and no substitutability has been solved recently [Armstrong, 2000].) We show that AMD can be used to generate optimal combinatorial auctions.

In our first combinatorial auction example, two items, A and B, are for sale. The auctioneer can award each item to any bidder, or not award it (and the auctioneer's valuation is 0). There are two bidders, 1 and 2, each of whom has four possible, equally likely types: LL, HL, LH, and HH. The type indicates whether each item is strongly desired or not; for instance, the type HL indicates that the bidder strongly desires the first item, but not the second. Getting an item that is strongly desired gives utility 2; getting one that is not strongly desired gives utility 1. The utilities derived from the items are simply additive (no substitution or complementarity effects), with the exception of the case where the bidder has the type HH. In this case there is a complementarity bonus of 2 for getting both items (thus, the total utility of getting both items is 6). (One way to interpret this is as follows: a bidder will sell off any item it wins and does not strongly desire, on a market where it is a price taker, so that there are no substitution or complementarity effects with such an item.)

In designing the auction, we required ex-interim IR and implementation in Bayes-Nash equilibrium. Randomization was allowed (although in this setting, it turned out that the probabilities were all 0 or 1). The objective to maximize was the expected payments from the bidders to the seller. The mechanism generated by the solver has the following allocation rule: 1. If one bidder bid LL, then the other bidder gets all the items he bid high on, and all the other items (that both bid low on) are not awarded. 2. If exactly one bidder bid HH, that bidder gets both items. If both bid HH, bidder 1 gets both items. 3. If both bidders bid high on only one item, and they did not bid high on

[7] The payment rule generated is slightly different, because CPLEX chooses to distribute the payments slightly differently across different type vectors.

the same item, each bidder gets his preferred item. 4. If both bidders bid high on only one item, and they bid high on the same item, bidder 2 gets the preferred item, and bidder 1 gets the other item.

	LL	LH	HL	HH
LL	0, 0	0, 2	2, 0	2, 2
LH	0, 1	1, 2	2, 1	2, 2
HL	1, 0	1, 2	2, 1	2, 2
HH	1, 1	1, 1	1, 1	1, 1

The allocation rule in the optimal combinatorial auction. The row indicates bidder 1's type, the column bidder 2's type. i, j indicates that item A goes to bidder i, and item B to bidder j. (0 means the item is not awarded to anyone.)

It is interesting to observe that suboptimal allocations occur only when one bidder bids LL and the other other does not bid HH. All the inefficiency stems from not awarding items, never from allocating items to a suboptimal bidder.

The expected revenue from the mechanism is 3.9375. For comparison, the expected revenue from the VCG mechanism is only 2.6875. It is interesting to view this in light of a recent result that the VCG mechanism is asymptotically (as the number of bidders goes to infinity) optimal in multi-item auctions, that is, it maximizes revenue in the limit [Monderer and Tennenholtz, 1999].[8] Apparently the auction will need to get much bigger (have more bidders) before no significant fraction of the revenue is lost by using the VCG mechanism. (Of course, this is only a single instance—future research may determine how much revenue is typically lost by the VCG mechanism for instances of this size, as well as determine how this changes when the instances become somewhat larger.)

We now move on to designing a bigger auction, again with 2 items, but now with 3 bidders (for a total of 16 possible allocations of items) and a bigger type space. Again, the bidders can have a high or low type for each item, resulting in a utility for that item alone of 3 or 1, respectively; part of their type now also includes whether the items have complementarity or substitutability to them, resulting in a total of 8 types per bidder—that is, $8^3 = 512$ type vectors (possible joint preference revelations by the bidders). In the case where the items have substitutability, the utility of getting both items is the sum of the items' individual values, minus 0.2 times the value of the lesser valued item.[9] In the case of complementarity, 0.2 times the value of the lesser-valued item is *added*.

This is the only instance in this section where CPLEX took more than 0.00 seconds to solve the instance. (It took 5.90 seconds.) The optimal auction generated has an expected revenue of 5.434. The allocation rule generated (an 8x8x8 table) is too large to present, but we point out some interesting properties of the optimal auction generated nonetheless:

1. Sometimes, items are again not awarded, for example, when two bidders report a low valuation for both items and the remaining bidder does not report a high valuation on both items;

[8]This result is particularly easy to prove in a discretized setting such as the one we are considering. The following sketches the proof. As the number of bidders grows, it becomes increasingly likely that for each winning bid, there is another submitted bid that is exactly identical, but not accepted. If this is the case, the VCG payment for the winning bid is exactly the value of that bid, and thus the VCG mechanism extracts the maximum possible payment. (This is also roughly the line of reasoning taken in the more general result [Monderer and Tennenholtz, 1999].)

[9]Subtracting a fraction from the lesser valued item guarantees free disposal, *i.e.* additional items cannot make a bidder worse off.

2. Randomization now *does* occur, for instance sometimes (but not always) when one item is valued lowly by everyone and two of the three value the other item highly (the randomization is over which of the two gets the desired item);

3. The optimal auction takes the complementarity and substitutability into account, for instance by doing the following. When one bidder bids high on both items and the other two each bid high on one item (not the same one), then the mechanism awards the items to the first bidder if that bidder revealed complementarity, but to the other bidders if the first bidder revealed substitutability. (Each one gets his/her desired item.)

It turns out, however, that the optimal deterministic mechanism generated for this instance has the same expected revenue (5.434). Thus, one may wonder if randomization is ever necessary to create optimal combinatorial auctions. The following example shows that there are indeed instances of the optimal combinatorial auction design problem where randomized mechanisms can perform strictly better than any deterministic mechanism.

In this example, there are two items A and B for sale, and there is only a single bidder (so that it does not matter which solution concept and which IR notion we use). There are three types: type α (occurring with probability 0.3) indicates that the bidder has a utility of 1 for receiving either $\{A\}$ or $\{A, B\}$, and 0 otherwise; type β (occurring with probability 0.3) indicates that the bidder has a utility of 1 for receiving either $\{B\}$ or $\{A, B\}$, and 0 otherwise; and type $\alpha\beta$ (occurring with probability 0.4) indicates that the bidder has a utility of 0.75 for receiving either $\{A\}$, $\{B\}$, or $\{A, B\}$, and 0 otherwise.

The optimal randomized mechanism generated by the solver allocates $\{A\}$ to the bidder if the bidder reports α; $\{B\}$ if the bidder reports β; and $\{A\}$ with probability 0.75, and $\{B\}$ with probability 0.25, if the bidder reports $\alpha\beta$. The payment in each case is the agent's entire valuation (1 for types α and β, and 0.75 for type $\alpha\beta$). The resulting expected revenue is 0.9.

By contrast, the optimal deterministic mechanism generated by the solver allocates $\{A\}$ to the bidder if the bidder reports α; $\{A, B\}$ if the bidder reports β; and $\{A\}$ if the bidder reports $\alpha\beta$. The payment is 0.75 for type α, 1 for type β, and 0.75 for type $\alpha\beta$. The resulting expected revenue is 0.825.

This example demonstrates how AMD can be used to disprove conjectures in mechanism design: the conjecture that one can restrict attention to deterministic mechanisms in the design of optimal combinatorial auctions is disproved by an example where the optimal randomized mechanism produced by the solver is strictly better than the optimal deterministic one.

6.5.2 Public goods problems

As another example application domain, we now turn to public good problems. A *public good* is a good from which many agents can benefit simultaneously; the good is said to be *nonexcludable* if we cannot prevent any agents from obtaining this benefit, given that we produce the good. Examples of nonexcludable public goods include clean air, national defense, pure research, *etc.*

A typical mechanism design problem that arises is that a certain amount of money is required to construct or acquire the (nonexcludable) public good, and this money must be collected from the agents that may benefit from it. However, how much the good is worth to each agent is information that is private to that agent, and we cannot obtain a larger payment from an agent than what the agent claims the good is worth to him (an individual rationality constraint). This leads to the potential problem of *free riders* who report very low values for the good in the hope that other agents will value the good enough for it to still be produced. Formally, every agent i has a type v_i (the value of the good to him), and the mechanism decides whether the good is produced, as well as each agent's payment π_i. If the good is produced, we must have $\sum_i \pi_i \geq c$, where c is the cost of producing the good. Results similar to the Myerson-Satterthwaite impossibility theorem can be proved here to show that even in quite simple settings, there is no mechanism that is *ex post* efficient, *ex post* budget balanced, *ex-interim* individually rational, and BNE incentive-compatible. In fact, Theorem 36 from Chapter 5 shows exactly this.

The advantage of applying AMD in this setting is that we do not desire to design a mechanism for general (quasilinear) preferences, but merely for the specific mechanism design problem instance at hand. In some settings this may allow one to circumvent the impossibility entirely, and in all settings it minimizes the pain entailed by the impossibility.

Building a bridge

Two agents are deciding whether to build a good that will benefit both (say, a bridge). The bridge, if it is to be built, must be financed by the payments made by the agents. Building the bridge will cost 6. The agents have the following type distribution: with probability .4, agent 1 will have a low type and value the bridge at 1. With probability .6, agent 1 will have a high type and value the bridge at 10. Agent 2 has a low type with probability .6 and value the bridge at 2; with probability .4, agent 2 will have a high type and value the bridge at 11. (Thus, agent 2 cares for the bridge more in both cases, but agent 1 is more likely to have a high type.)

We used AMD to design a randomized, dominant-strategies incentive compatible, *ex post* IR mechanism that is as efficient as possible—*taking into account unnecessary payments ("money burning") as a loss in efficiency*. The optimal mechanism generated by our AMD implementation has the following outcome function (here the entries of the matrix indicate the probability of building the bridge in each case):

	Low	High
Low	0	.67
High	1	1

The payment function is as follows (here a, b gives the payments of agents 1 and 2, respectively):

	Low	High
Low	0, 0	.67, 3.33
High	4, 2	4, 2

The payments in the case where agent 1 bids low but agent 2 bids high are the *expected payments* (as we argued before, risk-neutral agents only care about this); the agents will need to pay more than this when the good is actually built, but can pay less when it is not. (The constraints on the expected payments in the linear program are set so that the good can always be afforded when it is built.)

It is easy to see that no money is burned: all the money the agents pay goes towards building the bridge. However, we do not always build the bridge when this is socially optimal—namely, when the second agent has a high type (which is enough to justify building the bridge) we do not always build the bridge.

If we relax our solution concept to implementation in Bayes-Nash equilibrium, however, we get a mechanism with the following outcome function:

	Low	High
Low	0	1
High	1	1

The payment function is now as follows:

	Low	High
Low	0, 0	0, 6
High	4, 2	.67, 5.33

Again, no money is burned, but now also, the optimal outcome is always chosen. Thus, with Bayes-Nash equilibrium incentive compatibility, our mechanism achieves everything we hope for in this instance—even though the impossibility result shows that this is not possible for *all* instances.

Building a bridge and/or a boat

Now let us move to the more complex public goods setting where two goods could be built: a bridge and a boat. There are 4 different outcomes corresponding to which goods are built: None, Boat, Bridge, Boat and Bridge. The boat costs 1 to build, the bridge 2, and building both thus costs 3.

The two agents each have one of four different types: None, Boat Only, Bridge Only, Boat or Bridge. These types indicate which of the two possible goods would be helpful to the agent (for instance, maybe one agent would only be helped by a bridge because this agent wants to take the car to work, which will not fit on the boat). All types are equally likely; if something is built which is useful to a agent (given that agent's type), the agent gets a utility of 2, otherwise 0.

We used AMD to design the optimal randomized dominant-strategy mechanism that is ex post IR, and as ex post efficient as possible—taking into account money burning as a loss in efficiency. The mechanism has the following outcome function, where a vector (a, b, c, d) indicates the probabilities for None, Boat, Bridge, Boat and Bridge, respectively.

	None	Boat	Bridge	Either
None	(1,0,0,0)	(0,1,0,0)	(1,0,0,0)	(0,1,0,0)
Boat	(.5,.5,0.0)	(0,1,0,0)	(0,.5,0,.5)	(0,1,0,0)
Bridge	(1,0,0,0)	(0,1,0,0)	(0,0,1,0)	(0,0,1,0)
Either	(.5,.5,0.0)	(0,1,0,0)	(0,0,1,0)	(0,1,0,0)

The (expected) payment function is as follows:

	None	Boat	Bridge	Either
None	0,0	0,1	0,0	0,1
Boat	.5,0	0,1	1,1	0,1
Bridge	0,0	0,1	1,1	1,1
Either	.5,0	0,1	1,1	0,1

Again, no money is burned, but we do not always build the public goods that are socially optimal—for example, sometimes nothing is built although the boat would have been useful to someone.

6.6 Scalability experiments

To assess the scalability of the automated mechanism design approach in general, we generated random instances of the automated mechanism design problem. Each agent, for each of its types, received a utility for each outcome that was uniformly randomly chosen from the integers $0, 1, 2, \ldots, 99$. (All random draws were independent.) Real-world automated mechanism design instances are likely to be more structured than this (for example, in allocation problems, if one agent is happy with an outcome, this is because it was allocated a certain item that it wanted, and thus other agents who wanted the item will be less happy); such special structure can typically be taken advantage of in computing the optimal mechanism, even by nonspecialized algorithms. For instance, a random instance with 3 agents, 16 outcomes, 8 types per agent, with payment maximization as its goal, ex-interim IR, implementation in Bayes-Nash equilibrium, where randomization is allowed, takes 14.28 seconds to solve on average in our implementation. The time required to compute the last optimal combinatorial auction from Section 6.5, which had exactly the same parameters (but much more structure in the utility functions), compares (somewhat) favorably to this at 5.90 seconds.

We are now ready to present the scalability results. For every one of our experiments, we consider both implementation in dominant strategies and implementation in Bayes-Nash equilibrium. We also consider both the problem of designing a deterministic mechanism and that of designing a randomized mechanism. All the other variables that are not under discussion in a particular experiment are fixed at a default value (4 agents, 4 outcomes, 4 types per agent, no IR constraint, no payments, social welfare is the objective); these default values are chosen to make the problem hard enough for its runtime to be interesting. Experiments taking longer than 6 hours were cancelled, as well as experiments where the LP size was greater than 400MB. CPLEX does not provide runtime information more detailed than centiseconds, which is why we do not give the results with a constant number of significant digits, but rather all the digits we have.

The next table shows that the runtime increases fairly sharply with the number of agents. Also (as will be confirmed by all the later experiments), implementation in dominant strategies is harder than implementation in BNE, and designing deterministic mechanisms is harder than designing randomized mechanisms. (The latter part is consistent with the transition from NP-completeness to solvability in polynomial time by allowing for randomness in the mechanism (Sections 6.3 and 6.4).)

#agents	D/DSE	R/DSE	D/BNE	R/BNE
2	.02	.00	.00	.00
3	.04	.00	.05	.01
4	8.32	1.32	1.68	.06
5	709.85	48.19	10.47	.52

The time (in seconds) required to solve randomly generated AMD instances for different numbers of agents, for deterministic (D) or randomized (R) mechanisms, with implementation in dominant strategies (DSE) or Bayes-Nash equilibrium (BNE). All experiments had 4 outcomes and 4 types per agent, required no IR constraint, did not allow for payments, and had social welfare as the objective.

The next table shows that the runtime tends to increase with the number of outcomes, but not at all sharply.

#outcomes	D/DSE	R/DSE	D/BNE	R/BNE
2	.07	.07	.04	.03
3	.36	.08	.46	.05
4	8.32	1.32	1.68	.06
5	10.91	.59	.69	.07

The next table shows that the runtime increases fairly sharply with the number of types per agent.

#types	D/DSE	R/DSE	D/BNE	R/BNE
2	.00	.00	.00	.00
3	.04	.01	.30	.01
4	8.32	1.32	1.68	.06
5	563.73	14.33	36.60	.21

Because the R/BNE case scales reasonably well in each setting, we increased the numbers of agents, outcomes, and types further for this case to test the limits of our implementation. Our initial implementation requires the linear program to be written out explicitly, and thus space eventually became the bottleneck for scaling in agents and types. ("*" indicates that the LP size exceeded 400MB.) Mature techniques exist for linear programming when the LP is too large to write down, and future implementations could make use of these techniques.

#	agents	outcomes	types
6	4.39	.07	.88
7	33.32	.07	1.91
8	*	.09	4.52
10	*	.11	22.05
12	*	.13	67.74
14	*	.13	*
100	*	1.56	*

The next table shows that the impact of IR constraints on runtime is entirely negligible.

IR constraint	D/DSE	R/DSE	D/BNE	R/BNE
None	8.32	1.32	1.68	.06
Ex post	8.20	1.38	1.67	.12
Ex interim	8.11	1.42	1.65	.11

The next table studies the effects of allowing for payments and changing the objective. Allowing for payments (without taking the payments into account) in social welfare maximization reduces the runtime. This appears consistent with the fact that for this setting, a general (and easy-to-compute) mechanism exists that always obtains the maximum social welfare—the VCG mechanism. However, this speedup disappears when we start taking the payments into account. Interestingly, payment maximization appears to be much harder than social welfare maximization. In particular, in one case (designing a deterministic mechanism without randomization), an optimal mechanism had not been constructed after 6 hours!

Objective	D/DSE	R/DSE	D/BNE	R/BNE
SW (1)	8.20	1.38	1.67	.12
SW (2)	.41	.14	.92	.10
SW (3)	7.98	.51	4.44	.10
π	-	1.89	84.66	3.47

SW=social welfare (1) without payments, (2) with payments that are not taken into account in social welfare calculations, (3) with payments that are taken into account in social welfare calculations; π=payment maximization.

The sizes of the instances that we can solve may not appear very impressive when compared with the sizes of (for instance) combinatorial auctions currently being studied in the literature. While this is certainly true, we emphasize that 1. We are studying a much more difficult problem than the auction clearing problem: we are *designing* the mechanism, rather than executing it; 2. AMD is still in its infancy, and it is likely that future (possibly approximate) approaches will scale to much larger instances; and 3. Although many real-world instances are very large, there are also many small ones. Moreover, the "small" instances may concern equally large dollar values as the large ones. For example, selling two masterpieces by Picasso in a combinatorial auction could create revenue comparable to that of selling a million plane tickets in a combinatorial auction.

6.7 An algorithm for single-agent settings

The definitions from Section 6.1 simplify significantly when applied to the setting where a deterministic mechanism without payments must be designed, with a single type-reporting agent. For one, the different possible IC (truthfulness) constraints differ only in what a type-reporting agent is assumed to know about other type-reporting agents' preferences and reports. Because in this setting, there are no other type-reporting agents, the different IC constraints coincide. The same is true for the IR (participation) constraints. We also do not need distributions over outcomes, or payment functions. The result is the following formal definition for our special case.

Definition 27 (AUTOMATED-MECHANISM-DESIGN (AMD)) *We are given a set of outcomes O, and a set of types Θ for the agent together with a probability distribution p over these types. Additionally we are given a utility function for the agent, $u : \Theta \times O \to \mathbb{R}$, and an objective function for the designer, $g : \Theta \times O \to \mathbb{R}$. We are asked to find an outcome function $o : \Theta \to O$ (a deterministic mechanism without payments) such that:*

1. *For every $\theta, \hat{\theta} \in \Theta$, $u(\theta, o(\theta)) \geq u(\theta, o(\hat{\theta}))$ (the IC constraint).*
2. *If there is an IR constraint, for every $\theta \in \Theta$, $u(\theta, o(\theta)) \geq 0$. (In this case there typically also is a* default outcome *o_0 with $u(\theta, o_0) = 0$ for all $\theta \in \Theta$.[10])*
3. *Subject to the previous constraints, the mechanism maximizes $\sum_{\theta \in \Theta} p(\theta) g(\theta, o(\theta))$.*

We note that by Theorem 37, even this specialized problem is NP-complete (even without the IR constraint, and even when the objective function is a social welfare function including another agent that does not report a type).

6.7.1 Application: One-on-one bartering

As an interlude, we first present an application. Consider the situation where two agents each have an initial endowment of goods. Each agent has a valuation for every subset of the m goods that the agents have together. It is possible that both agents can become better off as a result of trade. Suppose, however, that the agents cannot make any form of payment; all they can do is swap goods. This is known as *bartering*. Additionally, suppose that one agent (agent 1) is in the position of dictating the rules of the bartering process. Agent 1 can credibly say to agent 2, "we will barter by my rules, or not at all." This places agent 1 in the position of the mechanism designer, and corresponds to the following AMD problem. The set of outcomes is the set of all allocations of the goods (there are 2^m of them). Agent 2 is to report his preferences over the goods (the valuation that agent has for each subset), and on the basis of this report an outcome is chosen. This outcome function, which is selected by agent 1 beforehand, must be incentive compatible so that agent 2 has no incentive to misreport. Also, it must be individually rational, or agent 2 simply will not trade.[11] Under these constraints, agent 1 wishes to make the expected value of her own allocation under the mechanism as large as possible. The revelation principle justifies that restricting agent 1 to this approach comes at no loss to that agent.

Automatically generated mechanisms for this setting are likely to be useful in barter-based electronic marketplaces, such as mybarterclub.com, Recipco, and National Trade Banc.

We now return to computational aspects, but we will readdress the bartering problem in our experiments. We will postpone dealing with IR constraints for a few subections, and then return to this.

[10]We can set the utility of the default outcome to 0 without loss of generality, by normalizing the utility function. (From a decision-theoretic point of view it does not matter how utilities compare across types, because the agent always knows her own type and will not take utilities for other types into account in making any decision.)

[11]If agent 1 actually wants to make the rules so that there is no trade for a certain type report, she can simply make the original allocation the outcome for this type report; so there is no loss to agent 1 in designing the outcome function in such a way that agent 2 always wishes to participate in the mechanism.

6.7.2 Search over subsets of outcomes

In this subsection, we associate with each subset of outcomes a truthful mechanism for that set of outcomes; we then show that for some subset of outcomes, the truthful mechanism associated with that subset of outcomes is an optimal mechanism for the setting. Because the mechanism associated with a subset of outcomes is easy to compute, we can search over subsets of outcomes (of which there are $2^{|O|}$) rather than over all possible outcome functions (of which there are $|O|^{|\Theta|}$).[12]

We first define the outcome function (mechanism) o_X associated with a particular subset of the outcomes.

Definition 28 *For a given subset $X \subseteq O$, let $o_X(\theta)$ be (the lowest-indexed element of) $\arg\max_{\{o \in X : (\forall o' \in X) u(\theta,o) \geq u(\theta,o')\}} g(\theta, o)$. Let $v(X)$ be given by $\sum\limits_{\theta \in \Theta} p(\theta) g(\theta, o_X(\theta))$.*

Intuitively, $o_X(\theta)$ is the outcome we wish to pick for type θ, if we (somehow) know that the set of other outcomes used in the mechanism is exactly X, and we wish to pick an outcome from X as well. $v(X)$ is the expected value of the objective function for the mechanism o_X, presuming that the agent reports truthfully. The next lemma shows that indeed, the agent has no incentive to report falsely.

Lemma 19 *For all $X \subseteq O$, o_X is truthful. (Thus, $v(X)$ is indeed the expected value of the objective function for it.)*

Proof: For any pair of types θ_1, θ_2, we have that $o_X(\theta_2) \in X$ because all outcomes ever chosen by o_X are in X; and thus that $u(\theta_1, o_X(\theta_1)) \geq u(\theta_1, o_X(\theta_2))$, because for any θ, $o_X(\theta)$ maximizes $u(\theta, \cdot)$ among outcomes $o \in X$. ■

The next lemma shows that for any subset X, the mechanism o_X dominates all mechanisms that use exactly the outcomes in X.

Lemma 20 *For any $X \subseteq O$, suppose that $o : \Theta \to X$ is a truthful mechanism making use only of outcomes in X, but using each outcome in X at least once—that is, $o(\Theta) = X$. Let its expected value of the objective function be $v_o = \sum\limits_{\theta \in \Theta} p(\theta) g(\theta, o(\theta))$. Then $v(X) \geq v_o$.*

Proof: For any $\theta \in \Theta$, we must have that for any $o \in X$, $u(\theta, o(\theta)) \geq u(\theta, o)$—because there exists some $\theta' \in \Theta$ such that $o(\theta') = o$, and thus the agent can guarantee herself at least utility $u(\theta, o)$ by reporting θ'. But $o_X(\theta)$ maximizes $g(\theta, \cdot)$ among such outcomes. Thus, $g(\theta, o_X(\theta)) \geq g(\theta, o(\theta))$. It follows that $v(X) = \sum\limits_{\theta \in \Theta} p(\theta) g(\theta, o_X(\theta)) \geq \sum\limits_{\theta \in \Theta} p(\theta) g(\theta, o(\theta)) = v_o$. ■

It is not necessarily the case that $v(X) = v_o$ for some truthful o making use of all outcomes in X; for instance, there could be some outcome in X that has both a very low utility value and a very low

[12] In the case where $|O|$ is bigger than $|\Theta|$, we can restrict ourselves to outcome subsets of size at most $|\Theta|$, making our approach still more efficient than the straightforward brute search approach. For simplicity of presentation, in this section we will focus on settings where $|\Theta| > |O|$ (as is commonly the case).

objective value. Then o_X will not use this outcome, and thereby have a higher expected value of the objective function than any mechanism that does use it.

We are now ready to present the main theorem of this subsection, which states that the best o_X is indeed an optimal mechanism.

Theorem 42 $\max_{X \subseteq O} v(X)$ *is the maximum expected value of the objective over* all *mechanisms (that are deterministic and use no payments).* o_X *is an optimal mechanism (among mechanisms that are deterministic and use no payments) if* $X \in \arg\max_{X \subseteq O} v(X)$.

Proof: Consider an optimal truthful mechanism o,[13] and let X be the set of all outcomes it uses ($X = o(\Theta)$). By Lemma 19, o_X is truthful and $v(X)$ is the expected value of the objective function for it. By Lemma 20, we have $v(X) \geq v_o$ where v_o is the expected value of the objective function for o. ■

6.7.3 A heuristic and its admissibility

We now proceed to define an outcome function that is associated with two disjoint subsets X and Y of the outcomes; we will use this outcome function to compute an admissible heuristic for our search problem. The interpretation is as follows. In the process of constructing a mechanism of the kind described in the previous subsection, we successively label each outcome as "in" or "out", depending on whether we wish to include this outcome in the set that the eventual mechanism is associated with. X consists of the outcomes that we have already decided are "in"; Y consists of the outcomes that we have already decided are "out". To get an optimistic view of the mechanisms we may eventually arrive at from here, we assign to each type the outcome in $O - Y$ that gives us the highest objective value for that type (the mechanisms certainly will not use any outcome in Y), under the constraint that this outcome will make that type at least as well off as any outcome in X (because we have already decided that these are certainly "in", so we know this constraint must apply).

Definition 29 *For given subsets* $X, Y \subseteq O$*, let* $o_{X,Y}(\theta)$ *be (the lowest-indexed element of)* $\arg\max_{o \in O - Y : (\forall o' \in X) u(\theta, o) \geq u(\theta, o')} g(\theta, o)$*. Let* $v(X, Y)$ *be given by* $\sum_{\theta \in \Theta} p(\theta) g(\theta, o_{X,Y}(\theta))$

Outcome functions of this type do *not* necessarily constitute truthful mechanisms. (For instance, if X and Y are both the empty set, then $o_{X,Y}$ will simply choose the objective-maximizing outcome for each type.) Nevertheless, because we are merely trying to obtain an optimistic estimate, we compute $v(X, Y)$ as before, presuming the agents will report truthfully. The following theorem shows that $v(X, Y)$ is indeed admissible.

Theorem 43 *For any subsets* $X, Y \subseteq O$*, for any* $Z \subseteq O - X - Y$*, for any* $\theta \in \Theta$*, we have* $g(\theta, o_{X,Y}(\theta)) \geq g(\theta, o_{X \cup Z}(\theta))$*; and* $v(X, Y) \geq v(X \cup Z)$.

[13] Which, by the revelation principle, is an optimal mechanism.

Proof: Using the facts that $X \subseteq X \cup Z$ and $X \cup Z \subseteq O - Y$, we can conclude that $\{o \in X \cup Z : (\forall o' \in X \cup Z) u(\theta, o) \geq u(\theta, o')\} \subseteq \{o \in X \cup Z : (\forall o' \in X) u(\theta, o) \geq u(\theta, o')\} \subseteq \{o \in O - Y : (\forall o' \in X) u(\theta, o) \geq u(\theta, o)\}$. It follows that $g(\theta, o_{X,Y}(\theta)) = \max_{o \in O-Y:(\forall o' \in X) u(\theta,o) \geq u(\theta,o')} g(\theta, o)$ $\geq \max_{o \in X \cup Z:(\forall o' \in X \cup Z) u(\theta,o) \geq u(\theta,o')} g(\theta, o) = g(\theta, o_{X \cup Z}(\theta))$. Thus $v(X, Y) = \sum_{\theta \in \Theta} p(\theta) g(\theta, o_{X,Y}(\theta)) \geq \sum_{\theta \in \Theta} p(\theta) g(\theta, o_{X \cup Z}(\theta)) = v(X \cup Z)$. ■

The following theorem shows that conveniently, at a leaf node, where we have decided for every outcome whether it is in or out, the heuristic value coincides with the value of that outcome set.

Theorem 44 *For any $X \subseteq O, \theta \in \Theta$, we have $o_{X,O-X}(\theta) = o_X(\theta)$ and $v(X, O - X) = v(X)$.*

Proof: Immediate using $O - (O - X) = X$. ■

6.7.4 The algorithm

We are now ready to present the algorithm.

Basic structure

We first summarize the backbone of our algorithm. A node in our search space is defined by a set of outcomes that are definitely "in" (X)[14] and a set of outcomes that are definitely out (Y). For a node at depth d, $X \cup Y$ always constitutes the first d outcomes for some fixed order of the outcomes; thus, a node has two children, one where the next outcome is added to X, and one where it is added to Y. The expansion order is fixed at putting the node in X first, and then in Y. The heuristic value (bound) of a node is given by $v(X, Y)$, as described above.

We can now simply apply A*; this, however, quickly fills up the available memory, so we resort to more space-efficient methods. We first present branch-and-bound depth-first search (expanding every node that still has a chance of leading to a better solution than the best one so far, in depth-first order) for our setting, and then IDA* (in which we maintain a target objective value, and do not expand nodes that do not have a chance of reaching this value; if we fail to find a solution, we decrease the target value and try again). (An overview of search methods such as these can be found in Russell and Norvig [2003].)

[14]We emphasize that this does *not* mean that the outcome will definitely be used by the mechanism corresponding to any descendant leaf node; rather, this outcome *may* be used by any descendant leaf node; and for any descendant leaf node, in the mechanism associated with this node, any type must receive an outcome at least as good to it as this one.

In the following, v is the heuristic for the current node. d is the depth of the current node. ω (a global variable) is the number of outcomes. CB (another global) is the outcome set corresponding to the best mechanism found so far. L (another global) is the expected value of the objective function for the best mechanism we have found so far. o_i is outcome i. The other variables are as described above.

```
BRANCH-AND-BOUND-DFS()
 CB := NULL
 L := −∞
 SEARCH1({}, {}, 0, 1)
 return CB

SEARCH1(X, Y, v, d)
 if d = ω + 1
  CB = X
  L = v
 else
  if v(X ∪ {o_d}, Y) > L
   SEARCH1(X ∪ {o_d}, Y, v(X ∪ {o_d}, Y), d + 1)
  if v(X, Y ∪ {o_d}) > L
   SEARCH1(X, Y ∪ {o_d}, v(X, Y ∪ {o_d}), d + 1)
```

Our implementation of IDA* is similar, except we do not initialize L to $-\infty$. Rather, we initialize it to some high value, and decrease it every time we fail to find a solution—either to a fraction of itself, or to the highest value that is still feasible (whichever is *less*). This also requires us to keep track of the highest value still feasible (given by HF, another global variable), so that we have to modify the search call slightly.

```
IDA*()
 CB := NULL
 L := initial-limit
 while CB = NULL
  HF := −∞
  SEARCH2({}, {}, 0, 1)
  L := min{HF, fraction·L}
 return CB

SEARCH2(X, Y, v, d)
 if d = ω + 1
  CB = X
  L = v
 else
  if v(X ∪ {o_d}, Y) > L
   SEARCH2(X ∪ {o_d}, Y, v(X ∪ {o_d}, Y), d + 1)
  else if v(X ∪ {o_d}, Y) > HF
   HF := v(X ∪ {o_d}, Y)
  if v(X, Y ∪ {o_d}) > L
   SEARCH2(X, Y ∪ {o_d}, v(X, Y ∪ {o_d}), d + 1)
  else if v(X, Y ∪ {o_d}) > HF
   HF := v(X, Y ∪ {o_d})
```

Efficiently updating the heuristic

Rather than computing the heuristic anew each time, it can be computed much more quickly from information used for computing the heuristic at the parent node. For instance, when adding an outcome o to X, we will not have to change $o_{X,Y}(\theta)$ unless $u(\theta, o) > u(\theta, o_{X,Y}(\theta))$. As another example, when adding an outcome o to Y, we will not have to change $o_{X,Y}(\theta)$ unless $o_{X,Y}(\theta) = o$. In addition to this, maintaining appropriate data structures (such as a list of the outcomes sorted by objective value for a given type) allows us to quickly find the new outcome when we do need to make a change.

6.7.5 Individual rationality

We now show how to deal with an individual rationality constraint in this setting.

Theorem 45 *o_X is individually rational if and only if for every $\theta \in \Theta$, there is some $o \in X$ such that $u(\theta, o) \geq 0$.*

Proof: If for some $\theta \in \Theta$, there is no $o \in X$ such that $u(\theta, o) \geq 0$, o_X cannot give the agent nonnegative utility for type θ because o_X uses only outcomes from X; so it is not individually rational. On the other hand, if for every $\theta \in \Theta$, there is some $o \in X$ such that $u(\theta, o) \geq 0$, then o_X will give the agent nonnegative utility for that type θ, because o_X is constrained to choose an outcome that maximizes $u(\theta, \cdot)$ among outcomes from X, and at least one of the outcomes in X gives nonnegative utility. So it is individually rational. ■

It follows that when we have an individual rationality constraint, in our search procedures, we do not need to expand nodes where for some type θ, there are no outcomes left in $O - Y$ that give the agent a nonnegative utility for θ.

6.7.6 Experimental results

In this subsection, we compare the performances of branch-and-bound DFS and IDA* over our search space with the performance of the mixed integer programming approach described earlier (using CPLEX 8.0), on random instances drawn from three different distributions. In each case, we investigate both scalability in the number of types and in the number of outcomes.

Uniform distribution, no IR

For this distribution, each value $u(\theta, o)$ and each value $g(\theta, o)$ is independently and uniformly drawn from $[0, 100]$. No IR constraint applies (all utilities are nonnegative).

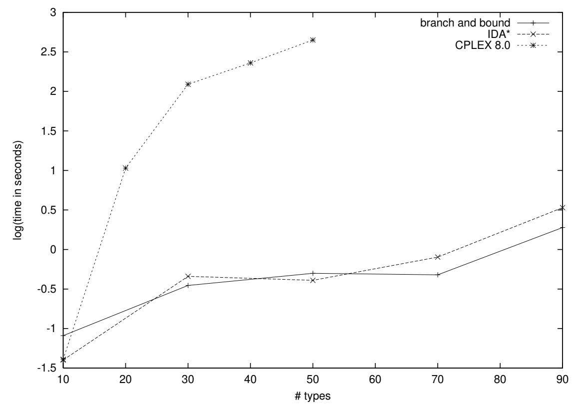


Figure 6.1: Performance vs. types for the uniform, no IR case with 20 outcomes.

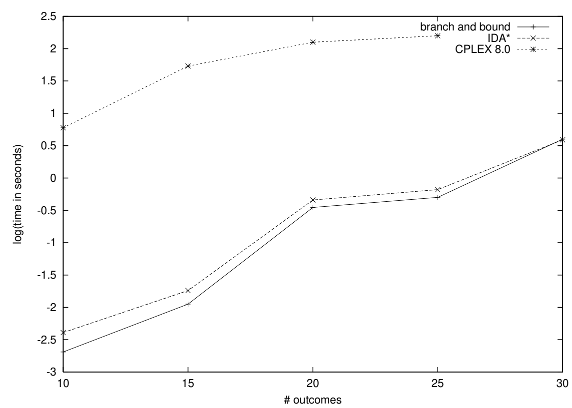


Figure 6.2: Performance vs. outcomes for the uniform, no IR case with 30 types.

Both versions of our algorithm outperform CPLEX soundly; our approach is especially more scalable in the number of types.

Uniform distribution, with IR

Now, each value $u(\theta, o)$ and each value $g(\theta, o)$ is independently and uniformly drawn from $[-50, 50]$. We apply an IR constraint (the agent can never get negative utility).

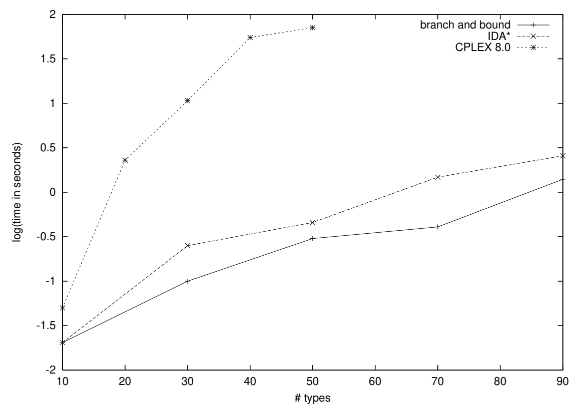


Figure 6.3: Performance vs. types for the uniform, with IR case with 20 outcomes.

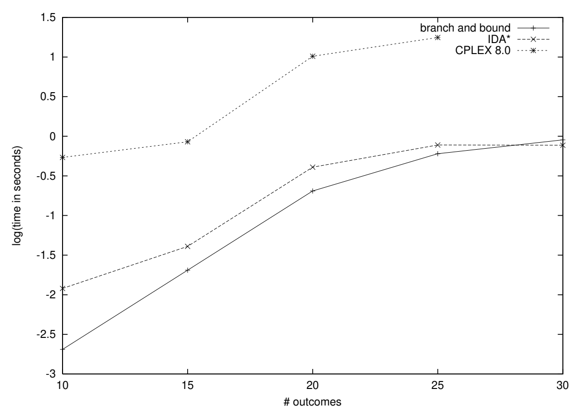


Figure 6.4: Performance vs. outcomes for the uniform, with IR case with 30 types.

Both versions of our algorithm still solidly outperform CPLEX, but the gaps are a little tighter; CPLEX manages to get a greater speedup factor out of the IR constraint.

Bartering

The final distribution corresponds to the bartering problem described earlier. The designer and the agent each have $m/2$ goods (for 2^m outcomes—each good can end up with either agent); the designer has a randomly drawn value (from $[0, 10]$) for each individual good (constituting g, which does not depend on θ in this case), and the agent has a randomly drawn value (from $[0, 10]$) for each individual good for each type (constituting u). The value of a bundle to an agent is the sum of the values of the individual goods.[15] If the total number of goods is odd, the agent gets one more good than the designer.

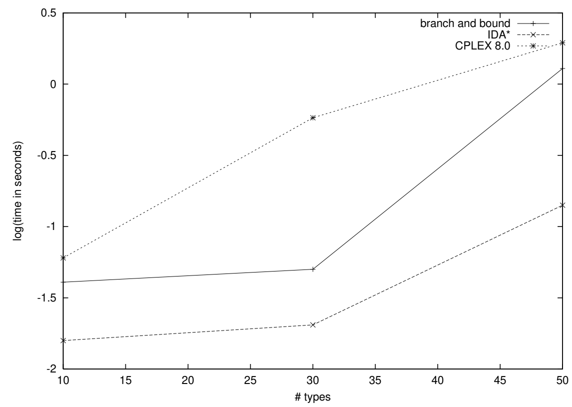


Figure 6.5: Performance vs. types for the bartering case with 32 outcomes.

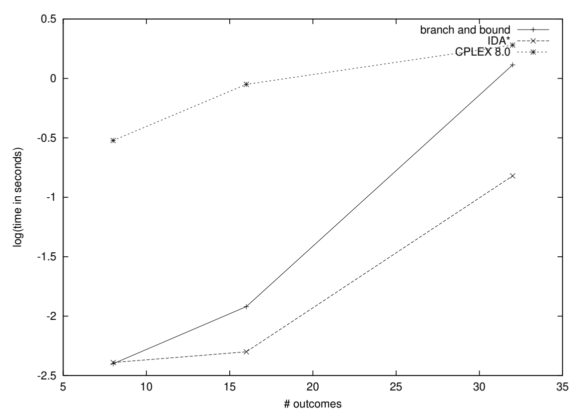


Figure 6.6: Performance vs. outcomes for the bartering case with 50 types.

The gaps here are much tighter, and it appears that CPLEX may in fact get the upper hand on even larger instances. (Space limitations prevented us from taking the experiments further.) CPLEX apparently makes very good use of the additional structure in this domain, whereas our algorithm

[15]There is nothing preventing our approach from having more complicated values over bundles; we simply felt it was nice to present the simplest example.

is not geared towards exploiting this structure. Also, IDA* seems to outperform branch-and-bound DFS now.

6.8 Structured outcomes and preferences

So far, we have only studied a flat representation of automated mechanism design problem instances, *e.g.* we assumed that all possible outcomes were explicitly listed in the input. However, in expressive preference aggregation, the outcome space is often too large to enumerate all the outcomes. Nevertheless, in such settings, the outcomes and the agents' preferences over them usually have some structure that allows the problem to still be concisely represented. In this section, we study one particular type of such structure: the agents may have to simultaneously decide on multiple, otherwise unrelated *issues*. In this case, the outcome space can be represented as the cross product of the outcome spaces for the individual issues. The next definition makes this precise.

Definition 30 *$O = O_1 \times O_2 \times \ldots \times O_r$ is a* valid decomposition *of O (where r is the number of* issues*) if the following two conditions hold:*

- *For each agent i, for each $1 \leq k \leq r$ there exists a function $u_i^k : \Theta_i \times O_k \to \mathbb{R}$ such that $u_i(\theta_i, (o^1, \ldots, o^r)) = \sum_{1 \leq k \leq r} u_i^k(\theta_i, o^k)$;*
- *For each $1 \leq k \leq r$ there exists a function $g^k : \Theta_1 \times \ldots \times \Theta_n \times O_k \to \mathbb{R}$ such that $g(\theta_1, \ldots, \theta_n, (o^1, \ldots, o^r)) = \sum_{1 \leq k \leq r} g^k(\theta_1, \ldots, \theta_n, o^k)$.*

We observe that when g is a social welfare function, the first condition implies the second, because if the first condition holds, $g(\theta_1, \ldots, \theta_n, (o^1, \ldots, o^r)) = \sum_{1 \leq i \leq n} u_i(\theta_i, (o^1, \ldots, o^r)) = \sum_{1 \leq i \leq n} \sum_{1 \leq k \leq r} u_i^k(\theta_i, o^k) = \sum_{1 \leq k \leq r} \sum_{1 \leq i \leq n} u_i^k(\theta_i, o^k)$, so that we can define $g^k(\theta_1, \ldots, \theta_n, o^k) = \sum_{1 \leq i \leq n} u_i^k(\theta_i, o^k)$.

We call automated mechanism design with a valid decomposition *multi-issue automated mechanism design*. It may seem that we can solve a multi-issue AMD instance simply by solving the AMD problem for each individual issue separately. However, doing so will in general not give the optimal mechanism. The reason is that in general, the designer may use one issue to tailor the incentives to get better results on another issue. For example, in an auction setting, one could think of the allocation as one issue, and the payments as another issue. Even when the designer is only concerned with bringing about the optimal allocation, the payments are still a useful instrument to give the bidders an incentive to bid truthfully. (We caution the reader that apart from this minor deviation, we do not consider the payments to be part of the outcome space O here.) As another example, we saw in Chapter 5 that using a single mechanism to decide on the donations to multiple charities can be more efficient than using a separate mechanism for each charity (Proposition 8). The hardness results later in this section will also imply that solving the AMD problem separately for each issue does not give the optimal solution. (The multi-issue AMD problem is NP-complete even in settings where the corresponding single-issue AMD problem is in P, so if the approach of solving the problem separately for each issue were optimal, we would have shown that P=NP.)

6.8.1 Example: Multi-item auctions

Consider auctioning a set of distinguishable items. If each of the m items can be allocated to any of n agents (or to no agent at all), the outcome space O has size $(n+1)^m$ (one for each possible allocation). If, for every bidder, the bidder's valuation for any bundle of items is the sum of the bidder's valuations of the individual items in the bundle, then we can decompose the outcome space as $O = O_1 \times O_2 \times \ldots \times O_m$, where $O_k = \{0, 1, 2, \ldots, n\}$ is the set of all possible allocations for item k (0 indicating it goes to no agent). Agent i's utility function can be written as $u_i((o^1, o^2, \ldots, o^m)) = \sum_{k \in \{1,\ldots,m\}} u_i^k(o^k)$ where u_i^k is given by $u_i^k(i) = v_i(k)$ and $u_i^k(j) = 0$ for $j \neq i$, where $v_i(k)$ is agent i's valuation for item k.

Two extensions of this that also allow for decomposable outcome spaces are the following:

- An agent, if it does not receive an item, still cares which agent (if any) receives that item—that is, there are *externalities* (as discussed in Chapter 2, Section 2.4). Here we no longer always have $u_i^k(j) = 0$ for $j \neq i$. For example, John may prefer it if the museum wins the painting rather than a private collector, because in the former case he can still see the painting.

- Some items exhibit substitutability or complementarity (so an agent's valuation for a bundle is not the sum of its valuations of the individual items in the bundle), but the items can be partitioned into subsets so that there are no substitution or complementarity effects across subsets in the partition. In this case, we can still decompose the outcome space over these subsets. For example, a plane trip, a hotel stay, a cell phone and a pager are all for sale. The plane trip and the hotel stay are each worthless without the other: they are perfect complements. The cell phone and the pager each reduce the value of having the other: they are (partial) substitutes. But the value of the plane trip or the hotel stay has nothing to do with whether one also gets the cell phone or the pager. Thus, we decompose the outcome space into two issues, one indicating the winners of the plane trip and hotel stay, and one indicating the winners of the cell phone and the pager.

In each of these settings, the approach of this section can be used directly to maximize any objective that the designer has. (This requires that the vaulations lie in a finite interval and are discretized.)

6.8.2 Complexity

In this subsection we show that for the multi-issue representation, the three most important variants of the problem of designing a deterministic mechanism are NP-complete. Of course, the hardness results from Section 6.3 already imply this, because flatly represented problem instances are a special case of the multi-issue representation. However, it turns out that under the multi-issue representation, hardness occurs even in much more restricted settings (with small type spaces and a small outcome space for each issue).

Theorem 46 *The AMD problem for designing deterministic mechanisms without payments is NP-complete under the multi-issue representation, even when the objective is social welfare, there is only a single type-reporting agent (in addition to an agent that does not report a type), the probability distribution over Θ is uniform, there are only two possible types, and $|O_i| = 2$ for all i. (Membership in NP is guaranteed only if the number of agents is constant.)*

Proof: The problem is in NP because we can nondeterministically generate the full outcome selection function o (as long as the number of agents is constant, because otherwise there are exponentially many type vectors). To show NP-hardness, we reduce an arbitrary KNAPSACK instance (given by a set of pairs $\{(c_j, v_j)\}_{j\in\{1,\dots,m\}}$, a cost limit C, and a value goal V) to the following single-agent deterministic multi-issue AMD instance, where payments are not allowed and the objective is social welfare. Let the number of issues be $r = m+1$. For every $j \in \{1,\dots,m+1\}$, we have $O_j = \{t, f\}$. The agent's type set, over which there is a uniform distribution, is $\Theta = \{\theta^1, \theta^2\}$, and the agent's utility function $u = \sum_{k\in\{1,\dots,r\}} u^k$ is given by:

- For all $k \in \{1,\dots,m\}$, $u^k(\theta^1, t) = AB$ where $A = 2\sum\limits_{j\in\{1,\dots,m\}} c_j$ and $B = 2\sum\limits_{j\in\{1,\dots,m\}} v_j$; and $u^k(\theta^1, f) = 0$.
- $u^{m+1}(\theta^1, t) = u^{m+1}(\theta^1, f) = 0$.
- For all $k \in \{1,\dots,m\}$, $u^k(\theta^2, t) = c_k$, and $u^k(\theta^2, f) = 0$.
- $u^{m+1}(\theta^2, t) = C$, and $u^{m+1}(\theta^2, f) = 0$.

The part of the social welfare that does not correspond to any agent in the game is given by $u_0 = \sum\limits_{k\in\{1,\dots,r\}} u_0^k$ where

- For all $k \in \{1,\dots,m\}$, $u_0^k(t) = 0$, and $u^k(f) = v_k A$.
- $u_0^{m+1}(t) = u_0^{m+1}(f) = 0$.

The goal social welfare is given by $G = \frac{A(mB+V)}{2}$. We show the two instances are equivalent. First suppose there is a solution to the KNAPSACK instance, that is, a subset S of $\{1,\dots,m\}$ such that $\sum\limits_{j\in S} c_j \leq C$ and $\sum\limits_{j\in S} v_j \geq V$. Then consider the following mechanism:

- For all $k \in \{1,\dots,m\}$, $o^k(\theta^1) = t$.
- For $k \in \{1,\dots,m\}$, $o^k(\theta^2) = f$ if $k \in S$, $o^k(\theta^2) = t$ otherwise.
- $o^{m+1}(\theta^1) = f$, and $o^{m+1}(\theta^2) = t$.

First we show there is no incentive for the agent to misreport. If the agent has type θ^1, then it is getting the best possible outcome for each issue by reporting truthfully, so there is certainly no incentive to misreport. If the agent has type θ^2, reporting truthfully gives it a utility of $C + \sum\limits_{j\in\{1,\dots,m\},\notin S} c_j$, whereas reporting θ^1 instead gives it a utility of $\sum\limits_{j\in\{1,\dots,m\}} c_j$; so the marginal utility of misreporting is $-C + \sum_{j \in S} c_j \leq -C + C = 0$. Hence there is no incentive to misreport. Now we show that the goal social welfare is reached. If the agent has type θ^1, the social welfare will be mAB. If it has type θ^2, it will be $\sum_{j \in S} v_j A + \sum_{j \in \{1,\ldots,m\}, \notin S} c_j + C \geq \sum_{j \in S} v_j A \geq VA$. Hence the expected social welfare is at least $\frac{mAB+VA}{2} = G$. So there is a solution to the AMD instance. Now suppose there is a solution to the AMD instance. If it were the case that, for some $j \in \{1, \ldots, m\}$, $o^j(\theta^1) = f$, then the maximum social welfare that could possibly be obtained (even if we did not worry about misreporting) would be $\frac{(m-1)AB+v_jA}{2} + \frac{\sum_{j \in \{1,\ldots,m\}} v_jA+C}{2} = \frac{(m-1)AB+\frac{AB}{2}+v_jA+C}{2} < \frac{mAB+VA}{2} = G$. Thus, for all $j \in \{1, \ldots, m\}$, $o^k(\theta^1) = t$. Now, let $S = \{j \in \{1, \ldots, m\} : o^j(\theta^2) = f\}$. Then, if the agent has type θ^2 and reports truthfully, it will get utility at most $C + \sum_{j \in \{1,\ldots,m\}, \notin S} c_j$, as opposed to the at least $\sum_{j \in \{1,\ldots,m\}} c_j$ that it could get for this type by reporting θ^1 instead. Because there is no incentive to misreport in the mechanism, it follows that $\sum_{j \in S} c_j \leq C$. Also, the total social welfare obtained by the mechanism is at most $\frac{mAB+\sum_{j \in S} v_jA + \sum_{j \in \{1,\ldots,m\}, \notin S} c_j + C}{2}$. Because $\sum_{j \in \{1,\ldots,m\}, \notin S} c_j + C < A$, and all the other terms in the numerator are some integer times A, it follows that this fraction is greater than or equal to the goal $\frac{mAB+VA}{2}$ (where the numerator is also an integer times A) if and only if $\sum_{j \in S} v_j \geq V$—and this must be the case because by assumption, the mechanism is a solution to the AMD instance. It follows that S is a solution to the KNAPSACK instance. ■

Theorem 47 *The AMD problem for designing deterministic mechanisms with payments is NP-complete under the multi-issue representation, even when the objective does not depend on the payments made, there is only a single type-reporting agent, the probability distribution over Θ is uniform, there are only two possible types, and $|O_i| = 2$ for all i. (Membership in NP is guaranteed only if the number of agents is constant.)*

Proof: It is easy to see that the problem is in NP. (We can nondeterministically generate the outcome function as before, after which setting the payments is a linear programming problem and can hence be done in polynomial time—presuming, again, that the number of agents is constant.) To show NP-hardness, we reduce an arbitrary KNAPSACK instance (given by a set of pairs $\{(c_j, v_j)\}_{j \in \{1,\ldots,m\}}$, a cost limit C, and a value goal V) to the following single-agent deterministic multi-issue AMD instance, where payments are allowed. Let the number of issues be $r = m + 1$. For every $j \in \{1, \ldots, m+1\}$, we have $O_j = \{t, f\}$. The agent's type set, over which there is a uniform distribution, is $\Theta = \{\theta^1, \theta^2\}$, and the agent's utility function $u = \sum_{k \in \{1,\ldots,r\}} u^k$ is given by:

- For all $k \in \{1, \ldots, m\}$, $u^k(\theta^1, t) = c_k$, and $u^k(\theta^1, f) = 0$.
- $u^{m+1}(\theta^1, t) = C$, and $u^{m+1}(\theta^1, f) = 0$.
- For all $k \in \{1, \ldots, m\}$, $u^k(\theta^2, t) = 0$, $u^k(\theta^2, f) = c_k$.

- $u^{m+1}(\theta^2, t) = 0$, and $u^{m+1}(\theta^2, f) = C$.

The objective function $g = \sum_{k \in \{1,\ldots,r\}} g^k$ is given by

- For all $k \in \{1, \ldots, m\}$, $g^k(\theta^1, t) = 0$, and $g^k(\theta^1, f) = A$ where $A = 2 \sum_{j \in \{1,\ldots,m\}} v_j$.
- For all $k \in \{1, \ldots, m\}$, $g^k(\theta^2, t) = v_k$, $g^k(\theta^2, f) = 0$.
- $g^{m+1}(\theta^1, t) = g^{m+1}(\theta^1, f) = g^{m+1}(\theta^2, t) = g^{m+1}(\theta^2, f) = 0$.

The goal for the objective function is given by $G = \frac{mA+V}{2}$. We show the two instances are equivalent. We first observe a useful fact about the utility function: when there are no payments, for any outcome function, the incentive for the agent to misreport when it has type θ^1 is the same as the incentive for the agent to misreport when it has type θ^2. That is, for any outcome function o, $u(\theta^1, o(\theta^2)) - u(\theta^1, o(\theta^1)) = u(\theta^2, o(\theta^1)) - u(\theta^2, o(\theta^2))$. To see why, first consider that if we (say) set $o^k(\theta^1) = o^k(\theta^2) = f$ everywhere, obviously this is true. Then, whenever, for some k, we "flip" $o^k(\theta^1)$ to t, the second term (including the minus sign) on the left hand side decreases by the same amount as the first term on the right hand side. Similarly, whenever we "flip" $o^k(\theta^2)$ to t, the first term on the left hand side increases by the same amount as the second term (including the minus sign) on the right hand side. A corollary of this observation is that for this example, *payments cannot help us make the mechanism truthful*. For, if without payments, the mechanism would not be truthful, the agent would have an incentive to lie for both types (without payments). Then, if the agent needs to pay more for reporting one type than for the other, the agent will still have an (even bigger) incentive to lie for at least that type. Thus, we may as well assume payments are not possible. Now, suppose there is a solution to the KNAPSACK instance, that is, a subset S of $\{1, \ldots, m\}$ such that $\sum_{j \in S} c_j \leq C$ and $\sum_{j \in S} v_j \geq V$. Then consider the following mechanism:

- For all $k \in \{1, \ldots, m\}$, $o^k(\theta^1) = f$.
- For $k \in \{1, \ldots, m\}$, $o^k(\theta^2) = t$ if $k \in S$, $o^k(\theta^2) = f$ otherwise.
- $o^{m+1}(\theta^1) = t$, and $o^{m+1}(\theta^2) = f$.

(The payment function is 0 everywhere.) First we show there is no incentive for the agent to misreport. Because we observed that the incentive to misreport is the same for both types, we only need to show this for one type. We will show it when the agent's true type is θ^1. In this case, reporting truthfully gives utility C, and reporting θ^2 gives utility $\sum_{j \in S} c_j \leq C$. Hence there is no incentive to misreport. Now we show that the goal value of the objective is reached. If the agent has type θ^1, the value of the objective function will be mA. If it has type θ^2, it will be $\sum_{j \in S} v_j \geq V$. Hence the expected value of the objective function is at least $\frac{mA+V}{2} = G$. So there is a solution to the AMD instance. Finally, suppose there is a solution to the AMD instance. As a reminder, payments cannot help us, so we may assume they are always 0. If it were the case that, for some $j \in \{1, \ldots, m\}$, $o^k(\theta^1) = t$, then the maximum social welfare that could possibly be obtained (even if we did not worry about misreporting) would be $\frac{(m-1)A+\sum\limits_{j\in\{1,\ldots,m\}} v_j}{2} < \frac{mA}{2} < G$. Thus, for all $j \in \{1,\ldots,m\}$, $o^k(\theta^1) = f$. Now, let $S = \{j \in \{1,\ldots,m\} : o^j(\theta^2) = t\}$. The incentive for the agent to misreport when it has type θ^1 is then at least $-C + \sum\limits_{j\in\{1,\ldots,m\}} c_j$, which should be less than or equal to 0, so that $\sum\limits_{j\in\{1,\ldots,m\}} c_j \leq C$. Additionally, the expected value of the objective function is $\frac{mA+\sum\limits_{j\in S} v_j}{2}$, which should be at least $G = \frac{mA+V}{2}$. It follows that $\sum\limits_{j\in S} v_j \geq V$. Thus S is a solution to the KNAPSACK instance. ■

Theorem 48 *The AMD problem for designing deterministic mechanisms is NP-complete under the multi-issue representation, even when the objective is to maximize total payments made (under an IR constraint), there is only a single type-reporting agent, the probability distribution over Θ is uniform, there are only two possible types, and $|O_i| = 2$ for all i. (Membership in NP is guaranteed only if the number of agents is constant.)*

Proof: It is easy to see that the problem is in NP. (We can nondeterministically guess an outcome function, after which setting the payments is a linear programming problem and can hence be done in polynomial time.) To show NP-hardness, we reduce an arbitrary KNAPSACK instance (given by a set of pairs $\{(c_j, v_j)\}_{j\in\{1,\ldots,m+1\}}$, a cost limit C, and a value goal V) to the following single-agent deterministic multi-issue AMD instance, where we seek to maximize the expected payments from the agent. Let the number of issues be $r = m+1$. For every $j \in \{1,\ldots,m\}$, we have $O_j = \{t, f\}$. The agent's type set, over which there is a uniform distribution, is $\Theta = \{\theta^1, \theta^2\}$, and the agent's utility function $u = \sum_{k\in\{1,\ldots,r\}} u^k$ is given by:

- For all $k \in \{1,\ldots,m\}$, $u^k(\theta^1, t) = c_k A$ where $A = 4 \sum\limits_{j\in\{1,\ldots,m\}} v_j$; and $u^k(\theta^1, f) = 0$.
- $u^{m+1}(\theta^1, t) = 0$, and $u^{m+1}(\theta^1, f) = -CA$.
- For all $k \in \{1,\ldots,m\}$, $u^k(\theta^2, t) = v_k$, and $u^k(\theta^2, f) = 0$.
- $u^{m+1}(\theta^2, t) = 0$, and $u^{m+1}(\theta^2, f) = 0$.

The goal expected revenue is given by $G = \frac{AB+V}{2}$, where $B = \sum\limits_{j\in\{1,\ldots,m\}} c_j$. We show the two instances are equivalent. First suppose there is a solution to the KNAPSACK instance, that is, a subset S of $\{1,\ldots,m\}$ such that $\sum\limits_{j\in S} c_j \leq C$ and $\sum\limits_{j\in S} v_j \geq V$. Then consider the following mechanism. Let the outcome function be

- For all $k \in \{1,\ldots,m\}$, $o^k(\theta^1) = t$.
- For $k \in \{1,\ldots,m\}$, $o^k(\theta^2) = t$ if $k \in S$, $o^k(\theta^2) = f$ otherwise.
- $o^{m+1}(\theta^1) = t,\ o^{m+1}(\theta^2) = f$.

Let the payment function be $\pi(\theta^1) = AB$, $\pi(\theta^2) = \sum_{j \in S} v_j$. First, to see that the IR constraint is satisfied, observe that for each type, the mechanism extracts exactly the agent's utility obtained from the outcome function. Second, we show there is no incentive for the agent to misreport. If the agent has type θ^1, reporting θ^2 instead gives a utility (including payments) of $-CA + \sum_{j \in S} c_j A - \sum_{j \in S} v_j \leq -CA + CA - \sum_{j \in S} v_j < 0$, which is what the agent would have got for reporting truthfully. If the agent has type θ^2, reporting θ^1 instead gives a utility (including payments) of $\frac{A}{4} - AB < 0$, which is what the agent would have got for reporting truthfully. Hence there is no incentive to misreport. Third, the goal expected payment is reached because $\frac{AB + \sum_{j \in S} v_j}{2} \geq \frac{AB+V}{2} = G$. So there is a solution to the AMD instance. Now suppose there is a solution to the AMD instance. The maximum utility that the agent can get from the outcome function if it has type θ^2 is $\frac{A}{4}$, and by the IR constraint this is the maximum payment we may extract from the agent when the reported type is θ^2. Because the goal is greater than $\frac{AB}{2}$, it follows that the payment the mechanism should extract from the agent when the reported type is θ^1 is at least $AB - \frac{A}{4}$. Because the maximum utility the agent can derive from the outcome function in this case is AB, it follows that the agent's utility (including payments) for type θ^1 can be at most $\frac{A}{4}$. Now consider the set $S = \{j \in \{1, \ldots, m\} : o^j(\theta^2) = t\}$. Then, if the agent falsely reports type θ^2 when the true type is θ^1, the utility of doing so (including payments) is at least $\sum_{j \in S} c_j A - CA - \frac{A}{4}$. This is to be at most the agent's utility for reporting truthfully in this case, which is at most $\frac{A}{4}$. It follows that $\sum_{j \in S} c_j A - CA - \frac{A}{4} \leq \frac{A}{4}$, which is equivalent to $\sum_{j \in S} c_j \leq C + \frac{1}{2}$. Because the c_j and C are integers, this implies $\sum_{j \in S} c_j \leq C$. Finally, because we need to extract at least a payment of V from the agent when type θ^2 is reported, but the utility that the agent gets from the outcome function in this case is at most $\sum_{j \in S} v_j$ and we can extract at most this by the IR constraint, it follows that $\sum_{j \in S} v_j \geq V$. Thus, S is a solution to the KNAPSACK instance. ■

The NP-hardness of automatically designing optimal deterministic mechanisms under the multi-issue representation was already implied by similar results for the unstructured (single-issue) representation. However, the fact that (unlike under the unstructured representation) NP-hardness occurs even with very small type sets is perhaps discouraging. On the other hand, one can be positive about the fact that the problem remains in NP (if the number of agents is constant), even though the representation is exponentially more concise. In the next subsection, we show that *pseudopolynomial-time* algorithms do exist for this problem (given a constant number of types). More significantly, in the subsection after that, we show that optimal *randomized* mechanisms can still be designed in polynomial time even under the multi-issue representation. Hence, it seems that this representation is especially useful when we allow for randomized mechanisms.

6.8.3 A pseudopolynomial-time algorithm for a single agent

In this subsection we develop a pseudopolynomial-time algorithm that shows that the first two multi-issue AMD problems discussed in the previous subsection are only *weakly* NP-complete. (A problem is only weakly NP-complete if it is NP-complete, but there exists an algorithm that would run in polynomial time if the numbers in the instance were given in unary, rather than binary—a *pseudopolynomial-time* algorithm.) This algorithm only works when there is only one type-reporting agent. While this is still a significant problem because of the conflict of interest between the designer and the agent, it is an interesting open problem to see if the algorithm can be generalized to settings with multiple agents.

Theorem 49 *If there is only a single agent, the number of types is a constant, and the objective does not depend on payments made, then the optimal deterministic mechanism can be found in pseudopolynomial time under the multi-issue representation using dynamic programming, both with and without payments, both for* ex post *and* ex interim *IR, and both for implementation in dominant strategies and for implementation in Bayes-Nash equilibrium.*

Proof: The dynamic program adds in the issues one at a time. For each $k \in \{0, 1, \ldots, r\}$, it builds a matrix which concerns a reduced version of the problem instance where only the first k issues are included. Let $r(\theta^i, \theta^j) = u(\theta^i, o(\theta^j)) - u(\theta^i, o(\theta^i))$, that is, the regret that the agent has for reporting its true type θ^i rather than submitting the false report θ^j. (These regrets may be negative.) Any outcome function mapping the reported types to outcomes defines a vector of $|\Theta|(|\Theta|-1)$ such regrets, one for each pair (θ^i, θ^j). Then, our matrix for the first k issues contains, for each possible regret vector v, a number indicating the highest expected value of the objective function that can be obtained with an outcome function over the first k issues whose regret vector is dominated by v. (One vector is said to be dominated by another if all entries of the former are less than or equal to the corresponding ones of the latter.) This entry is denoted $M^k[v]$. We observe that if v_1 dominates v_2, then $M^k[v_1] \geq M^k[v_2]$. If the absolute value of the regret between any two types is bounded by R, it suffices to consider $(2R+1)^{|\Theta|(|\Theta|-1)}$ regret vectors (each entry taking on values in $\{-R, -R+1, \ldots, 0, \ldots, R-1, R\}$). The matrix for $k = 0$ (i.e., when no issues have yet been added) is 0 everywhere. We then successively build up the next matrix as follows. When we add in issue k, there are $|O^k|^{|\Theta|}$ possibilities for setting the outcome function o^k from types to elements of O^k. Denoting a possible setting of o^k by a vector $w = (o^k(\theta^1), o^k(\theta^2), \ldots, o^k(\theta^{|\Theta|}))$, letting $g^k(w) = \sum_{\theta \in \Theta} g^k(\theta, o^k(\theta))$ be the total value gained in the objective function as a result of this vector, and letting $r(w) = (u^k(\theta^i, o^k(\theta^j)) - u^k(\theta^i, o^k(\theta^i))_{\{\theta^i \neq \theta^j\}}$ be the regret vector over this issue alone, we have the following recursive identity for $k > 0$: $M^k[v] = \max_w\{g^k(w) + M^{k-1}[v - r(w)]\}$. It is possible that, when we use this identity to fill in the matrices, the identity refers to an entry "outside" the previous matrix, that is, one of the entries of $v - r(w)$ has absolute value greater than R. If this occurs, one of the following two cases applies:

- One of the entries is greater than R. This means that the regret allowed for one of the pairs (θ^i, θ^j) is greater than the maximum it could be. We may reduce the value of this entry to R, without causing a reduction in the highest value of the objective function that can be obtained.

- One of the entries is smaller than $-R$. This means that the regret allowed for one of the pairs (θ^i, θ^j) is smaller than the minimum it could be. Hence, it is impossible to construct an outcome function that satisfies this, and hence we simply say $M^{k-1}[v - r(w)] = -\infty$.

Once we have constructed M^r, we can use this matrix to solve any of our deterministic automated mechanism design problems. If payments are not allowed, we simply look at the entry $M^r[(0, 0, \ldots, 0)]$, because this is the highest possible expected value of the objective function that we can obtain without the agent having positive regret anywhere. If payments are allowed, then we look at all the entries $M^r[v]$ where the regret vector v is such that we can set the payments so as to make every regret disappear—that is, where we can set π_θ such that for any θ^i, θ^j, we have $r(\theta^i, \theta^j) + \pi(\theta^j) - \pi(\theta^i) \leq 0$. (This is a simple linear program and can hence be solved in polynomial time.) Of all these entries, we choose the one with the highest value of the objective function. If we want to know not only the highest possible expected value of the objective function, but also a mechanism that achieves it, we need to store at each step not only the highest possible expected value for each matrix entry, but also a partial outcome function that achieves it. ■

6.8.4 A polynomial-time algorithm for randomized mechanisms

When we allow randomization in the mechanism, it turns out that an optimal mechanism can be designed in time that is polynomial in the length of the concise representation, as in the case of flatly represented instances (Section 6.4).

Theorem 50 *With a constant number of agents, the optimal randomized mechanism can be found in polynomial time under the multi-issue representation using linear programming, both with and without payments, both for* ex post *and* ex interim *IR, and both for implementation in dominant strategies and for implementation in Bayes-Nash equilibrium.*

Proof: We cannot simply use the linear program from Section 6.4, because it would have an exponential number of variables under the multi-issue representation. However, we can reduce the number of variables to a polynomial number. To this end, we observe:

- For all i, $E(u_i|(\hat{\theta}_1, \ldots, \hat{\theta}_n), \theta_i) = \sum_{(o^1,\ldots,o^r) \in O} P((o^1, \ldots, o^r)|(\hat{\theta}_1, \ldots, \hat{\theta}_n)) \sum_{1 \leq k \leq r} u_i^k(\theta_i, o^k)$
$= \sum_{1 \leq k \leq r} \sum_{(o^1,\ldots,o^r) \in O} P((o^1, \ldots, o^r)|(\hat{\theta}_1, \ldots, \hat{\theta}_n)) u_i^k(\theta_i, o^k) =$
$\sum_{1 \leq k \leq r} \sum_{o_*^k \in O^k} u_i^k(\theta_i, o_*^k) \sum_{(o^1,\ldots,o^r): o^k = o_*^k} P((o^1, \ldots, o^r)|(\hat{\theta}_1, \ldots, \hat{\theta}_n)) =$
$\sum_{1 \leq k \leq r} \sum_{o_*^k \in O^k} P(o^k = o_*^k|(\hat{\theta}_1, \ldots, \hat{\theta}_n)) u_i^k(\theta_i, o_*^k)$.

- Similarly, $E(g|(\hat{\theta}_1, \ldots, \hat{\theta}_n)) = \sum_{1 \leq k \leq r} \sum_{o_*^k \in O^k} P(o^k = o_*^k|(\hat{\theta}_1, \ldots, \hat{\theta}_n)) g^k((\hat{\theta}_1, \ldots, \hat{\theta}_n), o_*^k)$.

It follows that for the purposes at hand, we care only about the quantities $P(o^k = o_*^k|(\hat{\theta}_1, \ldots, \hat{\theta}_n))$, rather than about the entire distribution. There are precisely $\sum_{1 \leq k \leq r} |O^k| \prod_{1 \leq i \leq n} |\Theta^i|$ such probabilities, which is a polynomial number when the number of agents, n, is a constant. Additionally, only

$n \prod_{1 \le i \le n} |\Theta^i|$ variables are needed to represent the payments made by the agents in each case (or none if payments are not possible).

The linear program, which contains constraints for the IC notion and IR notion in question, and attempts to optimize some linear function of the expected value of the objective function and payments made, is now straightforward to construct. Because linear programs can be solved in polynomial time, and the number of variables and equations in our program is polynomial for any constant number of agents, the problem is in P. ■

6.9 Summary

In this chapter, we introduced *automated mechanism design*, which consists of solving for the optimal mechanism for the instance at hand using constrained optimization techniques. We showed that automatically designing optimal deterministic mechanisms is NP-hard in most cases, but designing the optimal randomized mechanism can be done in polynomial time using linear programming. Moreover, by requiring the probability variables in these linear programs to take on integer variables, we obtain a mixed integer programming approach for designing optimal deterministic mechanisms. We showed some initial applications, including divorce settlement, optimally auctioning one or more items, and deciding on whether to build public goods. We presented scalability results for the mixed integer/linear programming approaches; we also gave a special-purpose algorithm for a special case that outperforms the mixed integer programming approach. Finally, we studied a representation for instances of the automated mechanism design problem that is concise when there are multiple unrelated issues, and studied how this changes the complexity of the problem.

In the next few chapters, we will take a break from automated mechanism design and instead focus on the effects of agents' computational limitations on their behavior in (manually designed) mechanisms. We will return to the topic of designing mechanisms automatically in Chapter 10, where we automatically design mechanisms for agents with computational limitations.

Chapter 7

Game-Theoretic Foundations of Mechanism Design

As mentioned in Chapter 4, a result known as the *revelation principle* is often used to justify restricting attention to truthful mechanisms. Informally, it states that, given a mechanism (not necessarily a truthful or even a direct-revelation mechanism) that produces certain outcomes when agents behave strategically, there exists a truthful mechanism that produces the same outcomes. Of course, this informal statement is too unspecific to truly understand its meaning. Which type of truthfulness is obtained—implementation in dominant strategies, Bayes-Nash equilibrium, or something else? More importantly, what exactly does it mean for the agents to "behave strategically"? It turns out that there are really multiple versions of the revelation principle: different types of strategic behavior lead to different types of truthfulness. In this chapter, we will review some basic concepts from game theory, which will provide us with basic definitions of strategic behavior. We will also present two versions of the revelation principle. This will give us a deeper understanding of the motivation for restricting attention to truthful mechanisms, which will be helpful in the next two chapters, where we argue that non-truthful mechanisms need to be considered when agents are computationally bounded.

7.1 Normal-form games

Perhaps the most basic representation of a strategic settings is a game in *normal* or *strategic* form. In such a game, there are n agents (or *players*), and each player i has a set of *strategies* S_i to select from. The players select their strategies simultaneously, and based on this each player i receives a utility $u_i(s_1, \ldots, s_n)$. In the case where $n = 2$ and the number of strategies for each agent is finite, we can represent the game in *(bi)matrix form*. To do so, we label one player the row player, and the other the column player; then, we add a row to the matrix for each row player strategy, and a column for each column player strategy; finally, in each entry of the matrix, we place the players' utilities (starting with the row player's) for the outcome of the game that corresponds to this entry's row and column.

For example, the well-known game of rock-paper-scissors has the following normal-form representation:

	R	P	S
R	0,0	-1,1	1,-1
P	1,-1	0,0	-1,1
S	-1,1	1,-1	0,0

(Note that here, each row and each column is given a label (R, P, S); such labels do not have any strategic importance, so we will sometimes omit them.) Rock-paper-scissors is what is known as a *zero-sum* game, because within each entry of the matrix, the payoffs sum to zero—what one player gains, the other loses. If the payoffs in each entry of the matrix sum to a constant other than zero, the game is effectively still a zero-sum game, because affine transformations of utility do not affect a player's behavior.

7.1.1 Minimax strategies

How should we play rock-paper-scissors (and other zero-sum games)? Let us suppose, pessimistically, that the other player has good insight into how we play. Then, having a deterministic strategy (say, playing "rock" with probability one) is not a good idea, because the other player can play "paper" and win. Instead, it is better to randomize—for example, play each action with probability $1/3$. The set of randomizations ΔS_i over player i's (original) set of strategies in the game is known as the set of player i's *mixed* strategies. (For contrast, we will refer to S_i as the set of *pure* strategies.)

The most conservative way to play a two-player zero-sum game is to assume that the other player is able to predict one's mixed strategy perfectly. Then, one should choose one's mixed strategy to minimize the maximum utility that the other player can obtain (given that that player knows the mixed strategy). Formally, using the common notation $-i$ to denote "the player other than i," player i should choose a strategy from $\arg\min_{\sigma_i \in \Delta(S_i)} \max_{s_{-i} \in S_{-i}} u_{-i}(\sigma_i, s_{-i})$. (When we give a utility function a mixed strategy as an argument, it simply produces the expected utility given that mixed strategy.) This cautious manner of play may appear very favorable to player $-i$ (given that that $-i$ really *does* know player i's strategy). However, in rock-paper-scissors, the minimax strategy is to play each pure strategy with probability $1/3$, and in this case, any action that the opponent takes will result in an expected utility of 0 for both players. So at least in this game, there is no benefit to being able to choose one's strategy based on the opponent's mixed strategy. This is no accident: in fact, the famous Minimax Theorem [von Neumann, 1927] shows that the players' expected utilities will be the same regardless of which player gets to choose last. Formally, we have (if the utilities in each entry sum to 0): $\arg\min_{\sigma_1 \in \Delta(S_1)} \max_{s_2 \in S_2} u_2(\sigma_1, s_2) = -\arg\min_{\sigma_2 \in \Delta(S_2)} \max_{s_1 \in S_1} u_1(\sigma_2, s_1)$. Hence, it is natural to play minimax strategies in two-player zero-sum games.

What if the game is not zero-sum? As will become clear shortly, no perfect generalization of the Minimax Theorem exists; nevertheless, there are still ways of solving these games. One simple, but not always applicable notion is that of *dominance*, which will be discussed in the next section.

7.1.2 Dominance and iterated dominance

Consider the following game, commonly known as the *Prisoner's Dilemma*:

	S	C
S	-1,-1	-3,0
C	0,-3	-2,-2

The story behind the Prisoner's Dilemma game is as follows. Two criminals are arrested in connection with a major crime, but there is only enough evidence to convict them of a minor crime. The criminals are put in separate rooms, and are each given the option of confessing to the major crime (C) or keeping silent (S). If both keep silent, they are convicted of the minor crime and sentenced to one year in prison. If one confesses and the other does not, no charges at all will be filed against the criminal that confesses, and the one that does not is convicted of the major crime and sentenced to 3 years in prison. Finally, if both confess, they are both convicted of the major crime and given a slightly reduced sentence of 2 years in prison.

How should each criminal play? (Note that it is assumed that there is no opportunity for retaliation afterwards, nor do the criminals care about each other's fate—each prisoner's objective is simply to minimize the amount of time that he spends in prison.) If the other criminal confesses, it is better to confess and get -2 rather than -3. But similarly, if the other criminal keeps silent, it is better to confess and get 0 rather than -1. So, confessing is always better, and both criminals will confess—even though this will give each of them a worse outcome than if they had kept silent.[1] We say that confessing is a *dominant strategy*. Formally:

Definition 31 *Player i's strategy $\sigma'_i \in \Delta(S_i)$ is said to be* strictly dominated *by player i's strategy $\sigma_i \in \Delta(S_i)$ if for any vector of strategies $s_{-i} \in S_{-i}$ for the other players, $u_i(\sigma_i, s_{-i}) > u_i(\sigma'_i, s{-}i)$. Player i's strategy $\sigma'_i \in \Delta(S_i)$ is said to be* weakly dominated *by player i's strategy $\sigma_i \in \Delta(S_i)$ if*

[1]While prisoners' confessing to a crime may not appear to be such a bad outcome, there are many other real-world strategic situations with roughly the same structure where we clearly would prefer the agents to cooperate with each other and obtain the higher utilities. For example, there are settings where both players would be better off if each invested in a given public good, but if players act selfishly, neither will invest. Perhaps due to the frustrating nature of such outcomes, many suggestions have been made as to why an agent may still choose to act cooperatively. For example, the agents may care about each other's welfare, or bad behavior may cause failed cooperation, or even retaliation, in the future. Such arguments amount to nothing more than saying that the game structure and its utilities are inaccurate (or at least incomplete). Indeed, one should always be careful to model one's setting accurately, but this does not resolve the problem in the many settings that really are modeled accurately by a Prisoner's Dilemma game. A possible exception is the following argument. Suppose a player believes that the other player reasons *exactly* like him, and will therefore always make the same decision. Then, if the former player cooperates, so will the other player; if he does not, neither will the other player. Therefore, the first player should cooperate. This type of reasoning has been called "superrationality" [Hofstadter, 1985], but it quickly leads to difficult questions of causality (does choosing to cooperate "cause" the other player to cooperate?) and free will (is one's decision already pre-ordained given that the other player must do the same?). This is closely related to *Newcomb's paradox* [Nozick, 1969], in which a superintelligent or even omniscient being presents an agent with two boxes, each of which contains some nonnegative amount of money. The agent can choose to take either the contents of the first box only, or the contents of both boxes. The catch is that when filling the boxes, the being predicted whether the agent would take one or both boxes, and if it predicted that the agent would choose only one box, it placed significantly more money in that one box than it otherwise would have placed in both boxes together. Moreover, the being has been absolutely flawless in predicting other, previous agents' choices. It can be argued that the agent should choose only the one box, because then the being presumably would have put much more money in that box; or that the agent should choose both boxes, since the amounts in the boxes are already fixed at this point. In this dissertation, I will not address these issues and simply follow the standard model in which one can make a decision without affecting one's beliefs about what the other players will decide or have decided (which, for most real-world settings, is an accurate model).

for any vector of strategies s_{-i} for the other players, $u_i(\sigma_i, s_{-i}) \geq u_i(\sigma'_i, s_{-i})$, and for at least one vector of strategies s_{-i} for the other players, $u_i(\sigma_i, s_{-i}) > u_i(\sigma'_i, s_{-i})$.

This definition allows the dominating strategy σ_i and the dominated strategy σ'_i to be mixed strategies, although the restriction where these strategies must be pure can also be of interest (especially to avoid assumptions on agents' attitudes towards risk). There are other notions of dominance, such as *very weak* dominance (in which no strict inequality is required, so two strategies can dominate each other), but this dissertation will not study those notions.

In *iterated dominance*, dominated strategies are removed from the game, and no longer have any effect on future dominance relations. For example, consider the following modification of the Prisoner's Dilemma in which the District Attorney severely dislikes the row criminal and would press charges against him even if he were the only one to confess:

	S	C
S	-1,-1	-3,0
C	-2,-3	-2,-2

Now, the dominance argument only works for the column player. However, because (using the dominance argument) it is clear that the column player will not keep silent, that column becomes irrelevant to the row player. Thus the row player effectively faces the following game:

	C
S	-3,0
C	-2,-2

In this remaining game, confessing does once again dominate keeping silent for the row player. Thus, iterated dominance can solve this game completely.

Either strict or weak dominance can be used in the definition of iterated dominance. We note that the process of iterated dominance is never helped by removing a dominated mixed strategy, for the following reason. If σ'_i gives player i a higher utility than σ_i against mixed strategy σ_j for player $j \neq i$ (and strategies $\sigma_{-\{i,j\}}$ for the other players), then for at least one pure strategy s_j that σ_j places positive probability on, σ'_i must perform better than σ_i against s_j (and strategies $\sigma_{-\{i,j\}}$ for the other players). Thus, removing the mixed strategy σ_j does not introduce any new dominances.

7.1.3 Nash equilibrium

Many games cannot be solved using (iterated) dominance. Consider the following game (commonly called "chicken"):

	S	D
S	-2,-2	1,-1
D	-1,1	0,0

The story behind this game is the following: to test who has the strongest nerves, two drivers drive straight at each other, and at the last moment each driver must decide whether to continue straight (S) or dodge the other car by turning (say) right (D). The preferred outcome is to "win" by going straight when the other dodges, but if both drivers continue straight, they collide and both suffer severely.

This game has no dominated strategies. In fact, the matrix has multiple strategically stable entries: if one player goes straight, and the other dodges, then neither player has an incentive to change strategies (the player going straight is winning, and the player dodging does not want to go straight and collide). This leads to the definition of a *Nash equilibrium*:

Definition 32 *Given a normal-form game, a* Nash equilibrium *is vector of mixed strategies* $\sigma_1, \ldots, \sigma_n$ *such that no agent has an incentive to deviate from its mixed strategy given that the others do not deviate. That is, for any* i *and any alternative mixed strategy* σ_i'*, we have* $u_i(\sigma_1, \ldots, \sigma_i, \ldots, \sigma_n) \geq u_i(\sigma_1, \ldots, \sigma_i', \ldots, \sigma_n)$.

Indeed, (S, D) and (D, S) are pure-strategy Nash equilibria of "chicken." There is another Nash equilibrium where both players play each pure strategy with probability 0.5. Every finite game has at least one Nash equilibrium if we allow for mixed strategies [Nash, 1950].

7.2 Bayesian games

The normal-form representation of games assumes that players' utilities for outcomes of the game are common knowledge. Hence, they cannot directly capture settings in which the players' have *private information* about their utilities, as they would, for example, in an auction. Such settings can be modeled using *Bayesian games*.

In a Bayesian game, each player first receives privately held preference information (the player's *type*) from a distribution, which determines the utility that that player receives for every outcome of (that is, vector of actions played in) the game. After receiving this type, the player plays an action based on it.[2]

Definition 33 *A* Bayesian game *is given by a set of players* $\{1, 2, \ldots, n\}$*; and, for each player* i*, a set of actions* A_i*, a type space* Θ_i *with a probability distribution* p_i *over it, and a utility function* $u_i : \Theta_i \times A_1 \times \ldots \times A_n \to \mathbb{R}$ *(where* $u_i(\theta_i, a_1, \ldots, a_n)$ *denotes player* i*'s utility when* i*'s type is* θ_i *and each player* j *plays action* a_j*). A* pure strategy *in a Bayesian game is a mapping from types to actions,* $s_i : \Theta_i \to A_i$*, where* $s_i(\theta_i)$ *denotes the action that player* i *plays for type* θ_i.

As an example, consider an unusual first-price sealed-bid auction with two bidders, in which the bidders can only bid 1 or 2. If the bids are tied, then the winner is chosen randomly. Each bidder draws a valuation from $\Theta_1 = \Theta_2 = \{2, 2.5\}$ uniformly at random. We can represent the utility function of player 1 (the row player) as follows:

[2]In general, a player can also receive a signal about the other players' preferences, but we will not concern ourselves with that in this dissertation.

	bid 1	bid 2
bid 1	.5	0
bid 2	0	0

Row player utilities when $\theta_1 = 2$.

	bid 1	bid 2
bid 1	.75	0
bid 2	.5	.25

Row player utilities when $\theta_1 = 2.5$.

The utility function for the column player is similar.

Any vector of pure strategies in a Bayesian game defines an (expected) utility for each player, and therefore we can simply translate a Bayesian game into a normal-form game. For example, the auction game above gives (letting x, y denote the strategy of bidding x when one's type is 2, and y when one's type is 3):

	1,1	1,2	2,1	2,2
1,1	.625, .625	.3125, .5	.3125, .375	0, .25
1,2	.5, .3125	.3125, .3125	.3125, .1875	.125, .1875
2,1	.375, .3125	.1875, .3125	.1875, .1875	0, .1875
2,2	.25, 0	.1875, .125	.1875, 0	.125, .125

Using this transformation, we can take any solution concept that we have defined for normal-form games (such as dominance or Nash equilibrium), and apply it to Bayesian games. For example, in the game above, the strategy 2,1 is strictly dominated by 1,2. The strategy 2,2 is weakly dominated by 1,2. After removing 2,2 for both players, 1,1 weakly dominates every other strategy, so iterated weak dominance can solve this game entirely, leaving only 1,1 for each player. Both players playing 2,2 is nevertheless a Nash equilibrium so is both players playing 1,2; and both players playing 1,1. There are no mixed-strategy equilibria.

One remark that should be made is that the normal-form representation of the Bayesian game is exponentially larger than the original representation, because each player i has $|A_i|^{|\Theta_i|}$ distinct pure strategies. For the purpose of defining solution concepts and other conceptual purposes, this causes no problem. But, later, when we will be interested in computing Bayesian games' solutions, it will not be sufficient to simply apply this transformation and solve the normal form, since this will require exponential time (and space).

So, one can define solution concepts for Bayesian games by applying normal-form solution concepts to the normal-form representation of a Bayesian game. In spite of the simplicity of this approach, the typical approach in mechanism design is nevertheless to define the solution concepts directly, as is done below. For simplicity of notation, in the remainder of this chapter, I discuss pure strategies only; the generalizations to mixed strategies (where agents choose a distribution over actions based on their types) are straightforward.

First, let us consider a direct definition of dominance that is typically used in mechanism design:

Definition 34 *Given a Bayesian game, the vector of strategies $(s_1, \ldots, s_n)$ is a* dominant-strategy equilibrium *if for every agent i, for every type $\theta_i \in \Theta_i$, every alternative action $a_i \in A_i$, and every action vector $a_{-i} \in A_{-i}$ of the other agents, we have $u_i(\theta_i, s_i(\theta_i), a_{-i}) \geq u_i(\theta_i, a_i, a_{-i})$.*

There are a few differences between this definition and using the normal-form representation definition of dominance given above. First, this definition only applies to games where each agent has a strategy that dominates all others, *i.e.* dominance can solve the game entirely (without iteration). Second, none of the inequalities are required to be strict—this is *very weak* dominance. A third, subtle, minor difference is that in this definition the strategy is supposed to give an optimal action *for every type of the agent*, against any opponent actions. The definition that appeals to the normal-form representation only requires that the strategy maximizes the *total expected utility over the agent's types*, against any opponent actions. The normal-form definition still requires that the strategy chooses the optimal action for any type with positive probability; the only difference is that the normal-form definition does not require optimal actions to be chosen on types that have probability zero. For games with finitely many types, this is an insignificant difference, since it does not make sense to even bother defining a type that occurs with zero probability. Under continuous type spaces, the difference is a little more significant since the normal-form definition may choose to play in a bizarre manner on a set of types with measure zero. Since we will be mainly concerned with finite type spaces, the difference between the definitions is immaterial.

Now we will consider *Bayes-Nash equilibrium*, under which agents strategies are optimal only given the other agents' strategies, and given that one does not know the other agents' types.

Definition 35 *The vector of strategies* $(s_1, \ldots, s_n)$ *is a* Bayes-Nash equilibrium *if for every agent i, for every type* $\theta_i \in \Theta_i$*, and every alternative action* $a_i \in A_i$*, we have* $E_{\theta_{-i}}[u_i(\theta_i, s_i(\theta_i), s_{-i}(\theta_{-i}))] \geq E_{\theta_{-i}}[u_i(\theta_i, a_i, s_{-i}(\theta_{-i}))]$.

This definition is identical to the one where we simply apply Nash equilibrium to the normal form of the Bayesian game—with the exception that agents can no longer behave arbitrarily for types that have zero probability.

Now that we have some methods for predicting strategic behavior in arbitrary games, we can return to mechanism design and begin to assess the quality of mechanisms that are not truthful, direct-revelation mechanisms. In the next section, we will use this ability to prove two variants of the revelation principle, showing that *if* agents play according to the solution concepts defined here, then there is no reason not to use a truthful, direct-revelation mechanism.

7.3 Revelation principle

To prove the revelation principle, we first need to assess what outcomes will be produced by a mechanism that is not a truthful, direct-revelation mechanism, based on the solution concepts for Bayesian games given above. Such a mechanism can be represented by a set of actions A_i for each agent i, and an outcome selection function $o : A_1 \times \ldots \times A_n \rightarrow O$. (To minimize notational overhead, payments should be considered part of the outcome here. Also, the outcome function may in general produce distributions over outcomes; everything below can be extended to allow for this as well simply by replacing O with $\Delta(O)$.)

We first define when a mechanism *implements* a given *social choice rule*:

Definition 36 *A* social choice rule *is a function* $f : \Theta_1 \times \ldots \times \Theta_n \rightarrow O$*. A mechanism* o implements *rule* f in dominant strategies *if there is a dominant strategy equilibrium* $(s_1, \ldots, s_n)$ *such that for* *all* $(\theta_1, \ldots, \theta_n) \in \Theta_1 \times \ldots \times \Theta_n$, $o(s_1(\theta_1), \ldots, s_n(\theta_n)) = f(\theta_1, \ldots, \theta_n)$. *Similarly, a mechanism* o *implements rule* f in Bayes-Nash equilibrium *if there is a Bayes-Nash equilibrium* $(s_1, \ldots, s_n)$ *such that for all* $(\theta_1, \ldots, \theta_n) \in \Theta_1 \times \ldots \times \Theta_n$, $o(s_1(\theta_1), \ldots, s_n(\theta_n)) = f(\theta_1, \ldots, \theta_n)$.

One should note that a game may have multiple equilibria, and may therefore implement multiple social choice rules. For example, consider the two-type first-price auction example in the previous section: two of its equilibria always allocate the item at random, but the third allocates the item to the bidder with the higher valuation if the valuations are not equal. If there are multiple equilibria, then we will assume that we can choose our favorite equilibrium. This strengthens the power of indirect/non-truthful mechanisms, and therefore strengthens the revelation principle result below. (It should be remarked that truthful direct-revelation mechanisms may have multiple equilibria as well; however, one may argue that the truth-telling equilibrium is "focal", *i.e.* the most natural one.)

We are now ready to review two known variants of the revelation principle, corresponding to dominant-strategies equilibrium and Bayes-Nash equilibrium. Before doing so, recall the simple intuition behind the revelation principle: the new, truthful direct-revelation mechanism that we construct requests the agents' types, and then plays "on their behalf" in the old mechanism (according to the equilibrium of that mechanism) to produce the outcome. There is no reason for an agent to misreport his type, since this will only result in the new mechanism playing the part of that agent suboptimally in the old mechanism.

Revelation Principle, version 1 *Suppose there is an (indirect/non-truthful) mechanism that implements social choice rule* f *in dominant strategies. Then there exists a dominant-strategies incentive-compatible direct-revelation mechanism with outcome selection function* o *that also implements* f *in dominant strategies (using the truth-telling equilibrium).*

Proof: We show how to transform the given mechanism that implements f into a truthful direct-revelation mechanism that implements f. For each i, let $s_i^{old} : \Theta_i \rightarrow A_i^{old}$ be the strategy played by agent i in the equilibrium that implements f in the given mechanism, and let o^{old} be the given game's outcome selection function, so that $o^{old}(s_1^{old}(\theta_1), \ldots, s_n^{old}(\theta_n)) = f(\theta_1, \ldots, \theta_n)$, and the s_i^{old} constitute a dominant strategies equilibrium. Then let our new mechanism have the outcome function o given by $o(\theta_1, \ldots, \theta_n) = o^{old}(s_1^{old}(\theta_1), \ldots, s_n^{old}(\theta_n)) = f(\theta_1, \ldots, \theta_n)$. All we need to show is that truthtelling is a dominant strategies equilibrium. To show this, we observe that for any i and $\theta_i \in \Theta_i$, for any alternative type $\hat{\theta}_i \in \Theta_i$, and for any $\theta_{-i} \in \Theta_{-i}$, $u_i(\theta_i, o(\theta_i, \theta_{-i})) = u_i(\theta_i, o^{old}(s_i^{old}(\theta_i), s_{-i}^{old}(\theta_{-i}))) \geq u_i(\theta_i, o^{old}(s_i^{old}(\hat{\theta}_i), s_{-i}^{old}(\theta_{-i}))) = u_i(\theta_i, o(\hat{\theta}_i, \theta_{-i}))$, where the inequality derives from the fact that the s_i^{old} constitute a dominant strategies equilibrium in the original mechanism. ∎

Revelation Principle, version 2 *Suppose there is an (indirect/non-truthful) mechanism that implements social choice rule* f *in Bayes-Nash equilibrium. Then there exists a Bayes-Nash equilibrium incentive-compatible direct-revelation mechanism with outcome selection function* o *that also implements* f *in Bayes-Nash equilibrium (using the truth-telling equilibrium).*

Proof: We show how to transform the given mechanism that implements f into a truthful direct-revelation mechanism that implements f. For each i, let $s_i^{old} : \Theta_i \rightarrow A_i^{old}$ be the strategy played by agent i in the equilibrium that implements f in the given mechanism, and let o^{old} be the given game's outcome selection function, so that $o^{old}(s_1^{old}(\theta_1), \ldots, s_n^{old}(\theta_n)) = f(\theta_1, \ldots, \theta_n)$, and the s_i^{old} constitute a Bayes-Nash equilibrium. Then let our new mechanism have the outcome function o given by $o(\theta_1, \ldots, \theta_n) = o^{old}(s_1^{old}(\theta_1), \ldots, s_n^{old}(\theta_n)) = f(\theta_1, \ldots, \theta_n)$. All we need to show is that truthtelling is a Bayes-Nash equilibrium. To show this, we observe that for any i and $\theta_i \in \Theta_i$, for any alternative type $\hat{\theta}_i \in \Theta_i$, $E_{\theta_{-i}}[u_i(\theta_i, o(\theta_i, \theta_{-i}))] = E_{\theta_{-i}}[u_i(\theta_i, o^{old}(s_i^{old}(\theta_i), s_{-i}^{old}(\theta_{-i})))] \geq E_{\theta_{-i}}[u_i(\theta_i, o^{old}(s_i^{old}(\hat{\theta}_i), s_{-i}^{old}(\theta_{-i})))] = E_{\theta_{-i}}[u_i(\theta_i, o(\hat{\theta}_i, \theta_{-i}))]$, where the inequality derives from the fact that the s_i^{old} constitute a Bayes-Nash equilibrium in the original game. ■

We have assumed that the strategies in the equilibrium of the original mechanism are pure; the result can be extended to the setting where they are mixed. In this case, though, the resulting truthful mechanism may become randomized, even if the original mechanism was not.

7.4 Summary

In this chapter we reviewed basic concepts from game theory. We reviewed basic solution concepts for normal-form games, including minimax strategies, dominance and iterated dominance, and Nash equilibrium. We then showed how to extend these solution concepts to Bayesian games. Armed with these concepts, we finally presented the (known) proofs of two variants of the *revelation principle*, which (informally stated) show that if agents act strategically (according to these solution concepts), then there is no reason not to use a truthful, direct-revelation mechanism.

Unfortunately, as we will see in the next chapters, the assumption that agents will behave in a strategically optimal way is often untenable in mechanisms for expressive preference aggregation. This is in part due to the fact that the agents' strategy spaces become too large to search exhaustively. Of course, exhaustive search is not necessarily required to behave in a strategically optimal way—perhaps there are efficient algorithms that home in on the optimal strategies quickly. In Chapter 8 we show that for some settings, this is unlikely to be the case, because even the problem of finding a best response to given strategies by the other players is computationally hard (NP-complete or harder). Additionally, intuitively, the problem of computing a best response is much easier than that of acting optimally when the other agents' actions are not yet known, and must be reasoned about first. In Chapter 9 we show that indeed, standard solution concepts such as (iterated) dominance and Nash equilibrium can be hard to compute (even when the strategy spaces are much more manageable in size).

Chapter 8

Mechanism Design for Bounded Agents

Any fool can tell the truth, but it requires a man of some sense to know how to lie well.

Samuel Butler

Mechanism design has traditionally taken the conservative view that agents will always choose the actions that are in their own best interest—the assumption of *perfect rationality*. Specifically, the revelation principle discussed in the previous chapter relies heavily on this assumption. However, this may be an overly conservative assumption in that agents may not always have the computational resources to find the action that is in their own best interest—their rationality is *bounded*. For instance, bidding optimally in a reverse auction for trucking tasks may require the bidder to solve multiple NP-complete vehicle routing problems [Sandholm and Lesser, 1997]. With this in mind, we can ask questions such as:

- Can *impossibility results* in mechanism design that rely on the assumption of perfect rationality be circumvented if agents have limited computational resources?
- Given that the revelation principle ceases to apply when agents' rationality is bounded, are there benefits to using *non-truthful* mechanisms (even when reasonable truthful mechanisms exist)?[1]

It is impossible to answer these questions without some characterization of how the agents' rationality is bounded. Previous work by Larson and Sandholm relies on explicitly modeling the agents' computational choices to derive direct tradeoffs between the cost of additional computation and the benefits of additional computation to the solution [Larson and Sandholm, 2001a, 2005]. In contrast,

[1] Various research has proposed the use of mechanisms that are only *approximately truthful*. These can be easier to execute [Kothari *et al.*, 2003; Archer *et al.*, 2003], or their use can be motivated by impossiblity results that apply to truthful mechanisms [Parkes *et al.*, 2001; Goldberg and Hartline, 2003]. For approximately truthful mechanisms, the idea is not that it will necessarily be computationally difficult for agents to act optimally, but rather that the incentives for the agents to act optimally (rather than simply tell the truth) are somehow too small for the agents to respond to them. However, if the agents do respond to these slight incentives, the desirable properties of the mechanism may unravel completely.

this chapter will not require specific models of how the agents do their computation. Rather, it relies on classic complexity-theoretic notions to determine whether it is hard for the agents to find strategically optimal actions.

The rest of this chapter is layed out as follows. In Section 8.1, we show that there are settings where using the optimal truthful mechanism requires the center to solve a hard computational problem; but there is another, non-truthful mechanism, under which the center does not have to solve any hard optimization problem, but the problem of finding a beneficial manipulation is hard for one of the agents. Moreover, if the agent manages to find the manipulation, the produced outcome is the same as that of the best truthful mechanism; and if the agent does not manage to find it, the produced outcome is strictly *better* [Conitzer and Sandholm, 2004c]. In Section 8.2, we show that adding a *preround* to voting rules can make manipulation of these voting rules computationally much harder [Conitzer and Sandholm, 2003g]. However, those hardness results (as well as others) rely on the number of candidates being unbounded. In Section 8.3, we show that if we consider *coalitional* manipulation by *weighted* voters, then we can get hardness even with constant numbers of candidates [Conitzer and Sandholm, 2002a; Conitzer *et al.*, 2003]. Unfortunately, all of the above hardness results only prove hardness in the worst case (as is common in complexity theory). In Section 8.4, we give an impossibility result that makes it appear unlikely that voting rules can be constructed that are *usually* hard to manipulate [Conitzer and Sandholm, 2006f].

8.1 A failure of the revelation principle with bounded agents

As we have seen in earlier chapters, in many real-world mechanism design settings, the center faces an intractable optimization problem in trying to execute the mechanism. In this section, we question the focus on truthful mechanisms when the setting requires the solution of computationally hard problems. In particular, we show that there are settings where by abandoning truthful mechanisms, we can shift a computationally hard problem from the center to one of the agents. Additionally, whereas not being able to cope with the issue of computational hardness would have hurt the center in achieving its objective, if the agent is unable to cope with it, this actually helps the designer in achieving its objective.

We first observe that dominant strategy implementation and Bayes-Nash implementation differ only on what agents can be expected to know about each other's types and actions. An interesting special case is that of games where only one agent needs to choose an action. In this case, the acting agent always knows everything there is to know about the other agents' actions (namely, nothing). So, both solution concepts coincide here. We prove the remaining two theorems for this types of game, so the results hold both for dominant strategy implementation and Bayes-Nash implementation.

Theorem 51 *Suppose that the center is trying to maximize social welfare, and neither payments nor randomization are allowed.*[2] *Then, even with only two agents (one of whom does not even report a type, so dominant strategy implementation and Bayes-Nash implementation coincide), there exists a family of preference aggregation settings such that:*

[2] It is not immediately clear if this result can be extended to cases with payments or randomization; we leave this as a question for future research.

- *the execution of any optimal truthful mechanism is NP-complete for the center, and*
- *there exists a non-truthful mechanism which 1) requires the center to carry out only polynomial computation, and 2) makes finding any beneficial insincere revelation NP-complete for the type-reporting agent. Additionally, if the type-reporting agent manages to find a beneficial insincere revelation, or no beneficial insincere revelation exists, the social welfare of the outcome is identical to the social welfare that would be produced by any optimal truthful mechanism. Finally, if the type-reporting agent does not manage to find a beneficial insincere revelation where one exists, the social welfare of the outcome is* strictly greater *than the social welfare that would be produced by any optimal truthful mechanism.*

Put in perspective, the mechanism designer would reap two benefits from using the second, non-truthful mechanism rather than a truthful mechanism:

- Doing so shifts the computational hardness from the center to the agent. This can also be seen as a statement about how the social welfare that can be obtained by truthful mechanisms compares to the social welfare that can be obtained by non-truthful mechanisms, as follows. If it is computationally infeasible to execute the optimal truthful mechanism, the designer might resort to another truthful mechanism which merely approximates the social welfare obtained by the optimal truthful mechanism (this approach is often advocated in algorithmic mechanism design).
- If the agent cannot consistently solve instances of an NP-complete problem, then, even if the agent is trying to act strategically, using the second mechanism improves social welfare in some cases (and never decreases it).

Hence, (by the argument under the second bullet) the non-truthful mechanism—which is computationally feasible to execute—outperforms the optimal truthful mechanism, which (by the argument under the first bullet) in turn outperforms any computationally feasible truthful mechanism.

We emphasize that we do *not* require that agents will never be able to solve an NP-complete problem. Our result is more cautious than that: if agents do solve the NP-complete problem, nothing is lost; whereas if they do not solve it, something is gained.

Another point is that individual rationality is still maintained under this approach, by making sure that telling the truth still guarantees an agent nonnegative utility (even if telling the truth is not strategically optimal).

We are now ready to give the proof.

Proof: We are given a graph $G = (V, E)$ (with at least some edges); the outcome space is the set of all subsets of size k of the vertices, $\{X \subseteq V : |X| = k\}$. The type-reporting agent (agent 1) has the following type set Θ. For each $X \subseteq V$ with $|X| = k$, there is a type θ_X which occurs with probability $1/(\binom{n}{k} + 1)$. The utility function for these types is as follows: $u_1(\theta_X, X) = 4$ if X is an independent set (that is, there are no edges within X); $u_1(\theta_X, X) = 3$ if X is not an independent set; $u_1(\theta_X, Y) = 1$ if $X \neq Y$ and Y is an independent set; and $u_1(\theta_X, Y) = 0$ if $X \neq Y$ and Y is not an independent set. Additionally, there is a single additional type θ_0 which occurs with probability $1/(\binom{n}{k} + 1)$, and the utility function for it is given as follows: $u_1(\theta_0, X) = 1$ if X is an independent set; and $u_1(\theta_0, X) = 0$ if X is not an independent set. Agent 2, who does not report a type, has the following utility function: $u_2(X) = 2$ if X is not an independent set; and $u_2(X) = 0$ if X is an independent set. Now let us consider creating a mechanism for such a setting that uses neither payments nor randomization.

First, we claim that all optimal *truthful* mechanisms are of the following form.

- If agent 1 reports a type θ_X, then choose outcome X;
- If agent 1 reports type θ_0, then 1) if there exists an independent set $X \subseteq V$ with $|X| = k$, choose such an independent set; or 2) if no independent set exists, choose any $X \subseteq V$ with $|X| = k$.

It is straightforward to verify that mechanisms of this form act in agent 1's best interest, that is, they always choose one of the outcomes that are optimal for agent 1 given its type. Hence, agent 1 never has any incentive to misreport its type, so these mechanisms are truthful. All that remains to show is that all other truthful mechanisms have strictly less expected social welfare than these. We first observe that the only case in which we get less than the optimal social welfare with the mechanisms of the given form is when agent 1 has type θ_0, and an independent set of size k exists. In this case, the mechanisms of the given form choose an independent set as the outcome, leading to a social welfare of 1; whereas a social welfare of 2 could have been obtained by choosing a set that is not independent. It follows that the expected social welfare that we get from one of the mechanisms of the given form is at most $\frac{1}{\binom{n}{k}+1}$ below the maximal expected social welfare that we could have obtained if the agents did not play strategically. Now consider an alternative truthful mechanism that, for some $X \subseteq V$ with $|X| = k$, does not choose X when agent 1 reports θ_X. In this case, this mechanism can obtain a social welfare of at most 2, whereas the optimal social welfare in this case is at least 4. It follows that the expected social welfare that we get from this mechanism is at least $\frac{2}{\binom{n}{k}+1}$ below the maximal expected social welfare that we could have obtained if the agents did not play strategically. Hence, all optimal truthful mechanisms always choose X when agent 1 reports θ_X. But then, if an independent set exists in G, an optimal truthful mechanism must choose such an independent set in the case where agent 1 reports θ_0: because if it does not, then when agent 1 has this type, it would benefit from misreporting its type as a type corresponding to the independent set—and the mechanism would no longer be truthful. Thus, we have established that all optimal truthful mechanisms are of the given form. We observe that executing such a mechanism requires solving an NP-complete problem, because we have to construct an independent set if it exists, which is NP-complete.

Now consider the following mechanism:

- If agent 1 reports a type θ_X, then choose outcome X;
- If agent 1 reports type θ_0, then choose some $X \subseteq V$ with $|X| = k$ that is not an independent set.

We observe that this mechanism is computationally easy to execute. Also, this mechanism is not truthful if there is an independent set, because in this case, if agent 1 has type θ_0, it would be better off reporting the type corresponding to the independent set. However, there are no other beneficial insincere revelations. Thus, it is straightforward to verify that if agent 1 always reports the type that is strategically optimal for it, the outcome of this mechanism is always identical to that of one of the optimal truthful mechanisms. Of course, in order for agent 1 to always report the type that is strategically optimal for it, agent 1 needs to construct an independent set (if possible) when it has type θ_0. Because this problem is NP-complete, it is reasonable to suspect that agent 1 will not always be able to construct such a set even when it exists. If agent 1 indeed fails to construct an independent set in this case, the outcome will be some $X \subseteq V$ with $|X| = k$ that is not an independent set. This outcome actually has a social welfare of 2, as opposed to the social welfare of 1 that would have been obtained if agent 1 had managed to construct an independent set. Hence the social welfare is strictly greater than in the case where agent 1 has unlimited computational power; and hence it is also a greater than it would have been with an optimal truthful mechanism. ■

Given this example setting in which the revelation principle "fails" in the sense that non-truthful mechanisms can outperform truthful ones,[3] one may wonder whether similar phenomena occur in other, more standard settings. In the remainder of this chapter, we will study whether this is so for voting settings.

8.2 Tweaking voting protocols to make manipulation hard

Early, seminal work on the complexity of manipulating elections demonstrated that several voting rules are hard to manipulate, including the second-order Copeland rule [Bartholdi *et al.*, 1989a] and the STV rule [Bartholdi and Orlin, 1991]. In this section we take the next step of designing new protocols that are especially hard to manipulate. Rather than designing these protocols from scratch, we show how to tweak existing voting protocols to make manipulation computationally much more difficult, while leaving much of the original nature of the protocol intact, for the following reasons:

- Results on the computational complexity induced by a tweak typically apply to a large family of protocols.
- Some of the original protocol's nice theoretical properties are preserved by the tweak. For example, if a protocol satisfies the *Condorcet criterion* (a candidate that wins all its pairwise elections always wins the election), the tweak will preserve this property.
- In practice, it will be much easier to replace a currently used protocol with a tweaked version of it, than with an altogether new protocol.

The type of tweak we introduce is the following. All the candidates are paired in a *preround*; of each pair of candidates, only the winner of their pairwise election survives. (Recall that the winner of the pairwise election between two candidates is the candidate that is ranked above the other more often in the votes.) After the preround, the original protocol is executed on the remaining candidates. The *schedule* of the preround (i.e., who faces who) can be determined before the votes are collected; after the votes are collected; or while the votes are collected (the processes are interleaved). We study these three cases in Subsections 8.2.2, 8.2.3, and 8.2.4, respectively.

[3] Of course, the principle does not fail in the sense that the formal statements of it are wrong; it is merely that the preconditions of the theorem fail to hold when agents are computationally bounded.

8.2.1 Definitions

Voting protocols

For the purposes of this section, a *deterministic* protocol is a protocol of the type that we have considered earlier in this dissertation, that is, a function from the set of all combinations of votes to C. (We will only be interested in the winner of the election in this section, not in an entire ranking of candidates.) A *randomized* protocol is a function from the set of all combinations of votes to probability distributions over C. An *interleaved* protocol is a procedure for alternating between collecting (eliciting) parts of the voters' votes (*e.g.* whether they prefer candidate a to candidate b) and drawing and publishing random variables (such as parts of the schedule for an election), together with a function from the set of all combinations of votes and random variables to C.

Preround

The tweaks we study in this section all involve the addition of a preround. We will now define how this works.

Definition 37 *Given a protocol P, the new protocol obtained by adding a preround to it proceeds as follows:*

1. *The candidates are paired. If there is an odd number of candidates, one candidate gets a bye.*
2. *In each pairing of two candidates, the candidate losing the pairwise election between the two is eliminated. A candidate with a bye is never eliminated.*
3. *On the remaining candidates, P is executed to produce a winner. For this, the implicit votes over the remaining candidates are used. (For example, if a voter voted $a \succ b \succ c \succ d \succ e$, and b and c were eliminated, the voter's implicit vote is $a \succ d \succ e$.)*

The pairing of the candidates is also known as the schedule *for the preround. If the schedule is decided and published before the votes are collected, we have a* deterministic preround *($DPRE$), and the resulting protocol is called $DPRE + P$. If the schedule is drawn completely randomly after the votes are collected, we have a* randomized preround *($RPRE$), and the resulting protocol is called $RPRE + P$. Finally, if the votes are elicited incrementally, and this elicitation process is interleaved with the scheduling-and-publishing process (which is again done randomly), as described in detail in Subsection 8.2.4, we have an* interleaved preround *($IPRE$), and the resulting protocol is called $IPRE + P$.*

Manipulation

We now define the computational problem of manipulation that we study in this section. Other definitions of manipulation are possible: in the next section, we will give a more thorough analysis of the different variants of the manipulation problem, and study some of the other variants.

Definition 38 (CONSTRUCTIVE-MANIPULATION) *We are given a protocol P, a candidate set C, a preferred candidate p, and a set of votes S corresponding to all the other voters' votes.*

The manipulator has yet to decide on its vote, and wants to make p win. Then the constructive manipulation question is:

- *(For deterministic protocols) Can the manipulator cast its vote to make p win under P?*
- *(For randomized protocols) Can the manipulator cast its vote to make the probability of p winning under P at least some given $k \in [0, 1]$?*
- *(For interleaved protocols) Given the initial random choices (if any) by the protocol, is there a contingency plan (based on the random decisions the protocol takes between eliciting parts of the votes) for the manipulator to answer the queries to make the probability of p winning under P at least some given $k \in [0, 1]$?*

8.2.2 NP-hardness when scheduling precedes voting

In this subsection, we examine the complexity induced by the preround when the voters know the schedule before they vote.

A sufficient condition for NP-hardness

We present a sufficient condition under which adding a preround with a preannounced schedule makes manipulation NP-hard. The condition can be thought of as an NP-hardness reduction template. If it is possible to reduce an arbitrary SAT instance to a set of votes satisfying certain properties under the given voting protocol, that protocol—with a preround—is NP-hard to manipulate.

Theorem 52 *Given a voting protocol P, suppose that it is possible, for any Boolean formula ϕ in conjunctive normal form (i.e., a SAT instance), to construct in polynomial time a set of votes over a candidate set containing at least $\{p\} \cup C_L$ where $C_L = \{c_l : l \in L\}$ (L is the set of literals $\{+v : v \in V\} \cup \{-v : v \in V\}$, where V is the set of variables used in ϕ), with the following properties:*

- *(Property 1a) If we remove, for each $v \in V$, one of c_{+v} and c_{-v}, p would win an election under protocol P against the remaining candidates if and only if for every clause $k \in K$ (where K is the set of clauses in ϕ), there is some $l \in L$ such that c_l has not been removed, and l occurs in k. This should hold even if a single arbitrary vote is added.*
- *(Property 1b) For any $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election after these votes.*

Then CONSTRUCTIVE-MANIPULATION in $DPRE + P$ is NP-hard (and NP-complete if P is deterministic and can be executed in polynomial time).

Proof: Consider the following election under $DPRE + P$. Let the candidate set be the set of all candidates occurring in the votes constructed from ϕ (the "original candidates"), plus one dummy candidate for each of the original candidates besides those in C_L. To each of the constructed votes, add all the dummy candidates at the bottom; let the resulting set of votes be the set of the nonmanipulators' votes. A single manipulator's vote is yet to be added. Let the schedule for the preround be as follows: for each v, c_{+v} and c_{-v} face each other in the preround; and every other original candidate faces (and, because of the dummy candidates' position in the votes, defeats) a dummy candidate. Thus, the set of candidates that make it through the preround consists of, for each $v \in V$, one of c_{+v} and c_{-v}; and all the other original candidates. The manipulator's vote will decide the winner of every c_{+v} vs. c_{-v} match-up, because by property 1b, all these pairwise elections are currently tied. Moreover, it is easy to see that the manipulator can decide the winner of each of these match-ups independently of how it decides the winners of the other match-ups. Thus, we can think of this as the manipulator giving the variables truth-values: v is set to *true* if c_{+v} survives, and to *false* if c_{-v} survives. By property 1a it then follows that p wins if and only if the manipulator's assignment satisfies all the clauses, i.e. is a solution to the SAT instance. Hence there is a successful constructive manipulation if and only if there is a solution to the SAT instance, and it follows that CONSTRUCTIVE-MANIPULATION in $DPRE + P$ is NP-hard. (It is also in NP if P is deterministic and can be executed in polynomial time, because in this case, given a vote for the manipulator, it can be verified in polynomial time whether this vote makes p win). ■

Examples

We now show how to apply Theorem 52 to the well-known protocols we discussed, thus showing that each of these protocols—with a preround—is NP-hard to manipulate.

Theorem 53 *There exists a reduction that satisfies properties 1a and 1b of Theorem 52 under the plurality rule.*

When it does not matter for our proofs whether a given vote is $a \succ b \succ c$ or $b \succ a \succ c$, we write $\{a, b\} \succ c$.

Proof: Given the formula ϕ, let the candidate set be the minimally required candidates $\{p\} \cup C_L$, plus a set of candidates corresponding to the set of clauses K of ϕ, $C_K = \{c_k : k \in K\}$. Then, let the set of votes be as follows: $4|K| + 2$ votes ranking the candidates $p \succ C_L \succ C_K$; for each $k \in K$, $4|K|$ votes ranking the candidates $c_k \succ \{c_{cl} \in C_K : cl \neq k\} \succ C_L \succ p$; and for each $k \in K$, 4 votes ranking the candidates $\{c_l \in C_L : l \in k\} \succ c_k \succ \{c_l \in C_L : l \notin k\} \succ \{c_{cl} \in C_K : cl \neq k\} \succ p$. Additionally, we require that these votes are such that after counting them, for each $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election, so that property 1b is satisfied. (This is possible because the total number of votes is even, and the majority of the votes do not yet have any restrictions on the order of the C_L.) We now show property 1a is satisfied. We first observe that regardless of which of the candidates corresponding to literals are removed, p will get $4|K| + 2$ votes. Now, if for some $k \in K$, all the candidates c_l with $l \in L, l \in k$ are removed, then c_k will get at least $4|K| + 4$ votes and p will not win. On the other hand, if for each $k \in K$, at least one candidate c_l with $l \in k$ remains, then each of the c_k will get precisely $4|K|$ votes. Because each remaining c_l can get at most $4|K|$ votes as well, p will win. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. Thus, property 1a is satisfied. ■

Theorem 54 *There exists a reduction that satisfies properties 1a and 1b of Theorem 52 under the Borda rule.*

Proof: Given the formula ϕ, let the candidate set be the minimally required candidates $\{p\} \cup C_L$; plus a set of candidates corresponding to the set of clauses K of ϕ, $C_K = \{c_k : k \in K\}$, which we order in some arbitrary way to get $\{c_1, \dots, c_{|K|}\}$. Let M be the total number of candidates this defines. Then, let the set of votes be as follows: for every $c_i \in C_K$, $4M$ votes ranking the candidates $c_{i+1} \succ c_{i+2} \succ \dots \succ c_{|K|} \succ p \succ c_1 \succ c_2 \succ \dots \succ c_{i-1} \succ \{c_l \in C_L : l \in c_i\} \succ c_i \succ \{c_l \in L : l \notin c_i\}$; (here, the slight abuse of notation $l \in c_i$ means that l occurs in the clause corresponding to c_i;) $4M$ votes ranking the candidates $c_1 \succ c_2 \succ \dots \succ c_{|K|} \succ p \succ C_L$; one vote $c_1 \succ c_2 \succ \dots \succ c_{|K|} \succ C_L \succ p$; one vote $c_{|K|} \succ c_{|K|-1} \succ \dots \succ c_1 \succ C_L \succ p$; and finally, $4|K|M$ votes ranking the candidates $p \succ c_1 \succ c_2 \succ \dots \succ c_n \succ C_L$, and $4|K|M$ votes ranking the candidates $c_n \succ c_{n-1} \succ \dots c_1 \succ p \succ C_L$. Additionally, we require that these votes are such that after counting them, for each $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election, so that property 1b is satisfied. (This is possible because the total number of votes is even, and the majority of the votes do not yet have any restrictions on the order of the c_l.) We now show property 1a is satisfied. It is easy to see that none of the c_l can win, regardless of which of them are removed. Thus, we only need to consider the c_i and p. The last $8|K|M$ votes will have no net effect on the relative scores of these candidates, so we need not consider these here. After the first $4(|K|+1)M$ votes, any c_k for which all the c_l with $l \in k$ have been removed will be tied with p, and any other c_k will be at least $4M$ points behind p. Finally, from the last remaining two votes, any c_k ($k \in K$) will gain $2M - 2|V| - |K| - 1$ points on p. It follows that p wins if and only if for every clause $k \in K$, there is some $l \in L$ with $l \in k$ such that c_l has not been removed. In both cases there is a "margin" of at least $M - |V|$ points, so a single additional vote will not change this. Thus, property 1a is satisfied. ■

Theorem 55 *There exists a reduction that satisfies properties 1a and 1b of Theorem 52 under the maximin rule.*

Proof: Given the formula ϕ, let the candidate set be the minimally required candidates $\{p\} \cup C_L$, plus a set of candidates corresponding to the set of clauses K of ϕ, $C_K = \{c_k : k \in K\}$. Then, let the set of votes be as follows: $8|K|$ votes ranking the candidates $p \succ C_L \succ C_K$, $8|K|$ votes ranking the candidates $C_L \succ C_K \succ p$, and $8|K|$ votes ranking the candidates $C_K \succ p \succ C_L$; $4|K|$ votes ranking the candidates $C_L \succ p \succ C_K$, $4|K|$ votes ranking the candidates $C_K \succ C_L \succ p$, and, for each $k \in K$, 4 votes ranking the candidates $p \succ \{c_{cl} \in C_K : cl \neq k\} \succ \{c_l \in C_L : l \in k\} \succ c_k \succ \{c_l \in C_L : l \notin k\}$; and finally, 2 votes ranking the candidates $p \succ C_K \succ C_L$, and 2 votes ranking the candidates $C_K \succ p \succ C_L$. Additionally, we require that these votes are such that after counting them, for each $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election, so that property 1b is satisfied. (This is possible because the total number of votes is even, and the majority of the votes do not yet have any restrictions on the order of the c_l.) We now show property 1a is satisfied. Regardless of which of the candidates corresponding to literals are removed, p's worst score in a pairwise election is against any of the c_k, namely $16|K| + 2$. Any c_k for which all the c_l with $l \in k$ have been removed will get its worst pairwise election score against any of the C_L, namely $16|K| + 4$. Finally, any other c_k will get its worst pairwise election score against one of the c_l with $l \in k$, namely, $16|K|$. It follows that p wins if and only if for every clause $k \in K$, there is some $l \in k$ such that c_l has not been removed. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. Thus, property 1a is satisfied. ■

Theorem 56 *There exists a reduction that satisfies properties 1a and 1b of Theorem 52 under the STV rule.*

Proof: Given the formula ϕ, let the candidate set be the minimally required candidates $\{p\} \cup C_L$, plus a set of candidates corresponding to the set of clauses K of ϕ, $\{c_{cl} : cl \in K\}$, which we order in some arbitrary way to get $\{c_1, \ldots, c_{|K|}\}$; plus $4|K|$ additional candidates $c_{a_1}, \ldots, c_{a_{8|K|}}$. Then, let the set of votes be as follows: for each $k \in K$, 4 votes ranking the candidates $\{c_l \in C_L : l \in k\} \succ c_k \succ \{c_l \in C_L : l \notin k\} \succ p \succ \{c_{a_j}\}$; for each c_{a_i}, 2 votes ranking the candidates $c_{a_i} \succ c_1 \succ c_1 \succ \ldots \succ c_{|K|} \succ p \succ C_L \succ \{c_{a_j} : j \neq i\}$; and finally, 4 votes ranking the candidates $p \succ C_K \succ C_L \succ \{c_{a_j}\}$. Additionally, we require that these votes are such that after counting them, for each $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election, so that property 1b is satisfied. (This is possible because the total number of votes is even, and the majority of the votes do not yet have any restrictions on the order of the C_L.) We now show property 1a is satisfied. Regardless of which of the candidates corresponding to literals are removed, p will have 4 votes initially, and every c_{a_j} will have 2 votes initially. Any c_k ($k \in K$) for which all the c_l ($l \in L$) with $l \in k$ have been removed will have 4 votes initially. Any other c_k will have 0 votes initially, and hence drop out in the first round. Then, before p or any more c_k drop out, all the c_{a_j} will drop out, because they have only 2 votes initially and no votes will transfer to them. All the $8|K|$ votes that the c_{a_j} have initially will transfer either to the c_i that has the lowest index i among the remaining c_{cl}, or, if there are no remaining c_{cl}, to p. Because these $8|K|$ votes are the majority of votes in the election, it follows that the candidate to which all of these votes transfer will win the election. It follows that p wins if and only if for every clause $k \in K$, there is some $l \in L$ with $l \in k$ such that c_l has not been removed. In both cases there is a "margin" of at least 2 in every round, so a single additional vote will not change this. Thus, property 1a is satisfied. ■

Theorem 57 *In any of $DPRE$+$plurality$, $DPRE$+$Borda$, $DPRE$+$maximin$, and $DPRE$+STV[4] , CONSTRUCTIVE-MANIPULATION is NP-complete.*

Proof: NP-hardness is immediate from the previous theorems. The problem is in NP because these protocols can be executed in polynomial time. ■

In the next subsections, we will raise the bar and bring the problem of manipulating elections to higher complexity classes by abandoning the assumption that the schedule for the preround should be known in advance.

[4]The NP-completeness of manipulating $DPRE + STV$ is, in itself, not that interesting, because STV is already NP-hard to manipulate without the preround as we discussed. Nevertheless, our method highlights a different aspect of the NP-hardness of manipulating $DPRE + STV$. We build on this reduction later to prove PSPACE-hardness of manipulating STV with a preround when the scheduling of the preround is interleaved with the vote elicitation.

8.2.3 #P-hardness when voting precedes scheduling

In this subsection, we will examine the complexity induced by the preround when the schedule is drawn completely (uniformly) randomly after all the votes have been collected.

A sufficient condition for #P-hardness

We present a sufficient condition for a voting protocol to become #P-hard[5] to manipulate in this setting. Again, this condition can be thought of as a reduction template. If it is possible to reduce an arbitrary PERMANENT instance to a set of votes satisfying certain properties under the given voting protocol, that protocol is #P-hard to manipulate when a randomized preround is added to it. (In the PERMANENT problem, we are given a bipartite graph B with the same number of vertices k in both parts, and are asked how many matchings there are. This problem is #P-complete [Valiant, 1979].)

Theorem 58 *Given a voting protocol P, suppose that it is possible, for any bipartite graph B with the same number of vertices k in both parts (labeled 1 to k in one part, $k+1$ to $2k$ in the other), to construct in polynomial time a set of votes over the candidate set $\{c_1, \ldots, c_{2k}, p\}$ (where c_i corresponds to vertex i in B) with the following properties:*

- *(Property 2a) If we remove k of the c_i, p would win an election under protocol P against the remaining c_i if and only if the removed c_i are exactly all the c_i with $k+1 \leq i \leq 2k$;*
- *(Property 2b) p loses its pairwise election against all c_i with $k+1 \leq i \leq 2k$;*
- *(Property 2c) For any $1 \leq i \leq k$ and $k+1 \leq j \leq 2k$, c_i defeats c_j in their pairwise election if and only if in B, there is an edge between vertices i and j.*
- *(Property 2d) All the previous properties still hold with any additional single vote.*

Then CONSTRUCTIVE-MANIPULATION in $RPRE + P$ is #P-hard.

Proof: Given the set of votes constructed on the basis of an arbitrary B, let us compute the probability that p wins under the protocol $RPRE + P$ with only these votes. In the preround, there are k matches and one bye. By property 2a, p will win the election if and only if the k candidates eliminated in this preround are precisely all the c_i with $k+1 \leq i \leq 2k$. By property 2b, p could not win a preround match against any of these, so p will win the election if and only if it gets the bye, and each of the c_j with $k+1 \leq j \leq 2k$ faces one of the c_i with $1 \leq i \leq k$ that defeats it in the preround. Then, by property 2c, it follows that p wins if and only if the preround pairing corresponds to a matching in B. Thus the probability of p winning is $\frac{m_B}{e(2k,2k+1)}$, where m_B is the number of matchings in B and $e(2k, 2k+1)$ is the number of different ways to pair $2k$ of the $2k+1$ candidates in the preround (which is straightforward to compute). Thus, evaluating p's chances of winning in this election is at least as hard as counting the number of matchings in an arbitrary B, which is #P-hard. Moreover, because we can compute p's chances of winning solely

[5] #P is the class of problems where the task is to count the number of solutions to a problem in NP.

on the basis of properties 2a, 2b, and 2c, and by property 2d, these properties are maintained for any single additional vote, it follows that a manipulator cannot affect p's chances of winning. Thus, CONSTRUCTIVE-MANIPULATION in this case simply comes down to computing p's chances of winning, which is #P-hard as demonstrated. ■

A broadly applicable reduction

In this subsubsection we present a single broadly applicable reduction which will satisfy the precondistions of Theorem 58 for many voting protocols, thus proving them #P-hard to manipulate when the voting precedes the preround scheduling.

Definition 39 *We label the following reduction R_1. Given a bipartite graph B with the same number of vertices k in both parts (labeled 1 to k in one part, $k+1$ to $2k$ in the other), we construct the following set of $12k^3+2k^2$ votes:*

- *$6k^3$ votes that rank the candidates $c_{k+1} \succ c_{k+2} \succ \ldots \succ c_{2k} \succ p \succ c_1 \succ c_2 \succ \ldots \succ c_k$;*
- *$3k^2$ votes that rank the candidates $p \succ c_k \succ c_{k-1} \succ \ldots \succ c_1 \succ c_{2k} \succ c_{2k-1} \succ \ldots \succ c_{k+1}$;*
- *$6k^3-3k^2$ votes that rank the candidates $c_k \succ c_{k-1} \succ \ldots \succ c_1 \succ c_{2k} \succ c_{2k-1} \succ \ldots \succ c_{k+1} \succ p$;*
- *For each edge (i,j) in B ($1 \leq i \leq k$, $k+1 \leq j \leq 2k$), one vote that ranks the candidates $c_i \succ c_j \succ p \succ c_1 \succ c_2 \succ \ldots \succ c_{i-1} \succ c_{i+1} \succ \ldots \succ c_k \succ c_{k+1} \succ c_{k+2} \succ \ldots \succ c_{j-1} \succ c_{j+1} \succ \ldots \succ c_{2k}$, and another one that ranks them $c_{2k} \succ c_{2k-1} \succ \ldots \succ c_{j+1} \succ c_{j-1} \succ \ldots \succ c_{k+1} \succ c_k \succ c_{k-1} \succ \ldots \succ c_{i+1} \succ c_{i-1} \succ \ldots \succ c_1 \succ p \succ c_i \succ c_j$ (i.e., the inverse of the former vote, apart from c_i and c_j which have maintained their order);*
- *For each pair i,j without an edge between them in B ($1 \leq i \leq k$, $k+1 \leq j \leq 2k$), one vote that ranks the candidates $c_j \succ c_i \succ p \succ c_1 \succ c_2 \succ \ldots \succ c_{i-1} \succ c_{i+1} \succ \ldots \succ c_k \succ c_{k+1} \succ c_{k+2} \succ \ldots \succ c_{j-1} \succ c_{j+1} \succ \ldots \succ c_{2k}$, and another one that ranks them $c_{2k} \succ c_{2k-1} \succ \ldots \succ c_{j+1} \succ c_{j-1} \succ \ldots \succ c_{k+1} \succ c_k \succ c_{k-1} \succ \ldots \succ c_{i+1} \succ c_{i-1} \succ \ldots \succ c_1 \succ p \succ c_j \succ c_i$ (i.e., the inverse of the former vote, apart from c_j and c_i which have maintained their order).*

We now have to show that this reduction satisfies the preconditions of Theorem 58. We start with the properties that are protocol-independent.

Theorem 59 *R_1 satisfies properties 2b and 2c of Theorem 58 (under any protocol P, because these properties are independent of P), even with a single additional arbitrary vote.*

Proof: In the pairwise election between p and any one of the c_i with $k+1 \leq i \leq 2k$, p is ranked higher in only $4k^2$ votes, and thus loses the pairwise election. So property 2b is satisfied. For a pairwise election between some c_i and c_j ($1 \leq i \leq k$ and $k+1 \leq j \leq 2k$), the first $12k^3$ votes' net contribution to the outcome in this pairwise election is 0. Additionally, the two votes associated with any pair q, r ($1 \leq q \leq k$ and $k+1 \leq r \leq 2k$) also have a net contribution of 0, if either $q \neq i$ or $r \neq j$. The only remaining votes are the two associated with the pair i, j, so c_i wins the pairwise election by 2 votes if there is an edge (i, j) in B, and c_j wins the pairwise election by 2 votes otherwise. So property 2c is satisfied. Because both are satisfied with a "margin" of at least 2, a single additional vote will not change this. ■

Finally, because property 2a is protocol-dependent, we need to prove it for our reduction on a per-protocol basis. This is what the following four theorems achieve.

Theorem 60 *R_1 satisfies property 2a of Theorem 58 under the plurality rule. This holds even when there is a single additional arbitrary vote.*

Proof: If at least one of the c_i with $k+1 \leq i \leq 2k$ is not removed, p can get at most $5k^2$ votes, whereas the lowest-indexed remaining candidate among the c_i with $k+1 \leq i \leq 2k$ will get at least $6k^3$ votes, so p does not win. On the other hand, if all the c_i with $k+1 \leq i \leq 2k$ are removed, p will get at least $6k^3 + 3k^2$ votes, which is more than half the votes, so p wins. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. ■

Theorem 61 *R_1 satisfies property 2a of Theorem 58 under the Borda rule. This holds even when there is a single additional arbitrary vote.*

Proof: If at least one of the c_i with $k+1 \leq i \leq 2k$ is not removed, consider the highest-indexed remaining candidate among the c_i with $k+1 \leq i \leq 2k$; call it h. The first $12k^3$ votes will put h at least $9k^3 - 3k^2$ points ahead of p. ($12k^3 - 3k^2$ of them rank h above p, and the $3k^2$ others can give p an advantage of at most k each.) The $2k^2$ remaining votes can contribute an advantage to p of at most k each, and it follows that h will still have at least $7k^3 - 3k^2$ more points than p. So p does not win. On the other hand, if all the c_i with $k+1 \leq i \leq 2k$ are removed, then there are two groups of $6k^3 - 3k^2$ among the first $12k^3$ votes which (over the remaining candidates) are each other's exact inverses and hence have no net effect on the scores. Also, the last $2k^2$ votes, which are organized in pairs, have no net effect on the score because (over the remaining candidates) the votes in each pair are each other's exact inverse. The remaining votes all rank p highest among the remaining candidates, so p wins. In both cases the "margin" is big enough that a single additional vote will not change this. ■

Theorem 62 *R_1 satisfies property 2a of Theorem 58 under the maximin rule. This holds even when there is a single additional arbitrary vote.*

Proof: If at least one of the c_i with $k+1 \leq i \leq 2k$ is not removed, then in any pairwise election between such a candidate and p, p will get at most $5k^2$ votes. However, the lowest-indexed remaining candidate among the c_i with $k+1 \leq i \leq 2k$ will get at least $6k^3$ votes in every one of its pairwise elections. So p does not win. On the other hand, if all the c_i with $k+1 \leq i \leq 2k$ are removed, p will get at least $6k^3 + 3k^2$ votes in every one of its pairwise elections, which is more than half the votes; so p wins. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. ■

Theorem 63 *R_1 satisfies property 2a of Theorem 58 under the STV rule. This holds even when there is a single additional arbitrary vote.*

Proof: If at least one of the c_i with $k+1 \leq i \leq 2k$ is not removed, consider the lowest-indexed remaining candidate among the c_i with $k+1 \leq i \leq 2k$; call it l. l will hold at least $6k^3$ votes as long as it is not eliminated, and p can hold at most $5k^2$ votes as long as l is not eliminated. It follows that p will be eliminated before l, so p does not win. On the other hand, if all the the c_i with $k+1 \leq i \leq 2k$ are removed, p will hold at least $6k^3 + 3k^2$ votes throughout, which is more than half the votes; so p cannot be eliminated and wins. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. ■

Theorem 64 *In any of $RPRE+plurality$, $RPRE+Borda$, $RPRE+maximin$, and $RPRE+STV$, CONSTRUCTIVE-MANIPULATION is #P-hard.*

Proof: Immediate from the previous theorems. ■

8.2.4 PSPACE-hardness when scheduling and voting are interleaved

In this subsection, we increase the complexity of manipulation one more notch, to PSPACE-hardness,[6] by interleaving the scheduling and vote elicitation processes.

We first discuss the precise method of interleaving required for our result. The method is detailed and quite complicated. Nevertheless, this does *not* mean that the interleaving should always take place in this particular way in order to have the desired hardness. If the interleaving method used for a particular election is (say, randomly) chosen from a wider (and possibly more naturally expressed) class of interleaving methods containing this one, our hardness result still goes through, as hardness carries over from the specific to the general. Thus, our goal is to find the most specific method of interleaving for which the hardness still occurs, because this gives us the most information about more general methods. We only define the method for the case where the number of candidates is a multiple of 4 because this is the case that we will reduce to (so it does not matter how we generalize the protocol to cases where the number of candidates is not a multiple of 4).

Definition 40 *$IPRE$ proceeds as follows:*

1. *Label the matchups (a matchup is a space in the preround in which two candidates can face each other; at this point they do not yet have candidates assigned to them) 1 through $\frac{|C|}{2}$;*

2. *For each matchup i, assign one of the candidates to play in it, and denote this candidate by $c(i,1)$. Thus, one of the candidates in each matchup is known.*

3. *For some k which is a multiple of 4, for each i with $1 \leq i \leq k$, assign the second candidate to play in matchup i, and denote this candidate $c(i,2)$. Thus, we have k fully scheduled matchups.*

[6] PSPACE is the class of problems solvable in polynomial *space*.

4. *For each pair of matchups* $(2i-1, 2i)$ *with* $i > \frac{k}{2}$, *assign two more candidates to face the candidates already in these two matchups, and denote them* $c((2i-1, 2i), 1)$ *and* $c((2i-1, 2i), 2)$. *(Thus, at this point, all that still needs to be scheduled is, for each* i, *which of these two faces* $c(2i-1, 1)$ *and which* $c(2i, 1)$.*)*

5. *For* $i = \frac{k}{2} + 1$ *to* $\frac{|C|}{4}$:

   • *Randomly decide which of* $c((2i-1, 2i), 1)$ *and* $c((2i-1, 2i), 2)$ *faces* $c(2i-1, 1)$, *and which faces* $c(2i, 1)$. *Denote the former* $c(2i-1, 2)$, *the latter* $c(2i, 2)$,

   • *Ask all the voters whether they prefer* $c(i - \frac{k}{2}, 1)$ *or* $c(i - \frac{k}{2}, 2)$. *(We observe that, even if the number of already scheduled matchups is* $k = 0$, *the elicitation process trails behind the scheduling process by a factor* 2.*)*

6. *Elicit the remainder of all the votes.*

One important property of this elicitation process is that the voters are treated symmetrically: when a query is made, it is made to all of the voters in parallel. Thus, no voter gets an unfair advantage with regard to knowledge about the schedule. Another important property is that the elicitation and scheduling process at no point depends on how the voters have answered earlier queries. Thus, voters cannot make inferences about what other voters replied to previous queries on the basis of the current query or the current knowledge about the schedule. These two properties guarantee that many issues of strategic voting that may occur with vote elicitation [Conitzer and Sandholm, 2002c] in fact do not occur here.

We are now ready to present our result.

Theorem 65 *Given a voting protocol* P, *suppose that it is possible, for any Boolean formula* ϕ *in conjunctive normal form (i.e., a SAT instance) over variables* $V = X \cup Y$ *with* $|X| = |Y|$ *(and corresponding literals* L*), to construct in polynomial time a set of votes over a candidate set containing at least* $\{p\} \cup C_L \cup \{c_y^1 : y \in Y\}$ *with the following properties:*

- *(Property 3a) If we remove, for each* $v \in V$, *one of* c_{+v} *and* c_{-v}, p *would win an election under protocol* P *against the remaining candidates if and only if for every clause* $k \in K$ *(where* K *is the set of clauses in* ϕ*), there is some* $l \in L$ *such that* c_l *has not been removed, and* l *occurs in* k. *This should hold even if a single arbitrary vote is added.*

- *(Property 3b) For any* $x \in X$, c_x *and* c_{-x} *are tied in their pairwise election after these votes.*

- *(Property 3c) For any* $y \in Y$, c_y *and* c_{-y} *are both losing their pairwise elections against* c_y^1 *by at least* 2 *votes (so that they will lose them regardless of a single additional vote).*

Then CONSTRUCTIVE-MANIPULATION in $IPRE + P$ *is PSPACE-hard (and PSPACE-complete if* P *can be executed in polynomial space).*

Proof: Consider the following election under $IPRE + P$. Let the candidate set be the set of all candidates occurring in the votes constructed from ϕ (the "original candidates"), plus one dummy candidate for each of the original candidates besides the c_{+v} and c_{-v}. To each of the constructed votes, add all the dummy candidates at the bottom; let the resulting set of votes be the set of the non-manipulators' votes, according to which they will answer the queries posed to them. The manipulator has yet to decide on its strategy for answering queries. After step 4 (according to Definition 40) of $IPRE + P$ (up to which point the manipulator will not have had to make any decisions), let the situation be as follows:

- The number of already fully scheduled matchups is $k = \frac{|C|}{2} - 2|Y|$. In matchup i ($1 \leq i \leq |X|$), c_{+x_i} faces c_{-x_i}. In the remaining fully scheduled matchups, candidates not corresponding to a literal face a dummy candidate.
- Matchups $k+2i-1$ and $k+2i$ ($1 \leq i \leq |Y|$) already have candidates c_{+y_i} and c_{-y_i} in them, respectively. The other two candidates to be assigned to these rounds are $c^1_{y_i}$ and a dummy candidate.

Thus, what will happen from this point on is the following. For i ranging from 1 to $|X|$, first the protocol will schedule which of c_{+y_i} and c_{-y_i} face which of $c^1_{y_i}$ and the dummy candidate. The c_l facing the dummy will move on, and the other will be defeated by $c^1_{y_i}$, by property 3c. Second, everyone will be asked which of c_{+x_i} and c_{-x_i} is preferred, and because the nonmanipulators will leave this pairwise election tied by property 3b, the manipulator's vote will be decisive. Thus, we can think of this as nature and the manipulator alternatingly giving the variables in Y and X respectively truth-values: v is set to *true* if c_{+v} survives, and to *false* if c_{-v} survives. By property 3a it then follows that p wins if and only if the resulting assignment satisfies all the clauses, i.e. is a solution to the SAT instance. Thus, the manipulator's strategy for setting variables should aim to maximize the chance of the SAT instance being satisfied eventually. But this is exactly the problem STOCHASTIC-SAT, which is PSPACE-complete [Papadimitriou, 1985].

If P can be executed in polynomial space, the manipulator can enumerate all possible outcomes for all possible strategies in polynomial space, so the problem is also in PSPACE. ■

Because the preconditions of Theorem 65 are similar to those of Theorem 52, we can build on our previous reductions to apply this theorem to the well-known protocols.

Theorem 66 *For each of $plurality$, $Borda$, $maximin$, and STV, there exists a reduction that satisfies properties 3a, 3b and 3c of Theorem 65. Thus, In any of $IPRE + plurality$, $IPRE + Borda$, $IPRE+maximin$, and $IPRE+STV$, CONSTRUCTIVE-MANIPULATION is PSPACE-complete.*

Proof: We can modify the reductions from Subsection 8.2.2 to satisfy the preconditions of Theorem 65. This is done by adding in the c^1_y in such a way as to achieve property 3c (ranking them just above their corresponding c_y and c_{-y} in slightly more than half the votes), while preserving property 3a (by ranking them as low as possible elsewhere). ■

This concludes the part of this dissertation studying how to tweak voting protocols to make them harder to manipulate. In the next section, we focus on existing, untweaked voting rules: although these rules are easy to manipulate in the sense described in this section (with the exception of STV), it turns out that there are other manipulation problems that are difficult even for these rules, even with few candidates.

8.3 Hardness of manipulating elections with few candidates

We did some of the work in this subsection jointly with Jérôme Lang (IRIT, France).

The hardness results that we proved in the previous section (as well as similar hardness results proven by others [Bartholdi *et al.*, 1989a; Bartholdi and Orlin, 1991; Elkind and Lipmaa, 2005a]) assume that not only the number of voters but also *the number of candidates* is unbounded. Such hardness results lose relevance when the number of candidates is small, because manipulation algorithms that are exponential only in the number of candidates (and only slightly so) might be available. In this section, we first give such an algorithm for an individual agent to manipulate the Single Transferable Vote (STV) rule, which has been shown hard to manipulate in the above sense. The algorithm applies whether or not the voters are weighted.

This motivates the core of this section, which studies the complexity of manipulating elections where the number of candidates is a small constant. Restricting the number of candidates to a constant reduces the number of possible votes for a single voter to a constant. If the voters all have equal weight in the election, the number of *de facto* possible combinations of votes that even a coalition can submit is polynomial in the number of voters in the coalition (since the voters have equal weight, it does not matter which agent in the coalition submitted which vote; only the multiplicities of the votes from the coalition matter). We thus get the following straightforward result.

Proposition 9 *Let there be a constant number of candidates, and suppose that evaluating the result of a particular combination of votes by a coalition is in P. If there is only one voter in the coalition, or if the voters are unweighted, the manipulation problem is in P. (This holds for all the different variants of the manipulation problem, discussed later.)*

Proof: The manipulators (an individual agent or a coalition) can simply enumerate and evaluate all possibilities for their votes (there is a polynomial number of them). Specifically, when there are n voters in the coalition and m candidates, then there are at most $(n+1)^{m!}$ possibilities, because for every one of the $m!$ possible orderings of the candidates there must be between 0 and n voters in the coalition voting according to this ordering (and, because the voters are unweighted, it does not matter which voters they are). This expression is polynomial in n. ■

In particular, in the complete-information manipulation problem in which the votes of the non-colluders are known, evaluating the result of a (coalitional) vote is roughly as easy as determining the winner of an election.[7] This leaves open two avenues for deriving high complexity results with few candidates. First, we may investigate the complete-information coalitional manipulation problem when voters have *different weights*. While many human elections are unweighted, the introduction of weights generalizes the usability of voting schemes, and can be particularly important in multiagent systems settings with very heterogenous agents. As a second avenue, we may ask whether there are reasonable settings where *evaluating* a manipulation is NP-hard. For instance, if

[7] Recall from Chapter 3 that there exist voting rules where determining the winner is computationally hard [Bartholdi *et al.*, 1989b; Hemaspaandra *et al.*, 1997; Cohen *et al.*, 1999; Dwork *et al.*, 2001; Rothe *et al.*, 2003; Davenport and Kalagnanam, 2004; Ailon *et al.*, 2005], including the Slater and Kemeny rules—but this is only so for large numbers of candidates.

we merely have probability distributions on the non-colluders' votes, how does the complexity of determining the probability that a given candidate wins change?

We devote most of this section to studying the first avenue. We study both *constructive* manipulation (making a given candidate win) and *destructive* manipulation (making a given candidate not win). We characterize the exact number of candidates for which manipulation becomes hard for *plurality*, *Borda*, *STV*, *Copeland*, *maximin*, *veto*, *plurality with runoff*, *regular cup*, and *randomized cup* rules. It turns out that the voting rules under study become hard to manipulate at 3 candidates, 4 candidates, 7 candidates, or never. The remainder of this section is devoted to the second avenue, by showing that hardness results from the complete-information coalitional weighted manipulation problem imply similar hardness results in the incomplete-information setting, *even without the assumptions of multiple manipulators and weighted votes*.

8.3.1 Manipulating an election

Due to Proposition 9, we cannot hope to obtain hardness of manipulation in the sense of Section 8.2. Hence, we will introduce more general manipulation problems. To do so, we first discuss the different dimensions of the election manipulation problem:

1. *What information do the manipulators have about the nonmanipulators' votes?* In the *incomplete information* setting, the manipulators are uncertain about the nonmanipulators' votes. This uncertainty could be represented in a number of ways, for example, as a joint probability distribution over the nonmanipulators' votes. In the *complete information* setting, the manipulators know the nonmanipulators' votes exactly. We (initially) focus on the complete information case for the following reasons: 1a. It is a special case of any uncertainty model. Therefore, our hardness results directly imply hardness for the incomplete information setting. 1b. As we will demonstrate later in this section, hardness results for manipulation by coalitions in the complete information setting also imply hardness of manipulation *by individuals* in the incomplete information setting. 2. Results in the complete information setting measure only the *inherent* complexity of manipulation rather than any potential complexity introduced by the model of uncertainty.

2. *Who is manipulating: an individual voter or a coalition of voters?* Both of these are important variants, but we focus on coalitional manipulation for the following reasons: 1. In elections with many voters it is perhaps unlikely that an individual voter can affect the outcome—even with unlimited computational power. 2. For any constant number of candidates (even with an unbounded number of voters), manipulation by individuals in the complete information setting is computationally easy because the manipulator can enumerate and evaluate all its possible votes (rankings of candidates) in polynomial time, as we pointed out in the Introduction.[8] 3. Again, as we will demonstrate later in this section, hardness results for manipulation *by coalitions* in the complete information setting also imply hardness of manipulation *by individuals* in the incomplete information setting.

[8]This assumes that the voting rule is easy to execute—as most rules are (including all the ones under study in this section).

3. *Are the voters weighted or unweighted?* Both of these are important variants, but we focus on weighted voters for the following reasons: 1. In the unweighted case, for any constant number of candidates (even with an unbounded number of voters), manipulation by a coalition in the complete information setting is computationally easy because the coalition can enumerate and evaluate all its effectively different vote vectors, as we pointed out earlier. (We recall that the number of effectively different vote vectors is polynomial due to the interchangeability of the different equiweighted voters, see Proposition 9.) 2. As we will demonstrate later in this section, hardness results for manipulation by *weighted* coalitions in the complete information setting also imply hardness of evaluating the probabilities of different outcomes in the incomplete information setting with *unweighted* (but correlated) voters. 3. In many real-world elections, voters are in fact weighted, for example, by their ownership share in the company, by seniority, or by how many other individuals they represent.

4. *What is the goal of manipulation?* We study two alternative goals: trying to make a given candidate win (we call this *constructive* manipulation), and trying to make a given candidate *not* win (we call this *destructive* manipulation). Besides these goals being elegantly crisp, there are fundamental theoretical reasons to focus on these goals.

   First, hardness results for these goals imply hardness of manipulation under any game-theoretic notion of manipulation, because our manipulation goals are always special cases. (This holds both for deterministic and randomized voting rules.) At one extreme, consider the setting where there is one candidate that would give utility 1 to each of the manipulators, and all other candidates would give utility 0 to each of the manipulators. In this case the only sensible game-theoretic goal for the manipulators is to make the preferred candidate win. This is exactly our notion of constructive manipulation. At the other extreme, consider the setting where there is one candidate that would give utility 0 to each of the manipulators, and all other candidates would give utility 1 to each of the manipulators. In this case the only sensible game-theoretic goal for the manipulators is to make the disliked candidate not win. This is exactly our notion of destructive manipulation.

   Second, at least for deterministic voting rules in the complete information setting, the easiness results transfer from constructive manipulation to any game-theoretic definitions of manipulation that would come down to determining whether the manipulators can make some candidate from a subset of candidates win. For example, one can consider a manipulation successful if it causes some candidate to win that is preferred by each one of the manipulators to the candidate who would win if the manipulators voted truthfully. As another example, one can consider a manipulation successful if it causes some candidate to win that gives a higher sum of utilities to the manipulators than the candidate who would win if the manipulators voted truthfully. (This definition is especially pertinent if the manipulators can use side payments or some other form of restitution to divide the gains among themselves.) Now, we can solve the problem of determining whether some candidate in a given subset can be made to win simply by determining, for each candidate in the subset in turn, whether that candidate can be made to win. So the complexity exceeds that of constructive manipulation by at most a factor equal to the number of candidates (*i.e.*, a constant).

   Third, the complexity of destructive manipulation is directly related to the complexity of determining whether enough votes have been elicited to determine the outcome of the election. Specifically, enough votes have been elicited if there is no way to make the conjectured winner not win by casting the yet unknown votes [Conitzer and Sandholm, 2002c].

In summary, we focus on coalitional weighted manipulation (CW-MANIPULATION), in the complete information setting. We study both constructive and destructive manipulation. Formally:

Definition 41 (CONSTRUCTIVE COALITIONAL WEIGHTED (CW) MANIPULATION) *We are given a set of weighted votes S (the nonmanipulators' votes), the weights for a set of votes T which are still open (the manipulators' votes), and a preferred candidate p. For deterministic rules, we are asked whether there is a way to cast the votes in T so that p wins the election. For randomized rules, we are additionally given a distribution over instantiations of the voting rule, and a number r, where $0 \leq r \leq 1$. We are asked whether there is a way to cast the votes in T so that p wins with probability greater than r.*

Definition 42 (DESTRUCTIVE COALITIONAL WEIGHTED (CW) MANIPULATION) *We are given a set of weighted votes S (the nonmanipulators' votes), the weights for a set of votes T which are still open (the manipulators' votes), and a disliked candidate h. For deterministic rules, we are asked whether there is a way to cast the votes in T so that h does* not *win the election. For randomized rules, we are additionally given a distribution over instantiations of the voting rule, and a number r, where $0 \leq r \leq 1$. We are asked whether there is a way to cast the votes in T so that h wins with probability* less *than r.*

For deterministic rules, we do not consider a manipulation successful if it leaves candidate p or h tied for the win.[9] One way of viewing this is as follows. If ties are broken randomly, then technically, we are dealing with a randomized rule. Then, we can set r so that a tie for the win does not give p enough probability of winning (for example, $r = 2/3$), or so that a tie gives h too much probability of winning (for example, $r = 1/(m+1)$).

Before we start studying these problems, we first complete our justification for requiring hardness even with few candidates. We do so by giving an algorithm for an individual voter to manipulate the STV rule that is exponential only in the number of candidates, and scales to reasonably large numbers of candidates. This is the subject of the next subsection (which can be skipped without affecting the reader's ability to comprehend the rest of this section).

8.3.2 Algorithm for individually manipulating the STV rule

When the number of candidates is unbounded, the STV rule is known to be NP-complete to constructively manipulate, even by a single manipulator when the votes are not weighted [Bartholdi and Orlin, 1991]. In this subsection we present an algorithm for manipulating STV as a single voter, when the votes of the others are known. Votes may be weighted.

To study the complexity fundamental to manipulating STV, rather than complexities introduced by tie-breaking rules, for this algorithm we make the following assumption. We assume that the STV rule uses some deterministic method for breaking ties (when choosing the loser to be eliminated at

[9]Our proofs do not depend on this specification, with the exception of Theorem 74.

the end of a round), where the tie-breaking does not depend on aspects of the votes that the STV rule has not considered so far (such as who has been ranked *lowest* by the largest number of voters). However, the tie-breaking can depend on the number of the round, candidates still in the running, *etc*. In the algorithm, the tie-breaking rule is used in the $\arg\min$ function.

The algorithm simulates the various ways in which the elimination of candidates may proceed given various votes by the manipulator. It follows the principle of least commitment in deciding which manipulative votes to consider. It returns the set of candidates that win for some vote by the manipulator. (It is easy to extend the algorithm so that it also provides a vote to effect such a victory.) Again, let there be n voters, where we index the manipulator n. Let C be the set of remaining candidates. Let v_i be the vote of voter i ($1 \leq i < n$) and let w_i be the weight of voter i ($1 \leq i \leq n$). Let s_j be the weight of the voters that rank candidate j first among the remaining candidates.

Some stages of the simulation can be reached only if the manipulator has a certain candidate f ranked first among the remaining candidates. At such stages, we will know precisely how the elimination will proceed, until f is eliminated and the manipulator's vote is freed up again. We say that $f = 0$ when there is no constraint on how the manipulator ranks the remaining candidates in the current stage of the simulation.

The function TRANSFERVOTES takes as input a candidate c, the remaining set of candidates C, the vector $(s_1, \ldots, s_m)$, and the votes and weights. It returns what would be the new vector $(s_1, \ldots, s_m)$ if c were eliminated in this round.

Now, we are ready to present the manipulation algorithm, which, when called with the original set of candidates C and $f = 0$, returns the set of all candidates that will win for some vote by the manipulator.

```
MANIPULATE(C, (s_1, ..., s_m), (v_1, ..., v_{n-1}), (w_1, ..., w_n), f)
if C = {c}
// we have eliminated all but a single candidate
    return {c}
else if (f ≠ 0)
// the manipulator's vote is already committed to a candidate at this stage
    c ← arg min_{j∈C}(s_j)
    (t_1, ..., t_m) ← TRANSFERVOTES(c, C, (s_1, ..., s_m), (v_1, ..., v_{n-1}),
        (w_1, ..., w_n))
    if c = f
    // the manipulator's vote is freed up
        return MANIPULATE(C − {c}, (t_1, ..., t_m), (v_1, ..., v_{n-1}),
            (w_1, ..., w_n), 0)
    else
    // the manipulator's vote remains committed
        return MANIPULATE(C − {c}, (t_1, ..., t_m), (v_1, ..., v_{n-1}),
            (w_1, ..., w_n), f)
else
// the manipulator's vote is not committed at this stage
    c_1 ← arg min_{j∈C}(s_j)
    // which candidate is losing before the manipulator assigns his vote?
    s_{c_1} ← s_{c_1} + w_n
    c_2 ← arg min_{j∈C}(s_j)
    // which candidate loses if the manipulator supports c_1?
    (t_1, ..., t_m) ← TRANSFERVOTES(c_1, C, (s_1, ..., s_m), (v_1, ..., v_{n-1}),
        (w_1, ..., w_n))
    if c_1 = c_2
    // the manipulator cannot rescue c_1 at this stage
        return MANIPULATE(C − {c_1}, (t_1, ..., t_m), (v_1, ..., v_{n-1}),
            (w_1, ..., w_n), 0)
    else
    // the manipulator can choose to rescue c_1 at this stage
        S_1 ← MANIPULATE(C − {c_1}, (t_1, ..., t_m), (v_1, ..., v_{n-1}),
            (w_1, ..., w_n), 0)
```

// case I: do not rescue c_1
$(t_1, \ldots, t_m) \leftarrow$ TRANSFERVOTES$(c_2, C, (s_1, \ldots, s_m), (v_1, \ldots, v_{n-1}),$
$(w_1, \ldots, w_n))$
$S_2 \leftarrow$ MANIPULATE$(C - \{c_2\}, (t_1, \ldots, t_m), (v_1, \ldots, v_{n-1}),$
$(w_1, \ldots, w_n), c_1)$
// case II: do rescue c_1
return $S_1 \cup S_2$

We now analyze how many recursive calls we will make to this algorithm. Let $T(k)$ be the maximal number of calls we need for a set of remaining candidates of size k. Let $T_0(k)$ be the maximal number of calls we need when the first call has $f \neq 0$. Then we have the following recurrences:

$$T(k) \leq 1 + T(k-1) + T_0(k-1) \tag{8.1}$$
$$T_0(k) \leq 1 + T(k-1) \tag{8.2}$$

Combining the two we get

$$T(k) \leq 2 + T(k-1) + T(k-2) \tag{8.3}$$

The asymptotic bound that we derive from this recurrence is indeed tight for the running time of MANIPULATE, as the following example shows. In the example, candidates are eliminated sequentially, but the manipulator can postpone the elimination of any candidate for exactly one round. Let the candidates be $(c_1, \ldots, c_m)$. For candidate i, let there be exactly i voters that rank it first, for a total of $\sum\limits_{i=1}^{m} i = \frac{(m+1)m}{2}$ voters other than the manipulator, of weight 1 each. All the voters other than the manipulator that do not rank candidate m first, rank it second. The manipulator has weight $1 + \epsilon$.[10] We claim that the arguments passed to MANIPULATE always satisfy one of the following two properties:

- (1) $C = \{c_i, c_{i+1}, \ldots, c_m\}$ and $f = 0$.
- (2) $C = \{c_i, c_{i+2}, c_{i+3}, \ldots, c_m\}$ and $f = c_i$.

The initial call is of type (1). If the current call is of type (1), this will lead to two recursive calls: one of type (1) (the manipulator does not rescue candidate c_i), and one of type (2) (the manipulator does rescue c_i). (The exception is when $i \geq m - 1$ but this is irrelevant to the asymptotic analysis.) This makes the recurrence in Equation 8.1 tight. If the current call is of type (2), this will lead to one recursive call of type (1) because c_i gets eliminated in spite of the fact that the manipulator is ranking it first. This makes the recurrence in Equation 8.2 tight. Because the recurrences in Equations 8.1 and 8.2 are tight, the recurrence in Equation 8.3 is tight.

[10] Alternatively, we can assume unweighted votes and a tie-breaking mechanism that always breaks ties towards lower-indexed candidates, that is, it breaks a tie between c_i and c_{i+1} in favor of c_i.

The solution to the recurrence is $O((\frac{1+\sqrt{5}}{2})^k)$. Observing that one call to MANIPULATE (not counting the recursive calls to MANIPULATE, but counting the calls to TRANSFERVOTES) can be done in $O(n)$ time (assuming that $(s_1, \ldots, s_m)$ can be stored in an array), we have the following result.

Theorem 67 *The algorithm MANIPULATE runs in time $O(n(\frac{1+\sqrt{5}}{2})^m)$, where m is the number of candidates and n is the number of voters.*

So, MANIPULATE runs in $O(n \cdot 1.62^m)$ time. While this function is exponential in m (which is to be expected given that the problem is NP-complete), it is nevertheless not exceedingly large for realistic numbers of candidates. For instance, with 10 candidates, $(\frac{1+\sqrt{5}}{2})^{10} < 123$. Furthermore, on most instances, the algorithm is likely to terminate much faster than this, since we make two recursive calls only when the manipulator can rescue a candidate from elimination in a given round. In large elections where the manipulator's weight is relatively insignificant, it is unlikely that this would happen even more than once.

8.3.3 Complexity of weighted coalitional manipulation with few candidates

We are now ready to present our results on the hardness of coalitional manipulation when there are few candidates and the votes are weighted. We will study not only *whether* any given rule is hard to manipulate with a constant number of candidates, but also *how many* candidates are needed for the hardness to occur. This number is important for evaluating the relative manipulability of different voting rules (the lower this number, the less manipulable the rule). For each rule that we show is hard to manipulate with some constant number of candidates, we show this for the smallest number of candidates for which the hardness occurs, and we show that manipulation becomes easy if we reduce the number of candidates by one. (Once we have identified this transition point, it is easy to see that the manipulation problem remains hard for any greater number of candidates, and remains easy for any smaller number of candidates, for example by adding "dummy" candidates.)

Constructive Manipulation

We first present our results for constructive manipulation.

We begin by laying out some cases where constructive manipulation can be done in polynomial time. We start with some rules that are easy to manipulate constructively regardless of the number of candidates. For the plurality rule, showing this is straightforward:

Theorem 68 *For the* plurality *rule,* CONSTRUCTIVE CW-MANIPULATION *can be solved in polynomial time (for any number of candidates).*

Proof: The manipulators can simply check if p will win if all the manipulators vote for p. If not, they cannot make p win. ■

For the cup rule, the proof is a little more involved:

Theorem 69 *For the* cup *rule (given the assignment of candidates to leaves),* CONSTRUCTIVE CW-MANIPULATION *can be solved in polynomial time (for any number of candidates).*

Proof: We demonstrate a method for finding all the potential winners of the election. In the binary tree representing the schedule, we can consider each node to be a subelection, and compute the set of potential winners for each subelection. (In such a subelection, we may say that the voters only order the candidates in that subelection since the place of the other candidates in the order is irrelevant.) Say a candidate *can* obtain a particular result in the election if it does so for some coalitional vote. The key claim to the proof, then, is the following: a candidate can win a subelection if and only if it can win one of its children, *and* it can defeat one of the potential winners of the sibling child in a pairwise election. It is easy to see that the condition is necessary. To show that it is sufficient, let p be a candidate satisfying the condition by being able to defeat h, a potential winner of the other child (or *half*). Consider a coalitional vote that makes p win its half, and another one that makes h win its half. We now let each coalitional voter vote as follows: it ranks all the candidates in p's half above all those in h's half; the rest of the order is the same as in the votes that make p and h win their halves. Clearly, this will make p and h the finalists. Also, p will win the pairwise election against h since it is always ranked above h by the colluders; and as we know that there is some coalitional vote that makes p defeat h pairwise, this one must have the same result. The obvious recursive algorithm has running time $O(m^3 n)$ according to the Master Theorem [Cormen *et al.*, 1990]. ■

The remaining easiness results that we show in this subsubsection only show easiness up to a certain number of candidates. For each of these results, we later show that adding one more candidate causes the problem to become NP-complete.

As a first observation, when there are only two candidates, all the rules are equivalent to the plurality rule, and hence both types of manipulation (constructive and destructive) are in P for all of the rules. However, some rules are still easy to manipulate constructively with more than 2 candidates. In each of the following cases, we prove easiness of manipulation by demonstrating that if there exists a successful manipulation, there also exists one *where all the manipulators vote the same way*. All such ways of voting can be easily enumerated: because the number of candidates is constant, the number of different orderings of the candidates is constant. Also, each way of voting is easy to evaluate in these rules.

Theorem 70 *If the* Copeland *rule with 3 candidates has a* CONSTRUCTIVE CW-MANIPULATION, *then it has a* CONSTRUCTIVE CW-MANIPULATION *where all of the manipulators vote identically. Therefore,* CONSTRUCTIVE CW-MANIPULATION *is in P.*

Proof: Let the 3 candidates be p, a, and b. We are given the nonmanipulators' votes S, and the weights for the manipulators' votes T. Let the total vote weight in T be K.

For a set of weighted votes V and two candidates x, y, we denote by $N_V(x, y)$ the cumulated weights of the votes in V ranking x prior to y, and we let $D_V(x, y) = N_V(x, y) - N_V(y, x)$. Let us consider the following four cases which cover all possible situations:
Case 1: $K > D_S(a, p)$ and $K > D_S(b, p)$.

In this case, *any* configuration of votes for T such that p is ranked first for all votes makes p win the election.
Case 2: $K > D_S(a, p)$ and $K = D_S(b, p)$.

It can easily be shown that it is harmless to assume that all votes in T rank p first. Therefore, what remains to be done in order to have p win is to find who in T should vote (p, a, b) and who should vote (p, b, a). What we know so far (before knowing how the votes in T will split between these two profiles) is: (1) $D_{S \cup T}(p, a) = K - D_S(p, a) > 0$ and (2) $D_{S \cup T}(p, b) = K - D_S(p, b) = 0$. (1) makes p get +1 and a get -1 while (2) makes both p and b get 0. Therefore, the partial Copeland scores (not taking account of the a vs. b pairwise election), are $+1$ for p (which will not change after taking account of the $a - b$ pairwise election), -1 for a and 0 for b; hence, the only way for p to win (with certainty) is to avoid b getting a point in the pairwise election against a, i.e., to ensure that $D_{S \cup T}(a, b) \geq 0$. It can easily be shown that this is possible if and only if $K \geq D_S(b, a)$. Therefore, we have found that there exists a successful manipulation for p iff $K \geq D_S(b, a)$, and in this case a successful manipulation is the one where all voters in the coalition vote (p, a, b).
Case 3: $K = D_S(a, p)$ and $K > D_S(b, p)$.

This is similar to Case 2, switching the roles of a and b; the condition then is $K \geq D_S(a, b)$ and the successful manipulation is the one where all vote (p, b, a).
Case 4: $K < D_S(a, p)$ or $K < D_S(b, p)$ or ($K \leq D_S(a, p)$ and $K \leq D_S(b, p)$).

Here, whatever the votes in T, the Copeland score of p is smaller than or equal to 0 and therefore p cannot be guaranteed to win, so there is no successful manipulation. Thus, in every case, either there is no successful manipulation, or there is a successful manipulation where all manipulators vote identically. ■

Theorem 71 *If the* maximin *rule with 3 candidates has a* CONSTRUCTIVE CW-MANIPULATION*, then it has a* CONSTRUCTIVE CW-MANIPULATION *where all of the manipulators vote identically. Therefore,* CONSTRUCTIVE CW-MANIPULATION *is in P.*

Proof: Let the 3 candidates be p, a, and b. We are given the nonmanipulators' votes S, and the weights for the manipulators' votes T. Let the total vote weight in T be K. Again, it is easy to show that all the manipulators can rank p first without harm.

Let us denote by $P_{K1,K2}$ a vote configuration for T such that a subset T_1 of T, whose cumulated weight is K_1, votes (p, a, b) and $T_2 = T \setminus T_1$, whose cumulated weight is K_2 (with $K_1 + K_2 = K$), votes (p, b, a). Now all that remains to show is the following: if p wins with the votes in T being $P_{K1,K2}$ then either p wins with the votes in T being $P_{K,0}$ or p wins with the votes in T being $P_{0,K}$. Let us consider these two cases for the outcome of the whole election (including the votes in T):
Case 1: the uniquely worst pairwise election for a is against b, and the uniquely worst pairwise election for b is against a. One of a and b must have got at least half the vote weight in the pairwise election against the other (say, without loss of generality, a) and therefore have a maximin score of at least half the vote weight. Since a did even better against p, p received less than half the vote weight in their pairwise election and therefore p does not win.
Case 2: One of a and b (say, without loss of generality, a) does at least as badly against p as against the other (so, a's worst opponent is p). Then all the voters in the coalition might as well vote (p, a, b), because this will change neither a's score nor p's score, and might decrease (but not increase) b's score. ■

Theorem 72 *If the* randomized cup *rule with 6 candidates has a* CONSTRUCTIVE CW-MANIPULATION, *then it has a* CONSTRUCTIVE CW-MANIPULATION *where all of the manipulators vote identically. (This holds regardless of which balanced tree is chosen.) Therefore,* CONSTRUCTIVE CW-MANIPULATION *is in P.*

Proof: We show how to transform a successful manipulation into a successful manipulation where all the manipulators cast the same vote, in a number of steps. Observe that swapping two candidates that are ranked directly after one another in a vote can only affect the outcome of their pairwise election, and not those of the others. (Throughout the proof, when we talk about swapping two candidates in a vote, this only means swapping two candidates *ranked directly behind each other*.)

If p is ranked directly behind another candidate c, the only effect that swapping them can have is to make p the winner of their pairwise election where it was not before; clearly this can never hurt p's chances of winning. Repeated application of this gives us a successful manipulation *where all the manipulators rank* p *at the top*.

We now divide the other candidates into two sets: B is the set of candidates that defeat p in their pairwise election, G is that of candidates that are defeated by p. We now claim that when a candidate $g \in G$ is ranked directly behind a candidate $b \in B$, swapping them cannot hurt p's chances of winning. This is because of the following reason. Again, the only effect that the swap can have is to make g the winner of its pairwise election with b, where it was not before. We show that for any schedule (assignment of candidates to leaves), if p wins when b defeats g, then p also wins in the case where this pairwise election is changed to make g defeat b (but nothing else is changed). For any schedule where g and b do not meet this is obvious. If b and g face each other in the final, p of course does not win. If b and g face each other in a semifinal, and b defeats g, then again p cannot win because it cannot defeat b in the final. So the only case left to check is a schedule where b and g meet in a quarterfinal (because the tree is balanced). If p wins with this schedule when b defeats g, that means that b is defeated by some other $g' \in G$ in the semifinal (if b wins the semifinal, p could never defeat it in the final; if b loses to some $b' \in B$, p could never defeat b' in the final; and p itself cannot defeat b in the semifinal either), and that p defeats this g' in the final. Then if we change the winner of the quarterfinal to g, regardless of whether g or g' wins the semifinal, p will win the final. Thus the claim is proven. Repeated application of this gives a successful manipulation *where all the manipulators rank* p *at the top, and all candidates in* G *above all candidates in* B.

All that remains is to get the manipulators to agree on the order of the candidates within G and the order of the candidates within B. We will show how to do this for G; the case of B is entirely analogous. If there are 0 or 1 candidates in G, there is only one order for these candidates. If there are 2 candidates in G, then swapping them whenever the winner of their pairwise election is ranked below the loser does not affect the outcome of their pairwise election and hence nothing at all.

If there are 3 candidates in G, that means there are only 2 in B. Say $B = \{b_1, b_2\}$. Divide the candidates in G into $G_B, G_{\{b_1\}}, G_{\{b_2\}}, G_{\{\}}$, where a candidate in G_S defeats all candidates in S but none in $B - S$. We first claim that it never hurts to swap when a candidate $g_B \in G_B$ is ranked directly below another candidate $g \in G$. Again we do so by showing that with any schedule where p wins when g defeats g_B, p also wins when g_B defeats g (but everything else is unchanged). If g and g_B never meet this is obvious; if g and g_B meet in a semifinal or the final, it does not matter to p which one wins. If g defeats g_B in a quarterfinal and p wins the election, then one of the three following cases applies: 1. p defeats g in the semifinal, 2. g defeats some $b \in B$ in the semifinal and is defeated by p in the final, 3. g faces the third candidate $g_3 \in G$ in the semifinal, and p wins the final agaist the winner of this. Now, if g_B instead had defeated g, then in case 1 p defeats g_B in the semifinal and goes on to win the final as before; in case 2, g_B defeats b as well in the semifinal and then loses to p in the final; and in case 3, p faces either g_B or g_3 in the final and wins. Repeated application of this gets all the elements in G_B ranked above all the other elements in G in all the manipulators' votes, and this rule also allows us to get the elements in G_B ordered among themselves in the same way in all the manipulators' votes. Similarly, we can show that we can always swap the elements in $G_{\{\}}$ downwards, so that all the elements of this set are ranked below all other elements in G, and ordered among themselves in a unique manner across all the votes.

Now, if indeed there is some element in G_B or $G_{\{\}}$, then there are at most two candidates left in G_B whose order might differ across manipulators' votes. As in the case of 2 candidates, we can simply always swap the winner of their pairwise election up without changing anything, and we are done. On the other hand suppose there is no element in G_B or $G_{\{\}}$, so that all the remaining elements are in $G_{\{b_1\}}$ or $G_{\{b_2\}}$. We claim that either it does not hurt to swap all the candidates from $G_{\{b_1\}}$ below all those of $G_{\{b_2\}}$, or vice versa. In the case where either $G_{\{b_1\}}$ or $G_{\{b_2\}}$ is empty, this claim is vacuous: so suppose without loss of generality that $G_{\{b_1\}}$ has two elements g_1 and g_2, and $G_{\{b_2\}}$ one element, g_3. Now suppose the contrary, that is, that always swapping g_3 above g_1 and g_2 decreases p's chances of winning the election, but that always swapping g_3 below g_1 and g_2 also decreases p's chances of winning the election. It follows that always swapping g_3 above g_1 and g_2 causes g_3 to win both pairwise elections, and that always swapping g_3 below g_1 and g_2 causes g_3 to lose both pairwise elections. (For otherwise, the manipulators cannot change the winner of one of these pairwise elections, and because the other pairwise election must have a winner in the current state, either always swapping g_3 up or always swapping g_3 down will not affect any pairwise election at all, and hence the probability of p winning would remain unchanged.) Hence in the current state g_3 must be winning exactly one of these pairwise elections (without loss of generality, the one against g_1). Now, because always swapping g_3 below g_1 and g_2 only has the effect of making it lose against g_1 as well, and because this reduces the probability of p winning, it follows that the number of schedules where p wins now is strictly greater than the number of schedules where p wins if g_3 lost to g_1 in its pairwise election but everything else remained the same. Thus, the number of schedules where p wins now but would not win if g_3 lost to g_1 in their pairwise election (call this set of schedules $L(g_1)$), is strictly greater than the number of schedules where p does not win now but would win if g_3 lost to g_1 in their pairwise election (call this set of schedules $W(g_1)$). Similarly we can show that the number of schedules where p wins now but would not win if g_3 defeated g_2 in their pairwise election (call this set of schedules $W(g_2)$), is strictly greater than the number of schedules where p does not win now but would win if g_3 defeated g_2 in their pairwise election (call this set of schedules $L(g_2)$). So, $|W(g_1)| < |L(g_1)|$, and $|W(g_2)| > |L(g_2)|$. We now derive the desired contradiction by showing $W(g_1)| \geq |W(g_2)|$ and $|L(g_1)| \leq |L(g_2)|$. Consider the mapping f from schedules to schedules that simply swaps the position of g_1 and g_2 in the schedule. This mapping is one-to-one. We now claim that if $\sigma \in W(g_2)$, then $f(\sigma) \in W(g_1)$, thereby demonstrating $|W(g_1)| \geq |W(g_2)|$; the case for $|L(g_1)| \leq |L(g_2)|$ is similar. $\sigma \in W(g_2)$ means that p wins in σ now but would not win if g_3 defeated g_2 in their pairwise election. It is straightforward to check that this only happens if g_2 and g_3 meet in a quarterfinal, and the winner goes on to meet b_1 in the semifinal (whom g_2 would defeat but g_3 would not), while p goes on to the final in the other half of the cup. Now we must show $f(\sigma) \in W(g_1)$. In $f(\sigma)$, g_1 and g_3 face each other in a quarterfinal. The semifinal after this quarterfinal will certainly have b_1 in it (if b_1 plays a quarterfinal before this, b_1 must have the same opponent in this quarterfinal as in σ, because b_1 cannot face g_1 in the quarterfinal in σ and still move on to the semifinal in σ; hence b_1 will defeat its quarterfinal opponent in $f(\sigma)$ as well). Also, the final will certainly have p in it (even if g_1 was in p's half, since the sets of candidates within p's half that g_1 and g_2 defeat are identical, this half will proceed exactly as in σ). Now, in the current state of the votes, g_3 will defeat g_1 in the quarterfinal, upon which b_1 will defeat g_3 in the semifinal and p in the final. On the other hand, if g_1 defeated g_3, it would defeat b_1 in the semifinal and p would win the final against g_1, and hence win the entire election. It follows that $f(\sigma) \in W(g_1)$, as was to be shown. So either it does not hurt to swap all the candidates from $G_{\{b_1\}}$ below all those of $G_{\{b_2\}}$, or vice versa. It remains to be shown that the manipulators' rankings of the candidates within $G_{\{b_1\}}$ and within $G_{\{b_2\}}$ can be made to coincide. Given (without loss of generality) $g_1, g_2 \in G_{\{b_1\}}$ it is straightforward to check that it does not matter to p which of them would win if they faced each other in the final or semifinal. In case they face each other in a quarterfinal, then if p is in the same half of the cup it does not matter which of g_1 and g_2 wins the quarterfinal, because p (if it reaches the semifinal) would defeat either one. If p is in the other half, the only thing that matters is whether this half's finalist is in B or in G. Thus, if the winner of the quarterfinal between g_1 and g_2 faces the last remaining candidate from G the finalist will certainly be in G, so who won the quarterfinal does not matter; on the other hand, if the winner of the quarterfinal between g_1 and g_2 faces a candidate in B, then again it does not matter who wins the quarterfinal, because g_1 and g_2 defeat the same set of candidates from B. It follows that it is irrelevant to p's chances of winning what the outcome of a pairwise election between candidates in $G_{\{b_1\}}$ or between candidates in $G_{\{b_2\}}$ is, so we can order them among themselves in whichever way we like. Hence we can make the manipulators' votes coincide on the order of the 3 candidates in G.

If there are 4 candidates in G, that means there is only one in B. Hence we can partition G into G_B and $G_{\{\}}$. With techniques similar to the case of 3 candidates in G, we can show that we can always swap candidates in G_B above those in $G_{\{\}}$; and we can show that a vote's order of the candidates within G_B (or $G_{\{\}}$) is irrelevant. Hence we can make the manipulators' votes coincide on the order of the 4 candidates in G.

Finally, if there are 5 candidates in G, then p defeats all other candidates in pairwise elections, so p is guaranteed to win regardless of how the candidates in G are ranked. ■

We are now ready to prove hardness results that match the bounds given by the easiness results above. (That is, for every rule which we showed is easy to manipulate with up to l candidates, we now show that it is hard to manipulate with $l+1$ candidates.) In many of the proofs of NP-hardness, we use a reduction from the PARTITION problem, which is NP-complete [Karp, 1972]:

Definition 43 PARTITION. *We are given a set of integers $\{k_i\}_{1 \leq i \leq t}$ (possibly with multiplicities) summing to $2K$, and are asked whether a subset of these integers sums to K.*

Theorem 73 *For any scoring rule other than the* plurality *rule,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 3 candidates.*

Proof: First, note that when there are only 3 candidates, a positional scoring rule is defined by a vector of integers $\vec{\alpha} = \langle \alpha_1, \alpha_2, \alpha_3 \rangle$ such that $\alpha_1 \geq \alpha_2 \geq \alpha_3$. Without loss of generality, we can assume that $\alpha_3 = 0$ (since translating the α_i's by a constant has no effect on the outcome of the election). We can also assume that $\alpha_2 \geq 2$. (If $\alpha_2 = 0$ then the rule is either meaningless (if $\alpha_1 = 0$) or equivalent to the plurality rule, so we can assume $\alpha_2 > 0$. Then, we can scale the α_i appropriately, which will not affect the rule.) Showing the problem is in NP is easy. To show NP-hardness, we reduce an arbitrary PARTITION instance to the following CONSTRUCTIVE MANIPULATION instance. The 3 candidates are a, b, and p. In S there are $(2\alpha_1 - \alpha_2)K - 1$ voters voting (a, b, p) and $(2\alpha_1 - \alpha_2)K - 1$ voters voting (b, a, p). In T, for each k_i there is a vote of weight $(\alpha_1 + \alpha_2)k_i$. Suppose there is a partition of the k_i. Then, let the votes in T corresponding to the one half of the partition be (p, a, b) and the votes corresponding to the other half be (p, b, a). Then the score of p is $2(\alpha_1 + \alpha_2)\alpha_1 K$ while the scores of both a and b are $(\alpha_1 + \alpha_2)(2\alpha_1 K - 1)$, therefore p is the winner and there is a manipulation. Conversely, suppose there exists a manipulation. Then, since scoring procedures satisfy monotonicity, we can assume without loss of generality that the voters in the coalition T rank p first. Let x (resp. y) be the total weight of voters in T of the voters who vote (p, a, b) (resp. (p, b, a)). Note that we have $x + y = 2K$. Then the score of p is $2(\alpha_1 + \alpha_2)\alpha_1 K$, the score of a is $(\alpha_1 + \alpha_2)((2\alpha_1 - \alpha_2)K - 1 + x\alpha_2)$, and the score of b is $(\alpha_1 + \alpha_2)((2\alpha_1 - \alpha_2)K - 1 + y\alpha_2)$. Since p is the winner, its score must be at least that of a, which is equivalent to $x\alpha_2 \leq K\alpha_2 + 1$. Because $\alpha_2 > 0$, the latter condition is equivalent to $x \leq K + \frac{1}{\alpha_2}$, which is equivalent to $x \leq K$ (because $\alpha_2 \geq 2$). Similarly, we get $y \leq K$, which together with $x + y = 2K$ enables us to conclude that $x = y = K$, so there exists a partition. ∎

Corollary 4 *For the* veto *and* Borda *rules,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 3 candidates.*

(On a historical note, we actually proved Corollary 4 before we discovered its generalization to Theorem 73; the generalization was independently discovered by us, Hemaspaandra and Hemaspaandra [2005], and Procaccia and Rosenschein [2006].)

Theorem 74 *For the* Copeland *rule,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 4 candidates.*

Proof: Showing the problem is in NP is easy. To show it is NP-hard, we reduce an arbitrary PARTITION instance to the following CONSTRUCTIVE CW-MANIPULATION instance. There are 4 candidates, a, b, c and p. In S there are $2K + 2$ voters voting (p, a, b, c), $2K + 2$ voting (c, p, b, a), $K + 1$ voting (a, b, c, p), and $K + 1$ voting (b, a, c, p). In T, for every k_i there is a vote of weight k_i. We show the instances are equivalent. First, every pairwise election is already determined without T, except for the one between a and b. p defeats a and b; a and b each defeat c; c defeats p. If there is a winner in the pairwise election between a and b, that winner will tie with p. So p wins the Copeland election if and only if a and b tie in their pairwise election. But, after the votes in S alone, a and b are tied. Thus, the votes in T maintain this tie if and only if the combined weight of the votes in T preferring a to b is the same as the combined weight of the votes in T preferring b to a. This can happen if and only if there is a partition. ∎

Theorem 75 *For the* maximin *rule,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 4 candidates.*

Proof: Showing the problem is in NP is easy. To show it is NP-hard, we reduce an arbitrary PARTITION instance to the following CONSTRUCTIVE CW-MANIPULATION instance. There are 4 candidates, a, b, c and p. In S there are $7K-1$ voters voting (a,b,c,p), $7K-1$ voting (b,c,a,p), $4K-1$ voting (c,a,b,p), and $5K$ voting (p,c,a,b). In T, for every k_i there is a vote of weight $2k_i$. We show the instances are equivalent. Suppose there is a partition. Then, let the votes in T corresponding to the k_i in one half of the partition vote (p,a,b,c), and let the other ones vote (p,b,c,a). Then, p does equally well in each pairwise election: it always gets $9K$ pairwise points. a's worst pairwise election is against c, getting $9K-1$. b's worst is against a, getting $9K-1$. Finally c's worst is against b, getting $9K-1$. Hence, p wins the election. So there is a manipulation. Conversely, suppose there is a manipulation. Then, since moving p to the top of each vote in T will never hurt p in this rule, there must exist a manipulation in which all the votes in T put p at the top, and p thus gets $9K$ as its worst pairwise score. Also, the votes in T cannot change which each other candidate's worst pairwise election is: a's worst is against c, b's worst is against a, and c's worst is against b. Since c already has $9K-1$ points in its pairwise election against b, no vote in T can put c ahead of b. Additionally, if any vote in T puts a right above c, swapping their positions has no effect other than to decrease a's final score, so we may also assume this does not occur. Similarly we can show it safe to also assume no vote in T puts b right above a. Combining all of this, we may assume that all the votes in T vote either (p,a,b,c) or (p,b,c,a). Since a already has $7K-1$ points in the pairwise election against c, the votes in T of the first kind can have a total weight of at most $2K$; hence the corresponding k_i can sum to at most K. The same holds for the k_i corresponding to the second kind of vote on the basis of b's score. Hence, in both cases, they must sum to exactly K. But then, this is a partition. ■

Theorem 76 *For the* STV *rule,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 3 candidates.*

Proof: Showing the problem is in NP is easy. To show it is NP-hard, we reduce an arbitrary PARTITION instance to the following CONSTRUCTIVE CW-MANIPULATION instance. There are 3 candidates, a, b and p. In S there are $6K-1$ voters voting (b,p,a), $4K$ voting (a,b,p), and $4K$ voting (p,a,b). In T, for every k_i there is a vote of weight $2k_i$. We show the instances are equivalent. Suppose there is a partition. Then, let the votes in T corresponding to the k_i in one half of the partition vote (a,p,b), and let the other ones vote (p,a,b). Then in the first round, b has $6K-1$ points, a has $6K$, and p has $6K$. So b drops out; all its votes transfer to p, so that p wins the final round. So there is a manipulation. Conversely, suppose there is a manipulation. Clearly, p cannot drop out in the first round; but also, a cannot drop out in the first round, since all its votes in S would transfer to b, and b would have at least $10K-1$ points in the final round, enough to guarantee it victory. So, b must drop out in the first round. Hence, from the votes in T, both a and c must get at least $2K$ weight that puts them in the top spot. The corresponding k_i in either case must thus sum to at least K. Hence, in both cases, they must sum to exactly K. But then, this is a partition. ■

This also allows us to show that constructively manipulating the plurality with runoff rule is hard:

Theorem 77 *For the* plurality with runoff *rule,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 3 candidates.*

Proof: Showing the problem is in NP is easy. To show it is NP-hard, we observe that with 3 candidates, the plurality with runoff rule coincides with the STV rule, and CONSTRUCTIVE MANIPULATION for STV with 3 candidates is NP-hard. ■

Theorem 78 *For the* randomized cup *rule,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 7 candidates.*

Proof: The problem is in NP because for any vector of votes, we can check for every one of the possible schedules for the cup whether p wins (because the number of candidates is constant, so is the number of possible schedules). To show it is NP-hard, we reduce an arbitrary PARTITION instance to the following CONSTRUCTIVE CW-MANIPULATION instance. There are 7 candidates, a, b, c, d, e, f, and p. In T, for every k_i there is a vote of weight $2k_i$. Let $W = 4K$. Let the votes in S be as follows: there are $\frac{5}{2}W$ votes (d, e, c, p, f, a, b), $\frac{1}{2}W$ votes (d, e, c, p, f, b, a), $\frac{5}{2}W$ votes (f, d, b, p, e, c, a), $\frac{1}{2}W$ votes (f, d, b, p, e, a, c), $\frac{5}{2}W$ votes (e, f, a, p, d, b, c), $\frac{1}{2}W$ votes (e, f, a, p, d, c, b), $2W$ votes (p, a, c, b, d, e, f), W votes (p, c, b, a, f, d, e), W votes (c, b, a, f, d, e, p), $2W$ votes (b, a, c, e, f, d, p), and $4K-1$ votes that we will specify shortly. Since it takes only $8\frac{1}{2}W$ votes to win a pairwise election, it follows that the outcomes of the following pairwise elections are already determined: p defeats each of a, b, c (it has $9W$ votes in each of these pairwise elections from the given votes); each of d, e, f defeats p ($9W$ in each case); d defeats e, e defeats f, f defeats d ($10W$ in each case); a defeats d, b defeats e, c defeats f ($9W$ in each case); d defeats b and c, e defeats a and c, f defeats a and b ($9W$ in each case). So, the only pairwise elections left to be determined are the ones between a, b and c. We now specify the remaining $8K-1$ votes in S: there are $2K-1$ votes (c, b, a, p, d, e, f) and $2K-1$ votes (b, a, c, p, d, e, f), and 1 vote (b, c, a, p, d, e, f). As a result, given all the votes in S, b is $4K-1$ votes ahead of a in their pairwise election, b is 1 vote ahead of c, and c is 1 vote ahead of a. We claim that the votes in T can be cast so as to make a defeat b, b defeat c, and c defeat a, if and only if a partition of the k_i exists. If a partition exists, let the votes corresponding to one half of the partition be (a, b, c, p, d, e, f), and those corresponding to the other half be (c, a, b, p, d, e, f); this is easily verified to yield the desired result. Conversely, suppose there is a way to cast the votes in T so as to yield the desired result. Then, since each vote in T has even weight, all the votes in T rank a above b; at least half the vote weight ranks b above c; and at least half the vote weight ranks c above a. Since, if a is ranked above b, it is impossible to have c be ranked simultaneously both below b and above a, it follows that precisely half the vote weight ranks b above c (and the other half, c above a); and hence, we have a partition. To complete the proof of the theorem, we claim that making a defeat b, b defeat c, and c defeat a strictly maximizes the probability that p wins. From this claim it follows that if we set r to a number slightly smaller than the probability that p wins *if* a defeats b, b defeats c, and c defeats a, then it is possible for the votes in T to make p win with probability at least r, if and only if there is a partition of the k_i. To prove the claim, we do a careful case-by-case analysis on the structure of the cup.

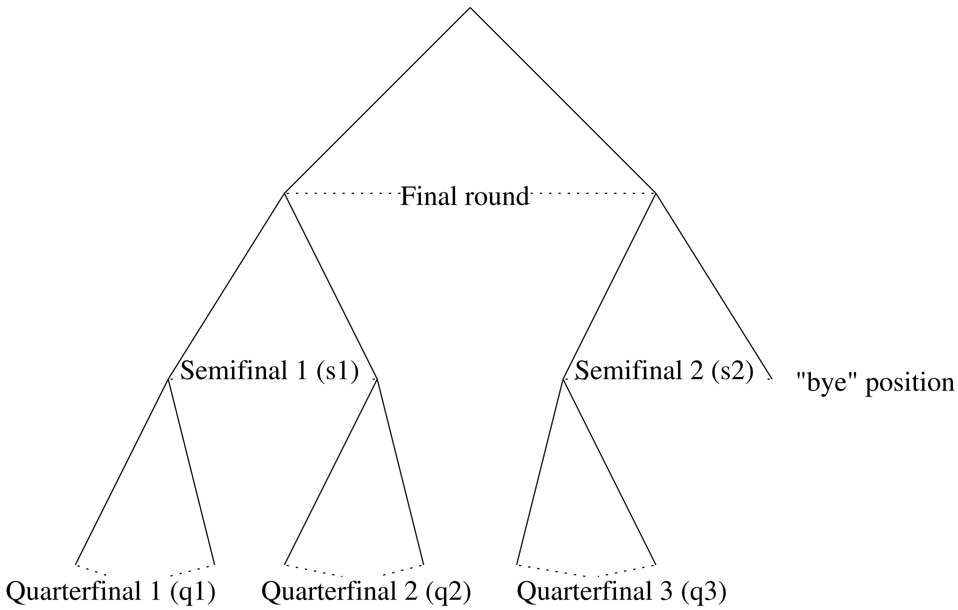


Figure 8.1: *The cup used in the proof.*

For each situation in which two of a, b and c face each other in a round, we analyze whose winning would be most favorable to p. If p does not get the "bye" position, it can only win by facing each of a, b and c in some round, which is impossible if any two of those ever face each other; so in this case the results of the pairwise elections between them are irrelevant as far as p's chances of winning are concerned. So let us assume p gets the bye position. If two of a, b and c face each other in s_1, then the outcome of this round is irrelevant as p would defeat either one in the final. If two of a, b and c face each other in q_3, then the outcome of this round is irrelevant as p would defeat either one in s_2. Now suppose a and b face each other in q_1. Then the only way for p to make it to the final is if c faces f in q_3, so let us assume this happens. Then, d and e face each other in q_2, which confrontation d will win. If a defeats b in q_1, it will win s_1 against d, and p will defeat it in the final. On the other hand, if b defeats a in q_1, it will lose s_1 against d, and d will defeat p in the final.. It follows that in this case, if we want p to win, we would (strictly) prefer a to defeat b in their pairwise election. Symmetrically, we also prefer a to defeat b if they face each other in q_2. Since we have analyzed all possibilities where a may face b, and in these is always favorable to p for a to defeat b, sometimes strictly so; it follows that it is strictly favorable to p's chances of winning if a defeats b in their pairwise election. Analogously (or by symmetry), it can be shown that it is strictly favorable to p's chances of winning if b defeats c, and c defeats b in their pairwise elections. Hence achieving these pairwise results simultaneously strictly maximizes p's chance of winning the election. ■

Recall that the cup rule (without randomization) is easy to manipulate constructively for any number of candidates. Thus, the previous result shows that *randomizing over instantiations of the rules* (such as schedules of a cup) *can be used to make manipulation hard.*

Destructive Manipulation

We now present our results for destructive manipulation.

We begin by laying out some cases where destructive manipulation can be done in polynomial time. It is easy to see that destructive manipulation can never be harder than constructive manipulation (except by a factor m) because in order to solve the former, we may simply solve the latter once for each candidate besides h. Thus, we immediately know that the plurality and cup rules can be destructively manipulated in polynomial time, for any number of candidates. Interestingly, for most of the other rules under study, destructive manipulation turns out to be drastically easier than constructive manipulation! The following theorem shows a sufficient condition for rules to be easy to manipulate destructively.

Theorem 79 *Consider any voting rule where each candidate receives a numerical score based on the votes, and the candidate with the highest score wins. Suppose that the score function is monotone, that is, if voter i changes its vote so that $\{b : a \succ_i^{old} b\} \subseteq \{b : a \succ_i^{new} b\}$ (here, $a \succ_i b$ means that voter i prefers a to b), a's score will not decrease. Finally, assume that the winner can be determined in polynomial time. Then for this rule, destructive manipulation can be done in polynomial time.*

Proof: Consider the following algorithm: for each candidate a besides h, we determine what the outcome of the election would be for the following coalitional vote. All the colluders place a at the top of their votes, h at the bottom, and order the other candidates in whichever way. We claim there is a vote for the colluders with which h does not win if and only if h does not win in one of these $m - 1$ elections. The *if* part is trivial. For the *only if* part, suppose there is a coalitional vote that makes $a \neq h$ win the election. Then, in the coalitional vote we examine where a is always placed on top and h always at the bottom, by monotonicity, a's score cannot be lower (because for each manipulator i, $\{b : a >_i b\}$ is maximal) and h's cannot be higher (because for each manipulator i, $\{b : h >_i b\}$ is minimal) than in the successful coalitional vote. It follows that here, too, a's score is higher than h's, and hence h does not win the election. The algorithm is in P since we do $m - 1$ winner determinations, and winner determination is in P. ■

Corollary 5 *Destructive manipulation can be done in polynomial time for the veto, Borda, Copeland, and maximin rules.*

Theorem 79 does not apply to the STV and plurality with runoff rules. We now show that destructive manipulation is in fact hard for these rules, even with only 3 candidates.

Theorem 80 *For the* STV *rule with 3 candidates,* DESTRUCTIVE CW-MANIPULATION *is NP-complete.*

Proof: Showing the problem is in NP is easy. To show it is NP-hard, we reduce an arbitrary PARTITION instance to the following DESTRUCTIVE CW-MANIPULATION instance. The 3 candidates are a, b and h. In S there are $6K$ voters voting (a, h, b), $6K$ voters voting (b, h, a), and $8K - 1$ voters voting (h, a, b). In T, for every k_i there is a vote of weight $2k_i$. We show the instances are equivalent.

We first observe that h will not win if and only if it gets eliminated in the first round: for if it survives the first round, either a or b gets eliminated in the first round. Hence either all the votes in S that ranked a at the top or all those that ranked b at the top will transfer to h, leaving h with at least $14K - 1$ votes in the final round out of a total of $24K - 1$, so that h is guaranteed to win the final round.

Now, if a partition of the k_i exists, let the votes in T corresponding to one half of the partition vote (a, b, h), and let the other ones vote (b, a, h). Then in the first round, a and b each have $8K$ votes, and h only has $8K - 1$ votes, so that h gets eliminated. So there exists a manipulation.

On the other hand, if a manipulation exists, we know by the above that with this manipulation, h is eliminated in the first round. Hence at least $2K-1$ of the vote weight in T ranks a at the top, and at least $2K-1$ of the vote weight in T ranks b at the top. Let A be the set of all the k_i corresponding to votes in T ranking a at the top; then $\sum_{k_i \in A} k_i \geq K - \frac{1}{2}$, and since the k_i are integers this implies $\sum_{k_i \in A} k_i \geq K$. If we let B be the set of all the k_i corresponding to votes in T ranking b at the top, then similarly, $\sum_{k_i \in B} k_i \geq K$. Since A and B are disjoint, it follows that $\sum_{k_i \in A} k_i = \sum_{k_i \in B} k_i = K$. So there exists a partition. ■

This result also allows us to establish the hardness of destructive manipulation in the plurality with runoff rule:

Theorem 81 *For the* plurality with runoff *rule with 3 candidates,* DESTRUCTIVE CW-MANIPULATION *is NP-complete.*

Proof: Showing the problem is in NP is easy. To show it is NP-hard, we observe that with 3 candidates, plurality with runoff coincides with STV, and DESTRUCTIVE MANIPULATION for STV with 3 candidates is NP-hard, as we proved in Theorem 80. ■

8.3.4 Effect of uncertainty about others' votes

So far we have discussed the complexity of coalitional manipulation when the others' votes are known. We now show how those results can be related to the complexity of manipulation by an individual voter when only a *distribution* over the others' votes is known. If we allow for arbitrary distributions, we need to specify a probability for each possible combination of votes by the others, that is, exponentially many probabilities (even with just two candidates). It is impractical to specify so many probabilities.[11] Therefore, we should acknowledge that it is likely that the language used

[11] Furthermore, if the input is exponential in the number of voters, an algorithm that is exponential in the number of voters is not necessarily complex in the usual sense of input complexity.

for specifying these probabilities would not be fully expressive (or would at least not be very convenient for specifying complex distributions). We derive the complexity results of this subsection for extremely restricted probability distributions, which any reasonable language should allow for. Thus our results apply to any reasonable language. We only present results on constructive manipulations, but all results apply to the destructive cases as well and the proofs are analogous. We restrict our attention to deterministic rules.

Weighted voters

First we show that with weighted voters, in rules where coalitional manipulation is hard in the complete-information case, even evaluating a candidate's winning probability is hard when there is uncertainty about the votes (even when there is no manipulator).

Definition 44 (WEIGHTED EVALUATION) *We are given a weight for each voter, a distribution over all possible vectors of votes, a candidate p, and a number r, where $0 \leq r \leq 1$. We are asked whether the probability of p winning is greater than r.*

Theorem 82 *If* CONSTRUCTIVE CW-MANIPULATION *is NP-hard for a deterministic rule (even with k candidates), then* WEIGHTED EVALUATION *is also NP-hard for it (even with k candidates), even if $r = 0$, the votes are drawn independently, and only the following types of (marginal) distributions are allowed: 1) the vote's distribution is uniform over all possible votes, or 2) the vote's distribution puts all of the probability mass on a single vote.*

Proof: For the reduction from CONSTRUCTIVE CW-MANIPULATION to WEIGHTED EVALUATION, we use exactly the same voters, and p remains the same as well. If a voter was not a colluder in the CONSTRUCTIVE CW-MANIPULATION instance and we were thus given its vote, in the WEIGHTED EVALUATION instance its distribution places all of the probability mass on that vote. If the voter was in the collusion, its distribution is now uniform. We set $r = 0$. Now, clearly, in the WEIGHTED EVALUATION instance there is a chance of p winning if and only if there exists some way for the latter votes to be cast so as to make p win - that is, if and only if there is an effective collusion in the CONSTRUCTIVE CW-MANIPULATION problem. ■

Next we show that if evaluating the winning probability is hard, individual manipulation is also hard.

Definition 45 (CONSTRUCTIVE INDIVIDUAL WEIGHTED (IW)-MANIPULATION UNDER UNCERTAINTY) *We are given a single manipulative voter with a weight, weights for all the other voters, a distribution over all the others' votes, a candidate p, and a number r, where $0 \leq r \leq 1$. We are asked whether the manipulator can cast its vote so that p wins with probability greater than r.*

Theorem 83 *If* WEIGHTED EVALUATION *is NP-hard for a rule (even with k candidates and restrictions on the distribution), then* CONSTRUCTIVE IW-MANIPULATION UNDER UNCERTAINTY *is also NP-hard for it (even with k candidates and the same restrictions).*

Proof: For the reduction from WEIGHTED EVALUATION to CONSTRUCTIVE IW-MANIPULATION UNDER UNCERTAINTY, simply add a manipulator with weight 0. ■

Combining Theorems 82 and 83, we find that with weighted voters, if in some rule coalitional manipulation is hard in the complete-information setting, then even individual manipulation is hard if others' votes are uncertain. Applying this to the hardness results from Subsection 8.3.3, this means that all of the rules of this section other than plurality and cup are hard to manipulate by individuals in the weighted case when the manipulator is uncertain about the others' votes.

Finally, we show that WEIGHTED EVALUATION can be hard even if CONSTRUCTIVE CW-MANIPULATION is not. If we relax the requirement that a vote is represented by a total order over the candidates, we can also allow for the following common voting rule:

- *approval.* Each voter labels each candidate as either approved or disapproved. The candidate that is approved by the largest number of voters wins.

For the approval rule, CONSTRUCTIVE CW-MANIPULATION is trivial: the universally most potent manipulation is for all of the manipulators to approve the preferred candidate p, and to disapprove all other candidates. However, WEIGHTED EVALUATION is hard:

Theorem 84 *In the approval rule,* WEIGHTED EVALUATION *is NP-hard, even if* $r = 0$*, the votes are drawn independently, and the distribution over each vote has positive probability for at most 2 of the votes.*

Proof: We reduce an arbitrary PARTITION instance to the following WEIGHTED EVALUATION instance. There are 3 candidates, p, a, and b. There are $2K + 1$ votes approving $\{p\}$. Additionally, for each k_i in the PARTITION instance, there is a vote of weight $2k_i$ that approves $\{a\}$ with probability $\frac{1}{2}$, and $\{b\}$ with probability $\frac{1}{2}$. We set $r = 0$. Clearly, p wins if and only if a and b are each approved by precisely $2K$ of the vote weight. But this is possible (and happens with positive probability) if and only if there is a partition. ■

Unweighted voters

Finally, we study what implications can be derived for the hardness of manipulation in settings with unweighted voters.

Definition 46 UNWEIGHTED EVALUATION *is the special case of* WEIGHTED EVALUATION *where all the weights are* 1. CONSTRUCTIVE INDIVIDUAL UNWEIGHTED (IU)-MANIPULATION UNDER UNCERTAINTY *is the special case of* CONSTRUCTIVE INDIVIDUAL WEIGHTED (IW)-MANIPULATION UNDER UNCERTAINTY *where all the weights are* 1.

First, we show that for rules for which WEIGHTED EVALUATION is hard, UNWEIGHTED EVALUATION is also hard. This assumes that the language for specifying the probability distribution is rich enough to allow for perfect correlations between votes (that is, some votes are identical with probability one[12]).

[12] Representation of such distributions can still be concise.

Theorem 85 *If* WEIGHTED EVALUATION *is NP-hard for a rule (even with k candidates and restrictions on the distribution), then* UNWEIGHTED EVALUATION *is also NP-hard for it if we allow for perfect correlations (even with k candidates and the same restrictions—except those conflicting with perfect correlations). (This is assuming that a group of κ perfectly correlated votes can be represented using only $O(\log(\kappa))$ space.)*

Proof: For the reduction from WEIGHTED EVALUATION to its unweighted version, we replace each vote of weight κ with κ unweighted votes; we then make these κ votes perfectly correlated. Subsequently we pick a representative vote from each perfectly correlated group, and we impose a joint distribution on this vote identical to the one on the corresponding vote in the WEIGHTED EVALUATION problem. This determines a joint distribution over all votes. It is easy to see that the distribution over outcomes is the same as in the instance from which we reduced; hence, the decision questions are equivalent. ∎

We would like to have an analog of Theorem 83 here, to show that UNWEIGHTED EVALUATION being hard also implies that CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY is hard. Unfortunately, the strategy used in the proof of Theorem 83—setting the manipulator's weight to 0—does not work, because the weight of the manipulator must now be 1. Instead, we rely on the following two theorems, which each require an additional precondition. The first one shows that if the WEIGHTED EVALUATION problem is hard even in settings where there is no possibility that the candidate p is tied for winning the election, then the CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY problem is also hard.

Theorem 86 *If* WEIGHTED EVALUATION *is NP-hard for a rule even in settings where ties will not occur (even with k candidates and restrictions on the distribution of the votes), then* CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY *is also NP-hard (with the same r) if we allow for perfect correlations (even with k candidates and the same restrictions on the distribution of the nonmanipulators' votes—except those conflicting with perfect correlations). (This is assuming that a group of κ perfectly correlated votes can be represented using only $O(\log(\kappa))$ space.)*

Proof: We reduce the EVALUATION instance to a MANIPULATION instance by first adding a single manipulator. Because ties will not occur, there must exist a (rational) weight $w > 0$ such that if the manipulator's vote has this weight, then the manipulator's vote will never affect the outcome. Without loss of generality, we can assume that this weight can be written as $w = \frac{1}{M}$ for some sufficiently large integer M. Now, multiply all the weights by M so that the manipulator's vote has weight 1 and all the weights are integers again. Then, replace each voter of weight κ by κ perfectly correlated, unweighted voters. Clearly, the manipulator will still not affect the outcome, and thus the distribution over outcomes is the same as in the instance we reduced from; hence, the decision questions are equivalent. ∎

Theorem 86 applies to most of the rules under study:

Corollary 6 *For each one of the following rules,* CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY *is NP-hard: Borda (even with 3 candidates), veto (even with 3 candidates), STV (even* *with 3 candidates), plurality with runoff (even with 3 candidates), and maximin (even with 4 candidates). This holds even if $r = 0$, the votes are either drawn independently or perfectly correlated, and only the following types of (marginal) distributions are allowed: 1) the vote's distribution is uniform over all possible votes, or 2) the vote's distribution puts all of the probability mass on a single vote. (This is assuming that a group of κ perfectly correlated votes can be represented using only $O(\log(\kappa))$ space.)*

Proof: To show that we can apply Theorem 86 to each of these rules, we first make the following observation. For each of these rules, the reduction that we gave to show that CONSTRUCTIVE CW-MANIPULATION is hard has the property that if there exists no successful manipulation, candidate p cannot even be tied for winning the election (and, of course, if there is a successful manipulation, no other candidate will tie with p for winning the election). Because of this, when we apply the reduction from Theorem 82 to these instances, there is no chance that a tie for winning the election between p and another candidate will occur, and we can apply Theorem 86. ■

Unfortunately, for the Copeland rule, in the reduction given in Theorem 74, an unsuccessful manipulation may still leave p tied for winning the election. To show that CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY is hard for this rule as well, we need the following theorem:

Theorem 87 *If* WEIGHTED EVALUATION *is NP-hard for a rule even in settings where $r = 0$ and one of the voters with a uniform distribution over votes has weight 1 (even with k candidates and restrictions on the distribution of the votes), then* CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY *is also NP-hard even in settings where $r = 0$ if we allow for perfect correlations (even with k candidates and the same restrictions on the distribution of the nonmanipulators' votes—except those conflicting with perfect correlations). (This is assuming that a group of κ perfectly correlated votes can be represented using only $O(\log(\kappa))$ space.)*

Proof: We reduce the EVALUATION instance to a MANIPULATION instance by replacing the voter with a uniform distribution over votes and weight 1 by the manipulator, and using the same distribution over the other voters' votes as before. If there is nonzero probability of p winning in the EVALUATION instance, then there must exists some vector of votes with nonzero probability for which p wins with nonzero probability. Then, in the MANIPULATION instance, consider the vote in this vote vector cast by the voter that was replaced by the manipulator. If the manipulator places this vote, then with nonzero probability, the same vector will occur and p will win with nonzero probability. Conversely, suppose that in the MANIPULATION instance there exists a vote for the manipulator such that p wins with nonzero probability. Then, in the EVALUATION instance, there is some nonzero probability that the voter replaced by the manipulator casts this vote (because that voter's distribution over votes is uniform). It follows that there is nonzero probability that p will win in the EVALUATION instance. Hence, the decision questions are equivalent. ■

Corollary 7 *For the Copeland rule (even with 4 candidates),* CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY *is NP-hard . This holds even if $r = 0$, the votes are either drawn independently or perfectly correlated, and only the following types of (marginal) distributions are allowed:*

1) the vote's distribution is uniform over all possible votes, or 2) the vote's distribution puts all of the probability mass on a single vote. (This is assuming that a group of κ perfectly correlated votes can be represented using only $O(\log(\kappa))$ space.)

Proof: Because PARTITION is hard even when one of the integers to be partitioned is 1, we can assume that one of the manipulators in the proof of Theorem 74 has weight 1, which allows us to apply Theorem 87. ■

As a final remark, we observe that the manipulation questions discussed in this subsection are not necessarily even in NP. However, when $r = 0$, the manipulation question can also be phrased as saying "does there exist a manipulation that has *some* chance of succeeding?" We note that this question is in fact in NP.

The following figure summarizes the flow of the theorems presented in this subsection.

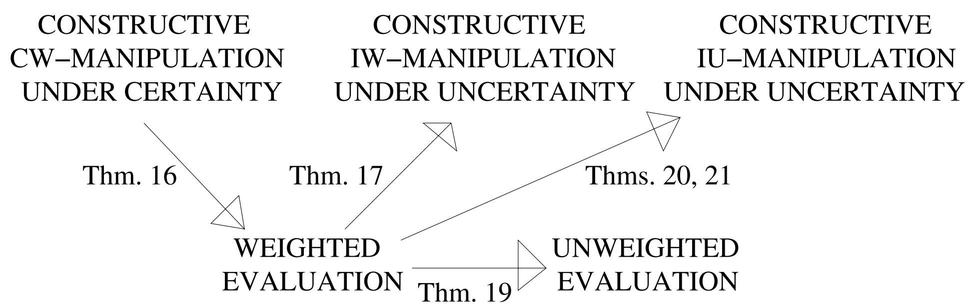


Figure 8.2: *The flow of the theorems in this subsection.*

This concludes the part of the dissertation studying worst-case hardness of manipulation. In the next section, we move on to a more ambitious goal: voting rules that are *usually* hard to manipulate.

8.4 Nonexistence of usually-hard-to-manipulate voting rules

One weakness that all of the above results have in common is that they only show *worst-case* hardness. That is, the results show that it is unlikely that an efficient algorithm can be designed that finds a beneficial manipulation in *all* instances for which a beneficial manipulation exists. However, this does not mean that there do not exist efficient manipulation algorithms that find a beneficial manipulation in *many* instances. If such algorithms do in fact exist, then computational hardness constitutes a leaky barrier to manipulation at best (though it is presumably still better than nothing).

A truly satisfactory solution to the problem would be to have a rule that is hard to manipulate in *all* instances. However, this is too much to ask for: for example, a manipulation algorithm could have a small database of instances with precomputed solutions, and it would merely need to check against this database to successfully manipulate some instances. Still, we may ask whether it is possible to make (say) 99% of instances hard to manipulate. It is generally agreed that this would have a much greater impact on the design of voting rules in practice than merely worst-case hardness [Conitzer *et al.*, 2003; Elkind and Lipmaa, 2005b], but none of the multiple efforts to achieve this objective have succeeded.

In this section, we present an impossibility result that makes it seem unlikely that such an objective can be achieved by any reasonable voting rule. This is not the first such impossibility result: a previous result [Procaccia and Rosenschein, 2006] shows that a specific subclass of voting rules is usually easy to manipulate when the number of candidates is constant and a specific distribution over instances is used (where the distribution is chosen to have certain properties that would appear to make manipulation more difficult). By contrast, our result does not require any restriction on the voting rule, number of candidates, or distribution over instances. Our result states that a voting rule/instance distribution pair cannot simultaneously be usually hard to manipulate, and have certain natural properties (which depend on the rule and distribution).

8.4.1 Definitions

Manipulation

As we saw in the previous section, the computational problem of manipulation has been defined in various ways, but typical definitions are special cases of the following general problem: given the nonmanipulators' votes, can the manipulator(s) cast their votes in such a way that one candidate from a given set of preferred candidates wins? In this section, we study a more difficult manipulation problem: we require that the manipulator(s) find the set of *all* the candidates that they can make win (as well as votes that will bring this about). This stronger requirement makes our impossibility result stronger: we will show that even this more powerful type of manipulation cannot be prevented.

Definition 47 *A* manipulation instance *is given by a voting rule R, a vector of* nonmanipulator votes $v = (r_1^{NM}, \ldots, r_n^{NM})$*, a vector of* weights $v^s = (d_1^{NM}, \ldots, d_n^{NM})$ *for the nonmanipulators, and a vector of weights $w^s = (d_1^M, \ldots, d_k^M)$ for the manipulators. A* manipulation algorithm succeeds *on this instance if it produces a set of pairs $\{(w_{i_1}, c_{i_1}), \ldots, (w_{i_q}, c_{i_q})\}$ such that 1) if the manipulators cast the vector of votes w_{i_j}, then c_{i_j} wins, and 2) if a candidate c does not occur in this set as one of the c_{i_j}, then there is no vector of manipulator votes w that makes c win.*

An instance is *manipulable* if the manipulators can make more than one candidate win. Nonmanipulable instances are easy to solve: any algorithm that is sound (in that it does not produce incorrect (w_{i_j}, c_{i_j}) pairs) and that returns at least one (w_{i_j}, c_{i_j}) pair (which is easy to do, by simply checking what the rule will produce for a given vector of votes) will succeed. Hence, we focus on manipulable instances only.

To investigate whether voting rules are *usually* easy to manipulate, we also need a probability distribution over (manipulable) instances. Our impossibility result does not require a specific distribution, but in the experimental subsection of the section, we study a specific family of distributions.

Weak monotonicity

Informally, a rule is *monotone* if ranking a candidate higher never hurts that candidate. All the rules mentioned before, with the exceptions of STV and plurality with runoff, are monotone. In this subsubsection, we formally define a weak notion of monotonicity that is implied by (but does not imply) standard notions of monotonicity. We note that we want our definition of monotonicity to be as weak as possible so that our impossibility result will be as strong as possible. We define when an *instance* is weakly monotone, so that even rules that are not (everywhere) weakly monotone can be (and typically are) weakly monotone on most instances.

We will first define a stronger, more standard notion of monotonicity. For rules that produce a score for every candidate, a natural definition of monotonicity is the following: if a manipulator changes his vote so that a given candidate is ranked ahead of a larger set of candidates, then that candidate's score should not decrease. However, not every rule produces a score. Thus, we use the following definition of monotonicity, which does not rely on scores. (Monotonicity as defined above for rules that produce a score implies monotonicity in the sense of the following definition.)

Definition 48 *We say that a voting rule R is* monotone *for manipulators with weights w^s and nonmanipulator votes v with weights v^s if for every pair of candidates c_1, c_2 and every pair of manipulator vote vectors $w_1 = (r_1^1, \ldots, r_k^1), w_2 = (r_1^2, \ldots, r_k^2)$, the following condition holds: if*

- *c_2 wins when the manipulators vote w_1, and*
- *for any manipulator i, for any candidate c such that $c_2 \succ_{r_i^1} c$, we have $c_2 \succ_{r_i^2} c$, and*
- *for any manipulator i, for any candidate c such that $c \succ_{r_i^1} c_1$, we have $c \succ_{r_i^2} c_1$;*

then c_1 does not win when the manipulators vote w_2.

Thus, given a monotone instance, if each manipulator decreases the set of candidates that he prefers to the current winner, and increases the set of candidates that he prefers to a given other candidate, then the latter candidate cannot become the winner due to this. (It is, however, possible that a third candidate will win after the change, since we did not restrict how that candidate's position in the ranking changed.)

Now we can define our weaker notion of monotonicity:

Definition 49 *We say that a voting rule R is* weakly monotone *for manipulators with weights w^s and nonmanipulator votes v with weights v^s if for every pair of candidates c_1, c_2, one of the following conditions holds: 1) c_2 does not win for any manipulator votes; or 2) if all the manipulators rank c_2 first and c_1 last, then c_1 does not win.*

We now show that our notion is indeed weaker:

Theorem 88 *Monotonicity implies weak monotonicity.*

Proof: Given a monotone rule R, consider any pair of candidates c_1, c_2, and votes $w_2 = (r_1^2, \ldots, r_k^2)$ for the manipulators in which c_2 is always ranked first and c_1 is always ranked last. Suppose that there are votes $w_1 = (r_1^1, \ldots, r_k^1)$ that make c_2 win. Then if the manipulators changed from w_1 to w_2, every manipulator would decrease the set of candidates that he prefers to c_2 and increase the set of candidates that he prefers to c_1. Hence, by monotonicity, c_1 cannot win. ■

On the other hand, weak monotonicity does not imply monotonicity. For instance, consider the scoring rule defined (for four candidates) by $\langle 3, 1, 2, 0 \rangle$. Ranking a candidate second instead of third can end up hurting that candidate, so this rule is clearly not monotone. However, under this rule, if all the manipulators rank candidate c_2 first and c_1 last, and c_1 still wins, then c_1 must be at least $3k$ points ahead of c_2 (not counting the manipulators' votes), so c_2 does not win for any manipulator votes. Hence, the rule does satisfy weak monotonicity.

8.4.2 Impossibility result

We now present an algorithm that seeks to identify two candidates that can be made to win. The algorithm is universal in that it does not depend on the voting rule used, except for the places in which it calls the rule as a (black-box) subroutine.

```
Find-Two-Winners(R, C, v, v^s, w^s)
choose an arbitrary manipulator vote vector w_1
c_1 ← R(C, v, v^s, w_1, w^s)
for every c_2 ∈ C, c_2 ≠ c_1 {
  choose w_2 in which every vote ranks c_2 first and c_1
last
  c ← R(C, v, v^s, w_2, w^s)
  if c_1 ≠ c return {(w_1, c_1), (w_2, c)} }
return {(w_1, c_1)}
```

If voting rule R can be executed in polynomial time, then so can Find-Two-Winners.

Theorem 89 Find-Two-Winners *will succeed on every instance that both a) is weakly monotone and b) allows the manipulators to make either of exactly two candidates win.*

Proof: Find-Two-Winners is sound in the sense that it will never output a manipulation that is incorrect. It will certainly find one candidate that the manipulators can make win (c_1). Thus, we merely need to show that it will find the second candidate that can be made to win; let us refer to this candidate as c'. If the algorithm reaches the iteration of the **for** loop in which $c_2 = c'$, then in this round, either $c \neq c_1$, in which case we must have $c = c'$ because there are no other candidates that can be made to win, or $c = c_1$. But in the latter case, we must conclude that $c_2 = c'$ cannot be made to win, due to weak monotonicity—which is contrary to assumption. Hence it must be that $c = c'$. If the algorithm does not reach the iteration of the **for** loop in which $c_2 = c'$, it must have found a manipulation that produced a winner other than c_1 in an earlier iteration, and (by assumption) this other winner can only be c'. ■

It is not possible to extend the algorithm so that it also succeeds on all weakly monotone instances in which *three* candidates can be made to win. When three candidates can be made to win, even under monotone rules, it is possible that one of these candidates can only win if some manipulators vote differently from the other manipulators. In fact, as we saw in the previous section, the problem of deciding whether multiple weighted manipulators can make a given candidate win is NP-complete, even when there are only three candidates and the Borda or veto rule is used (both of which are monotone rules). Any algorithm that does succeed on all weakly monotone instances in which at most three candidates can be made to win would be able to solve this NP-complete problem, and thus cannot run in polynomial time (or it would show that P = NP).

The impossibility result now follows as a corollary.

Corollary 8 *For any $p \in [0, 1]$, there does not exist any combination of an efficiently executable voting rule R and a distribution d over instances such that*

1. *the probability of drawing an instance that is both a) weakly monotone, and b) such that either of exactly two candidates can be made to win, is at least p; and*

2. *for any computationally efficient manipulation algorithm, the probability that an instance is drawn on which the algorithm succeeds is smaller than p.*

This impossibility result is relevant only insofar as one expects a voting rule to satisfy Property 1 in the corollary (high probability of drawing a weakly monotone instance in which either of exactly two candidates can be made to win). Before we argue why one should in fact expect this, it is helpful to consider how a skeptic might argue that an impossibility result such as this one is irrelevant. At a minimum, the skeptic should argue that one of the properties required by the result is not sufficiently desirable or necessary to insist on it. The skeptic could make her case much stronger by actually exhibiting a voting rule that satisfies all properties except for the disputed one, and that still seems intuitively desirable. (For example, Arrow's impossibility result [Arrow, 1963] is often criticized on the basis that the *independence of irrelevant alternatives* property is unnecessarily strong, and this is the only property that common voting rules fail to satisfy.)

Conversely, we will first argue directly for the desirability of Property 1 in Corollary 8. We will then provide indirect evidence that it will be difficult to construct a sensible rule that does not satisfy this property, by showing experimentally that all common rules satisfy it very strongly (that is, a large fraction of manipulable instances are weakly monotone and such that only two candidates can be made to win).

8.4.3 Arguing directly for Property 1

In this subsection, we argue why one should expect many manipulable instances to be both a) weakly monotone and b) such that the manipulator(s) can make either of exactly two candidates win. We first make a simple observation: if manipulable instances are usually weakly monotone, and they usually allow the manipulator(s) to make either of exactly two candidates win, then a significant fraction of manipulable instances have both of properties a) and b). More precisely:

Proposition 10 *If the probability of drawing a weakly monotone instance is p, and the probability of drawing an instance in which either of exactly two candidates can be made to win is q, then the probability of drawing an instance with both properties is at least $p + q - 1$.*

Proof: The probability of drawing an instance that is *not* weakly monotone is $1 - p$, and the probability of drawing an instance in which more than two candidates can be made to win is $1 - q$. From this, it follows that the probability of drawing an instance with both properties is at least $1 - (1 - p) - (1 - q) = p + q - 1$. ■

With this in mind, we will now argue separately for each of the two properties a) and b).

The argument for Property a)—most manipulable instances should be weakly monotone—is easy to make. The reason is that if the manipulators rank certain candidates higher, this should, in general, benefit those candidates. If this were not the case, then the manipulators' votes would lose their natural interpretation that they support certain candidates over others, and we are effectively asking the manipulators to submit a string of bits without any inherent meaning.[13] It should also be noted that most common voting rules are in fact monotone (on all instances), and the few rules for which nonmonotone instances can be constructed are often severely criticized because of this (even if the rule is in fact monotone on most instances).

Arguing for Property b)—most manipulable instances should be such that the manipulators can make either of exactly two candidates win—is somewhat more difficult. For simplicity, consider rules that produce a score for every candidate. As the number of voters grows, typically, candidates' scores tend to separate. This is especially the case if some candidates systematically tend to be ranked higher than others by voters, *e.g.* because these candidates are intrinsically better. (One interpretation of voting that dates back at least to Condorcet is the following: everyone has a noisy signal about the relative intrinsic quality of the candidates, and the purpose of an election is to maximize the probability of choosing the intrinsically best candidate [de Caritat (Marquis de Condorcet), 1785].) Thus, given a large number of nonmanipulators, it is unlikely that the scores of the two leading candidates will be close enough to each other that a few manipulators can make either one of them win; but it is significantly more unlikely that the scores of the *three* leading candidates will be close enough that the manipulators can make any one of them win. So, even given that some manipulation is possible, it is unlikely that more than two candidates can be made to win. This argument suggests that it is likely that most common voting rules in fact satisfy Property b). But it is also an argument for why we should *require* a voting rule to have this property, because, especially when we think of voting as being a process for filtering out the noise in voters' individual preferences to find the intrinsically best candidate, we *want* the candidates' scores to separate.

In the next subsection, we show experimentally that common voting rules in fact strongly satisfy properties a) and b).

8.4.4 Arguing experimentally for Property 1

In this subsection, we show experimentally that for all the common voting rules, most manipulable instances are in fact weakly monotone and such that either of exactly two candidates can be made to win. Because most of the rules that we study are in fact monotone on all instances, this mostly comes down to showing that at most two candidates can be made to win in most manipulable instances. Unfortunately, for quite a few of these rules, it is NP-hard to determine whether more than two candidates can be made to win (this follows from results in the previous section). Rather than trying to solve these NP-hard problems, we will be content to provide a *lower bound* on the fraction of manipulable instances in which either of exactly two candidates can be made to win. We obtain these lower bounds by characterizing, for each rule that we study, an easily computable sufficient (but not necessary) condition for an instance to be such that either of exactly two candidates can be made to win. For at least some rules, the lower bound is probably significantly below the actual fraction—which only strengthens the relevance of the impossibility result.

One useful property of these lower bounds is that they are independent of how the manipulators' total weight is distributed. Because of this, only the manipulators' total weight matters for the

[13]Incidentally, if this does not bother us, it is easy to design rules that are always hard to manipulate: for example, we can count an agent's vote only if part of its vote (represented as a string of bits) encodes the solution to (say) a hard factoring problem. Of course, this is not a very satisfactory voting rule.

purpose of our experiments, and we can assume, without loss of generality, that each manipulator has weight 1.

It should be noted that we can only show results for specific distributions of instances, because we need a specific distribution to conduct an experiment. Therefore, it cannot be said with certainty that other distributions would lead to similar results, although for reasonable distributions it appears likely that they would. One should keep in mind that the vote aggregator typically has no control over the distribution over voters' preferences, so that constructing an artificial distribution for which these results do not hold is unlikely to be helpful. We now present the specific distributions that we study.

For a given number of candidates, number of nonmanipulators, and number of manipulators, we generate instances as follows. (This is Condorcet's distribution, which we discussed in Chapter 3 because the maximum likelihood estimator of the correct ranking under this distribution is the Kemeny rule [Kemeny, 1959; Young, 1995].) We assume that there is a "correct" ranking t of the candidates (reflecting the candidates' unknown intrinsic quality), and the probability of drawing a given vote r is proportional to $p^{a(r,t)}(1-p)^{m(m-1)/2-a(r,t)}$, where $a(r,t)$ is the number of pairs of candidates on whose relative ranking r and t agree (they agree if either $c_1 \succ_r c_2$ and $c_1 \succ_t c_2$, or $c_2 \succ_r c_1$ and $c_2 \succ_t c_1$). p is a given noise parameter; if $p = 1$ then all voters will produce the correct ranking, and if $p = 0.5$ then we are drawing every vote (independently and uniformly) completely at random. This distribution is due to Condorcet [de Caritat (Marquis de Condorcet), 1785], and one way to interpret it is as follows. To draw a vote, for each pair of candidates c_1, c_2, randomly decide whether the vote is going to agree with the correct ranking on the relative ranking of c_1 and c_2 (with probability p), or disagee (with probability $1-p$). This may lead to cycles (such as $c_1 \succ c_2 \succ c_3 \succ c_1$); if so, restart.

These distributions often produce nonmanipulable instances. Ideally, we would discard all nonmanipulable instances, but this requires us to have an algorithm for detecting whether an instance is manipulable. If we know that the instance is weakly monotone, we can simply use algorithm Find-Two-Winners for this purpose. However, a few of the rules that we study (STV and plurality with runoff) are not monotone on all instances. In fact, for these rules, it is NP-hard to tell whether the instance is manipulable (this follows from results in the previous section). For these rules, we use simple sufficient (but not necessary) conditions to classify an instance as nonmanipulable. We will classify each nonmanipulable instance that does not satisfy this condition as having more than two potential winners, so that our results are still lower bounds on the actual ratio.

In the experiments below, we draw 1000 manipulable instances at random (by drawing and discarding instances as described above), and for each voting rule, we show our lower bound on the number of instances in which the manipulators can make either of exactly two candidates win. For rules that are not monotone everywhere, we also show a lower bound on the number of such instances that are *also* weakly monotone (indicated by "<rule> - monotone"). We also consider the Condorcet criterion—recall that a rule satisfies the Condorcet criterion if any candidate that wins all of its pairwise elections must win the overall election—and show a lower bound on the number of instances for which these properties are satisfied for *any* rule satisfying the Condorcet criterion.

In our first experiment (Figure 8.3), we have three candidates, one manipulator, and significant noise in the votes ($p = 0.6$). For all the rules under study, the fraction of instances satisfying the property approaches 1 as the number of nonmanipulator votes grows.

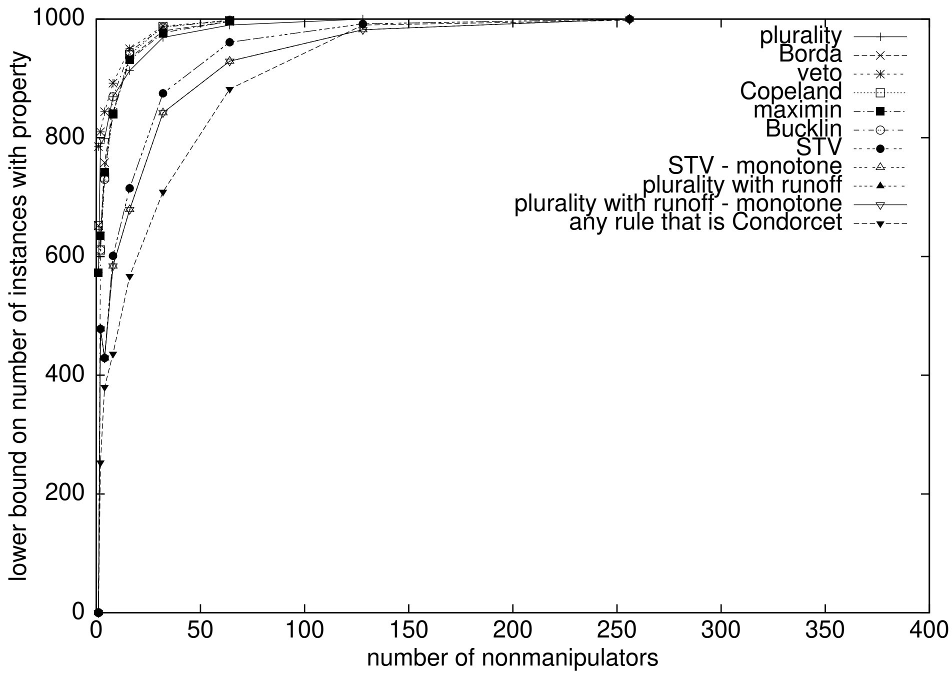


Figure 8.3: p = 0.6, one manipulator, three candidates.

Next, we show what happens when we maximize noise ($p = 0.5$), so that votes are drawn completely at random (Figure 8.4). Even under such extreme noise, the fraction of instances satisfying the property approaches 1 or at least becomes very large (> 0.7) for every one of the rules. However, it is no longer possible to say this for *any* rule satisfying the Condorcet criterion (although the specific common rules that do satisfy this criterion satisfy the property for a very large fraction of instances).

Next, we show results when there are multiple (specifically, 5) manipulators (Figure 8.5). The results are qualitatively similar to those in Figure 8.3, although for smaller numbers of nonmanipulators the fractions are lower. This makes sense: when the number of nonmanipulators is relatively small, a large coalition is likely to be able to make any candidate win.

Finally, we experiment with an increased number of candidates (Figure 8.6).

Now, the lower bound on the fraction of instances satisfying the property approaches 1 for all rules but STV. The lower fraction for STV is probably at least in part due to the fact that the lower bound that we use for STV is relatively weak. For example, any instance in which the manipulators can change the eliminated candidate in at least two rounds is counted as having more than two candidates that the manipulators can make win. This is extremely conservative because changes in which candidate is eliminated in a given round often do not change the winner.

8.4.5 Can the impossibility be circumvented?

One may wonder whether there are ways to circumvent the impossibility result presented in this section. Specifically, one may still be able to construct voting rules that are usually hard to ma-

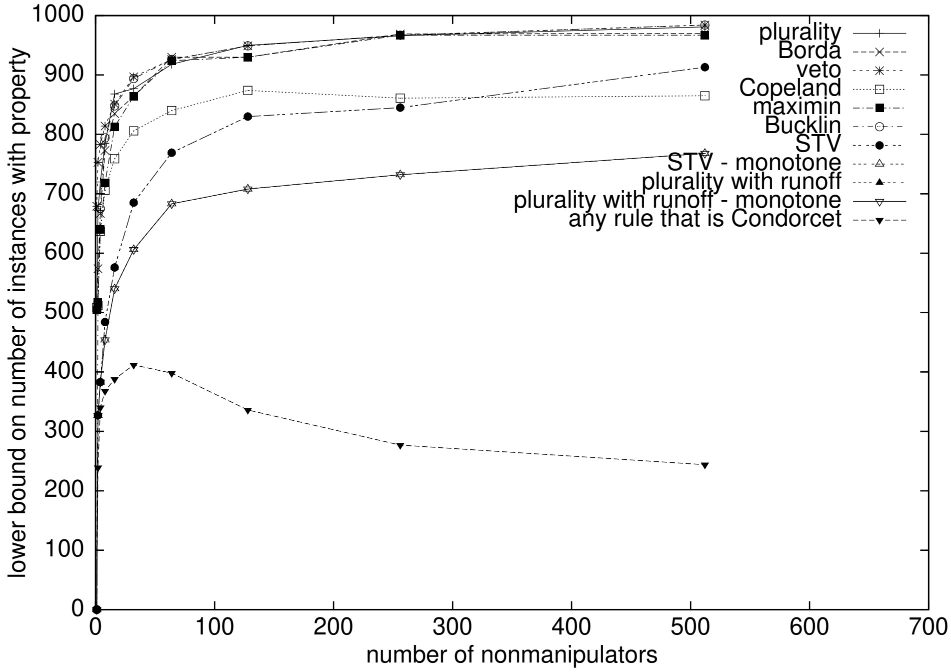


Figure 8.4: p = 0.5, one manipulator, three candidates.

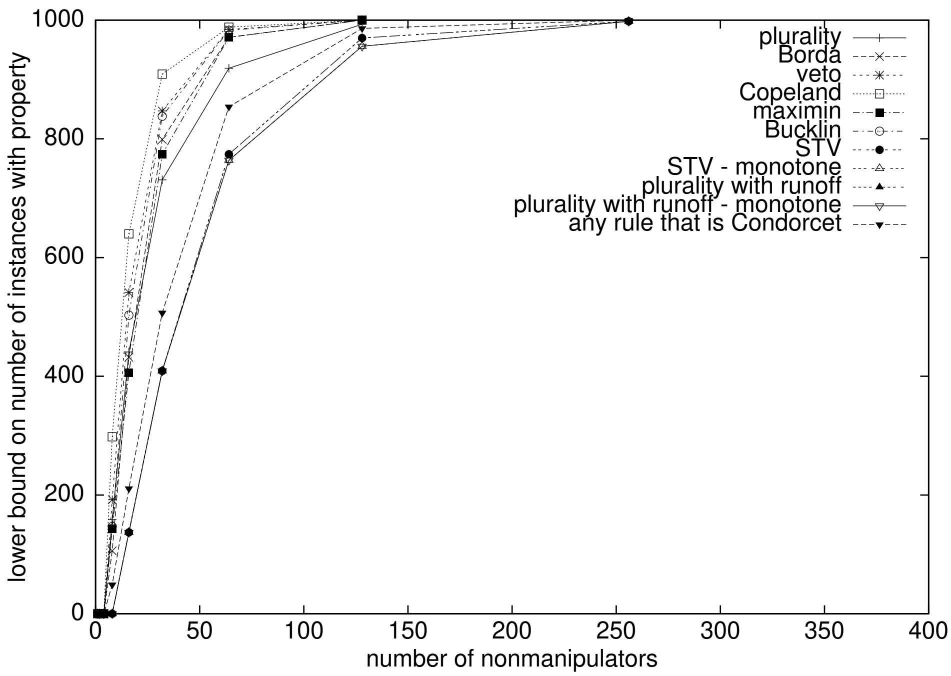


Figure 8.5: p = 0.6, five manipulators, three candidates.

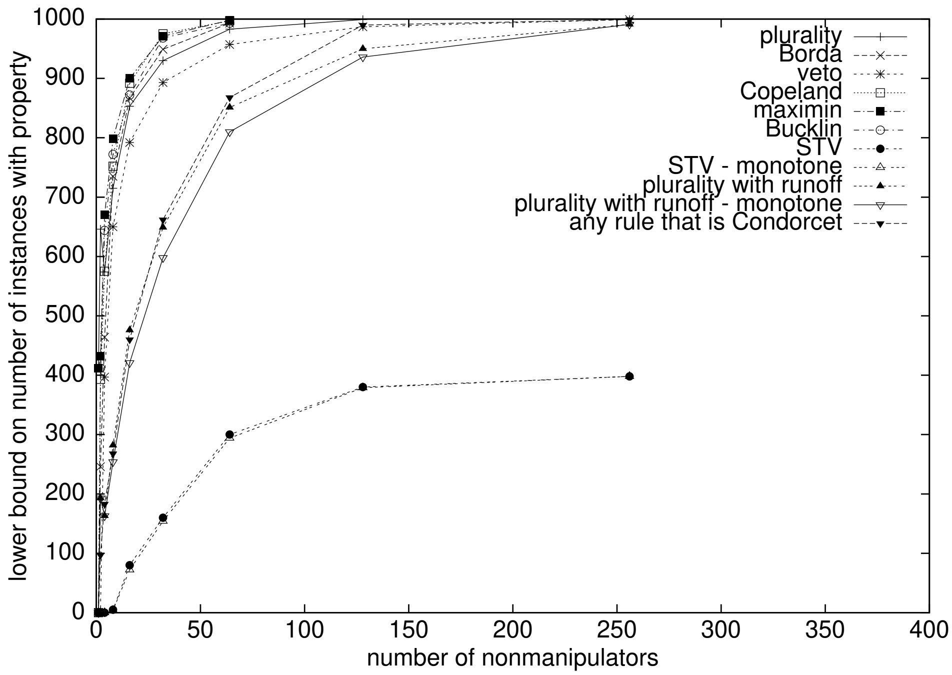


Figure 8.6: p = 0.6, one manipulator, five candidates.

nipulate by considering a larger class of voting rules, a class that contains rules that do not satisfy the preconditions of the impossibility result. In this subsection, we discuss various approaches for circumventing the impossibility result, and their prospects. (One approach that we will not discuss is that of constructing distributions over voters' preferences for which the impossibility result fails to hold, because, as we mentioned earlier, the distribution over voters' preferences is typically not something that the vote aggregator has any control over.)

Allowing low-ranked candidates to sometimes win

The impossibility result is only significant if in a sizable fraction of manipulable instances, only two candidates can be made to win. One may try to prevent this by using a voting rule that sometimes chooses as the winner a candidate that in fact did not do well in the votes (according to whatever criterion), thereby increasing the number of candidates that can be made to win in manipulable instances. Of course, having such a candidate win is inherently undesirable, but if it occurs rarely, it may be a price worth paying in order to achieve hardness of manipulation.

If we take this approach, and in addition allow for the rule to be *randomized*, then we can construct reasonable voting rules that are in fact strategy-proof (that is, no beneficial manipulation is ever possible). Consider, for example, the following voting rule:

Definition 50 *The* Copeland-proportional *rule chooses candidate c as the winner with probability proportional to c's Copeland score.*

An alternative interpretation of this rule is the following: choose a pair of candidates at random; the winner of their pairwise election wins the entire election. (If the pairwise election is tied, choose one of the two candidates at random.)

Theorem 90 *Copeland-proportional is strategy-proof.*

Proof: Suppose the manipulator knows which pair of candidates is chosen. Then, any vote in which he ranks his preferred candidate higher than the other candidate is strategically optimal. But the manipulator can guarantee that this is the case, even without knowing the pair of candidates, simply by voting truthfully. ■

Recall Gibbard [1977]'s result that a randomized voting rule is strategy-proof only if it is a probability mixture of unilateral and duple rules, where a rule is unilateral if only one voter affects the outcome, and duple if only two candidates can win. The Copeland-proportional rule only randomizes over duple rules (namely, pairwise elections).

Of course, the Copeland-proportional rule is still not ideal. For instance, even a Condorcet winner has a probability of only $(m-1)/(m(m-1)/2) = 2/m$ of winning under this rule. (However, this rule will at least never choose a candidate that loses every pairwise election.) Thus, it may be worthwhile to try to construct voting rules that are usually hard to manipulate, and that are more likely to choose a "good" winner than Copeland-proportional.

Expanding the definition of a voting rule

The impossibility result may cease to hold when the rule can choose from a richer outcome space. As in the previous subsubsection, this may prevent problems of manipulability completely, by allowing the construction of strategy-proof rules. For example, *if* payments are possible and the agents have quasilinear utility functions, then a payment scheme such as the VCG mechanism can induce strategy-proofness. As another example that does not require these assumptions, suppose that it is possible to exclude certain voters from the effects of the election—as an illustrative example, in an election for a country's president, suppose that it is possible to *banish* certain voters to another country, in which it will no longer matter to those voters who won the election. (It does not matter whether living in the other country is otherwise more or less desirable than in the original country.) Then, we can augment any voting rule as follows:

Definition 51 *For any voting rule (in the standard sense) R, the* banishing *rule $B(R)$ always chooses the same winner as R, and banishes every pivotal voter. (A voter is* pivotal *if, given the other votes, he can make multiple candidates win.)*

Theorem 91 *For any rule R, the banishing rule $B(R)$ is strategy-proof.*

Proof: A voter who is not pivotal has no incentive to misreport, because by definition, his vote does not affect which candidate wins, and he cannot affect whether he is pivotal. A voter who is pivotal also has no incentive to misreport, because he cannot affect whether he is pivotal, and the winner of the election will not matter to him because he will be banished. ■

However, this scheme also has a few drawbacks. For one, it may not always be possible to completely exclude a voter from the effects of the election. Another strange property of this scheme is that no voter is ever capable of affecting his own utility, so that *any* vote is strategically optimal. Finally, it may be necessary to banish large numbers of voters. In fact, the following lemma shows that for any rule, the votes may turn out to be such that more than half of the voters must be banished.

Theorem 92 *For any responsive voting rule R, it is possible that more than half the voters are simultaneously pivotal. (We say that a voting rule is* responsive *if there are two votes r_1, r_2 such that everyone voting r_1 will produce a different winner than everyone voting r_2.)*

Proof: Let there be n voters total. Denote by v^i the vote vector where i voters vote r_1, and the remaining $n-i$ vote r_2. Because v^0 and v^n produce different winners under R, there must be some i such that v^i and v^{i+1} produce different winners. Thus, in v^i, all $n-i$ voters voting r_2 are pivotal, and in v^{i+1}, all $i+1$ voters voting r_1 are pivotal. Since $(n-i)+(i+1) > n$, at least one of $n-i$ and $i+1$ must be greater than $n/2$. ■

Rules that are hard to execute

The impossibility result only applies when an efficient algorithm is available for executing the rule, because algorithm Find-Two-Winners makes calls to such an algorithm as a subroutine. Thus, one possible way around the impossibility is to use a rule that is hard to execute. Indeed, as we pointed out before, a number of voting rules have been shown to be NP-hard to execute [Bartholdi *et al.*, 1989b; Hemaspaandra *et al.*, 1997; Cohen *et al.*, 1999; Dwork *et al.*, 2001; Ailon *et al.*, 2005]. Of course, we do actually need an algorithm for executing the rule to determine the winner of the election; and, although we cannot expect this to be a worst-case polynomial-time algorithm, it should at least run reasonably fast in practice for the rule to be practical. But if the algorithm does run fast in practice, then it can also be used by the manipulators as the subroutine in Find-Two-Winners. Therefore, this approach does not look very promising.

8.5 Summary

In this chapter, we studied mechanism design for bounded agents. Specifically, we looked at how hard it is computationally for agents to find a best response to given opponent strategies in various expressive preference aggregation settings.

In Section 8.1, we showed that there are settings where using the optimal (social-welfare maximizing) truthful mechanism requires the center to solve an NP-hard computational problem; but there is another, non-truthful mechanism that can be executed in polynomial time, and under which the problem of finding a beneficial manipulation is hard for one of the agents. Moreover, if the agent manages to find the manipulation, the produced outcome is the same as that of the best truthful mechanism; and if the agent does not manage to find it, the produced outcome is strictly *better*.

In Section 8.2, we showed how to *tweak* existing voting rules to make manipulation hard, while leaving much of the original nature of the rule intact. The tweak studied in this section consists of adding one preround to the election, where candidates face each other one against one. The surviving candidates continue to the original protocol. Surprisingly, this simple and universal tweak makes typical rules hard to manipulate! The resulting protocols are NP-hard, #P-hard, or PSPACE-hard to manipulate, depending on whether the schedule of the preround is determined before the votes are collected, after the votes are collected, or the scheduling and the vote collecting are interleaved, respectively. We proved general sufficient conditions on the rules for this tweak to introduce the hardness, and showed that the most common voting rules satisfy those conditions. These are the first results in voting settings where manipulation is in a higher complexity class than NP (presuming PSPACE $\neq$ NP).

In Section 8.3, we noted that all of the previous results on hardness of manipulation in elections required the number of candidates to be unbounded. Such hardness results lose relevance when the number of candidates is small, because manipulation algorithms that are exponential only in the number of candidates (and only slightly so) might be available. We gave such an algorithm for an individual agent to manipulate the Single Transferable Vote (STV) rule, which had been shown hard to manipulate in the above sense. To obtain hardness-of-manipulation results in settings where the number of candidates is a small constant, we studied *coalitional* manipulation by *weighted* voters. (We show that for simpler manipulation problems, manipulation cannot be hard with few candidates.) We studied both *constructive* manipulation (making a given candidate win) and *destructive* manipulation (making a given candidate not win). The following tables summarize our results.

Number of candidates	2	3	4,5,6	≥ 7
Borda	P	NP-complete	NP-complete	NP-complete
veto	P	NP-complete	NP-complete	NP-complete
STV	P	NP-complete	NP-complete	NP-complete
plurality with runoff	P	NP-complete	NP-complete	NP-complete
Copeland	P	P	NP-complete	NP-complete
maximin	P	P	NP-complete	NP-complete
randomized cup	P	P	P	NP-complete
regular cup	P	P	P	P
plurality	P	P	P	P

Complexity of CONSTRUCTIVE CW-MANIPULATION

Number of candidates	2	≥ 3
STV	P	NP-complete
plurality with runoff	P	NP-complete
Borda	P	P
veto	P	P
Copeland	P	P
maximin	P	P
regular cup	P	P
plurality	P	P

Complexity of DESTRUCTIVE CW-MANIPULATION

We also showed that hardness of manipulation in this setting implies hardness of manipulation by an individual in unweighted settings when there is uncertainty about the others' votes.

All of the hardness results mentioned above only show hardness in the worst case; they do not preclude the existence of an efficient algorithm that *often* finds a successful manipulation (when it exists). There have been attempts to design a rule under which finding a beneficial manipulation is *usually* hard, but they have failed. To explain this failure, in Section 8.4, we showed that it is in fact impossible to design such a rule, if the rule is also required to satisfy another property: a large fraction of the manipulable instances are both weakly monotone, and allow the manipulators to make either of exactly two candidates win. We argued why one should expect voting rules to have this property, and showed experimentally that common voting rules satisfy it. We also discussed approaches for potentially circumventing this impossibility result, some of which appear worthwhile to investigate in future research.

The manipulation problems defined in this chapter did not involve sophisticated strategic reasoning: we simply assumed that the manipulator(s) knew the others' votes (or at least a distribution over them). Acting in a strategically optimal way becomes more difficult when this information is not available, and the manipulator(s) must reason over how the others are likely to act. This is the topic of the next chapter.

Chapter 9

Computing Game-Theoretic Solutions

In the previous chapter, we saw that in certain settings, it is computationally hard for an agent to act in a strategically optimal way even when the agent already knows the actions of all the other agents—that is, even the *best-response* problem is hard. There are also many settings in which this is not the case, *i.e.* in which it is computationally easy to find a best response to specific actions of the other agents. However, there is much more to optimal strategic behavior than merely best-responding to given actions of the other agents. In most settings, the agents (1) do not know each other's types, but rather only a distribution over them; and (2) must somehow deduce the other agents' strategies, which map types to actions, using game-theoretic analysis (such as equilibrium reasoning). To some extent, we already discussed (1) in the previous chapter: we saw that uncertainty about other agents' votes makes the manipulation problem more difficult. But those results assumed that the manipulator(s) somehow knew a probability distribution over the other agents' votes. This is sweeping (2) under the rug, because to obtain such a distribution, some strategic assessment needs to be made about the strategies that the other agents are likely to use. The traditional assumption in mechanism design has been that strategic agents will play according to some solution concept (such as Bayes-Nash equilibrium), and if this assumption is accurate, then by the revelation principle, we can restrict our attention to truthful mechanisms. But what if such solutions are too hard for the agents to compute? If that is the case, then the agents cannot play according to these solutions, and the revelation principle loses its relevance.[1] Thus, the complexity of computing solutions according to these concepts becomes a key issue when considering how to design mechanisms for bounded agents. This chapter investigates that issue. (An even more difficult question is how a mechanism designer should proceed when solutions do turn out to be hard to compute, but that

[1]It should be emphasized here that it only loses its relevance in the sense that we may be able to achieve better results with non-truthful mechanisms (due to the agents' computational boundedness). However, the revelation principle still holds in the sense that using truthful mechanisms, we can achieve any result that we would have achieved under a non-truthful mechanism if agents *had* acted according to game-theoretic solution concepts (even if this would have been computationally infeasible for them). Thus, if we want to take the perspective that we want to help the agents act strategically optimally, and that we do not want them to feel any regret about having failed to misreport their preferences in the optimal way, then the revelation principle still applies and we may as well restrict our attention to truthful mechanisms—thereby relieving the agents of the burden of acting strategically. The view taken in this dissertation, however, is that there is nothing bad about an agent failing to manipulate the mechanism if the overall outcome is better as a result. This view is what motivates the interest in non-truthful mechanisms in this chapter.

remains outside the scope of this chapter.)

Certainly, intuitively, it seems that computing an equilibrium is typically much harder than computing a best response, and we have already seen in the previous chapter that the latter is hard in some complex settings. This chapter will therefore focus on more basic settings: in fact, it will focus only on normal-form games and Bayesian games that are *flatly represented* (all types and actions are listed explicitly). We note that even in (say) a straightforward voting setting, the type and action spaces are exponential in size, so that flat representation is not reasonable. (In fact, this is why finding a best response can be computationally hard in those settings. By contrast, computing best responses in flatly represented games is easy.) If computing game-theoretic solutions is hard even under flat representation, this makes it seem even more unlikely that agents will be able to compute such solutions in the richer settings that we are interested in.

If we are in fact able to compute certain game-theoretic solutions, that is of interest for other reasons as well. It can be helpful in predicting the outcomes of non-truthful mechanisms. It also allows us to build computer players for game-theoretically nontrivial games such as poker [Koller and Pfeffer, 1997; Shi and Littman, 2001; Billings *et al.*, 2003; Gilpin and Sandholm, 2006b,a] or potentially even RoboSoccer. Finally, it can also potentially be helpful in other settings where computer systems are interacting with other agents (human or computer) whose interests are not aligned with the computer system, such as surveillance and fraud detection.

The rest of this chapter is layed out as follows. In Section 9.1, we characterize the complexity of some basic computational questions about dominance and iterated dominance in both normal-form and Bayesian games [Conitzer and Sandholm, 2005c], and in Section 9.2 we do the same for Nash equilibrium [Conitzer and Sandholm, 2003e]. In Section 9.3 we provide a parameterized definition of strategy eliminability that is more general than dominance, and give an algorithm for computing whether a strategy is eliminable whose running time is exponential in only one parameter of the definition [Conitzer and Sandholm, 2005e].

9.1 Dominance and iterated dominance

While an ever-increasing amount of research focuses on computing Nash equilibria, the arguably simpler concept of (iterated) dominance has received much less attention. After an early short paper on a special case [Knuth *et al.*, 1988], the main computational study of these concepts has taken place in a paper in the game theory community [Gilboa *et al.*, 1993].[2] Computing solutions according to (iterated) dominance is important for at least the following reasons: 1) it can be computationally easier than computing (for instance) a Nash equilibrium (and therefore it can be useful as a preprocessing step in computing a Nash equilibrium), and 2) (iterated) dominance requires a weaker rationality assumption on the players than (for instance) Nash equilibrium, and therefore solutions derived according to it are more likely to occur.

In this section, we study some fundamental computational questions concerning dominance and iterated dominance, including how hard it is to check whether a given strategy can be eliminated by each of the variants of these notions. We study both strict and weak dominance, by both pure and mixed strategies, in both normal-form and Bayesian games.

[2] This is not to say that computer scientists have ignored dominance altogether. For example, simple dominance checks are sometimes used as a subroutine in searching for Nash equilibria [Porter *et al.*, 2004].

9.1.1 Dominance (not iterated)

In this subsection, we study the notion of one-shot (not iterated) dominance. When we are looking at the dominance relations for player i, the other players $(-i)$ can be thought of as a single player.[3] Therefore, in the rest of this section, when we study one-shot (not iterated) dominance, we will focus without loss of generality on two-player games.[4] In two-player games, we will generally refer to the players as r (row) and c (column) rather than 1 and 2.

As a first observation, checking whether a given strategy is strictly (weakly) dominated by some *pure* strategy is straightforward, by checking, for every pure strategy for that player, whether the latter strategy performs strictly better against all the opponent's pure strategies (at least as well against all the opponent's pure strategies, and strictly better against at least one).[5] Next, we show that checking whether a given strategy is dominated by some *mixed* strategy can be done in polynomial time by solving a single linear program. (Similar linear programs have been given before [Myerson, 1991]; we present the result here for completeness, and because we will build on the linear programs given below in Theorem 98.)

Proposition 11 *Given the row player's utilities, a subset D_r of the row player's pure strategies Σ_r, and a distinguished strategy σ_r^* for the row player, we can check in time polynomial in the size of the game (by solving a single linear program of polynomial size) whether there exists some mixed strategy σ_r, that places positive probability only on strategies in D_r and dominates σ_r^*, both for strict and for weak dominance.*

Proof: Let p_{d_r} be the probability that σ_r places on $d_r \in D_r$. We will solve a single linear program in each of our algorithms; linear programs can be solved in polynomial time [Khachiyan, 1979]. For strict dominance, the question is whether the p_{d_r} can be set so that for every pure strategy for the column player $\sigma_c \in \Sigma_c$, $\sum\limits_{d_r \in D_r} p_{d_r} u_r(d_r, \sigma_c) > u_r(\sigma_r^*, \sigma_c)$. Because the inequality must be strict, we cannot solve this directly by linear programming. We proceed as follows. Because the game is finite, we may assume without loss of generality that all utilities are positive (if not, simply add a constant to all utilities.) Solve the following linear program:

minimize $\sum\limits_{d_r \in D_r} p_{d_r}$
such that
for all $\sigma_c \in \Sigma_c$, $\sum\limits_{d_r \in D_r} p_{d_r} u_r(d_r, \sigma_c) \geq u_r(\sigma_r^*, \sigma_c)$.

If σ_r^* is strictly dominated by some mixed strategy, this linear program has a solution with objective value < 1. (The dominating strategy is a feasible solution with objective value exactly 1. Because no constraint is binding for this solution, we can reduce one of the probabilities slightly

[3] This player may have a very large strategy space (one pure strategy for every vector of pure strategies for the players that are being replaced). Nevertheless, this will not result in an increase in our representation size, because the original representation already had to specify utilities for each of these vectors.

[4] We note that a restriction to two-player games would not be without loss of generality for *iterated* dominance. This is because for iterated dominance, we need to look at the dominated strategies of each individual player, so we cannot merge any players.

[5] Recall that the assumption of a single opponent (that is, the assumption of two players) is without loss of generality for one-shot dominance.

without affecting feasibility, thereby obtaining a solution with objective value < 1.) Moreover, if this linear program has a solution with objective value < 1, there is a mixed strategy strictly dominating σ_r^*, which can be obtained by taking the LP solution and adding the remaining probability to any strategy (because all the utilities are positive, this will add to the left side of any inequality, so all inequalities will become strict). Thus, we have strict dominance if and only if the linear program has a solution with objective value < 1.

For weak dominance, we can solve the following linear program:

maximize $\sum_{\sigma_c \in \Sigma_c} ((\sum_{d_r \in D_r} p_{d_r} u_r(d_r, \sigma_c)) - u_r(\sigma_r^*, \sigma_c))$

such that

for all $\sigma_c \in \Sigma_c$, $\sum_{d_r \in D_r} p_{d_r} u_r(d_r, \sigma_c) \geq u_r(\sigma_r^*, \sigma_c)$;

$\sum_{d_r \in D_r} p_{d_r} = 1$.

If σ_r^* is weakly dominated by some mixed strategy, then that mixed strategy is a feasible solution to this program with objective value > 0, because for at least one strategy $\sigma_c \in \Sigma_c$ we have $(\sum_{d_r \in D_r} p_{d_r} u_r(d_r, \sigma_c)) - u_r(\sigma_r^*, \sigma_c) > 0$. On the other hand, if this program has a solution with objective value > 0, then for at least one strategy $\sigma_c \in \Sigma_c$ we must have $(\sum_{d_r \in D_r} p_{d_r} u_r(d_r, \sigma_c)) - u_r(\sigma_r^*, \sigma_c) > 0$, and thus the linear program's solution is a weakly dominating mixed strategy. ■

9.1.2 Iterated dominance

We now move on to iterated dominance. It is well-known that iterated strict dominance is path-independent [Gilboa *et al.*, 1990; Osborne and Rubinstein, 1994]—that is, if we remove dominated strategies until no more dominated strategies remain, in the end the remaining strategies for each player will be the same, regardless of the order in which strategies are removed. Because of this, to see whether a given strategy can be eliminated by iterated strict dominance, all that needs to be done is to repeatedly remove strategies that are strictly dominated, until no more dominated strategies remain. Because we can check in polynomial time whether any given strategy is dominated (whether or not dominance by mixed strategies is allowed, as described in Subsection 9.1.1), this whole procedure takes only polynomial time. In the case of iterated dominance by pure strategies with two players, Knuth *et al.* [1988] slightly improve on (speed up) the straightforward implementation of this procedure by keeping track of, for each ordered pair of strategies for a player, the number of opponent strategies that prevent the first strategy from dominating the second. Hereby the runtime for an $m \times n$ game is reduced from $O((m+n)^4)$ to $O((m+n)^3)$. (Actually, they only study very weak dominance (for which no strict inequalities are required), but the approach is easily extended.)

In contrast, iterated weak dominance is known to be path-dependent.[6] For example, in the following game, using iterated weak dominance we can eliminate M first, and then D, or R first, and then U.

[6] There is, however, a restriction of weak dominance called *nice weak dominance* which is path-independent [Marx and Swinkels, 1997, 2000]. For an overview of path-independence results, see Apt [2004].

	L	M	R
U	$1, 1$	$0, 0$	$1, 0$
D	$1, 1$	$1, 0$	$0, 0$

Therefore, while the procedure of removing weakly dominated strategies until no more weakly dominated strategies remain can certainly be executed in polynomial time, which strategies survive in the end depends on the order in which we remove the dominated strategies. We will investigate two questions for iterated weak dominance: whether a given strategy is eliminated in some path, and whether there is a path to a unique solution (one pure strategy left per player). We will show that both of these problems are computationally hard.

Definition 52 *Given a game in normal form and a distinguished strategy σ^*, IWD-STRATEGY-ELIMINATION asks whether there is some path of iterated weak dominance that eliminates σ^*. Given a game in normal form, IWD-UNIQUE-SOLUTION asks whether there is some path of iterated weak dominance that leads to a unique solution (one strategy left per player).*

The following lemma shows a special case of normal-form games in which allowing for weak dominance by mixed strategies (in addition to weak dominance by pure strategies) does not help. We will prove the hardness results in this setting, so that they will hold whether or not dominance by mixed strategies is allowed.

Lemma 21 *Suppose that all the utilities in a game are in $\{0, 1\}$. Then every pure strategy that is weakly dominated by a mixed strategy is also weakly dominated by a pure strategy.*

Proof: Suppose pure strategy σ is weakly dominated by mixed strategy σ^*. If σ gets a utility of 1 against some opponent strategy (or vector of opponent strategies if there are more than 2 players), then all the pure strategies that σ^* places positive probability on must also get a utility of 1 against that opponent strategy (or else the expected utility would be smaller than 1). Moreover, at least one of the pure strategies that σ^* places positive probability on must get a utility of 1 against an opponent strategy that σ gets 0 against (or else the inequality would never be strict). It follows that this pure strategy weakly dominates σ. ■

We are now ready to prove the main results of this subsection.

Theorem 93 *IWD-STRATEGY-ELIMINATION is NP-complete, even with 2 players, and with 0 and 1 being the only utilities occurring in the matrix—whether or not dominance by mixed strategies is allowed.*

Proof: The problem is in NP because given a sequence of strategies to be eliminated, we can easily check whether this is a valid sequence of eliminations (even when dominance by mixed strategies is allowed, using Proposition 11). To show that the problem is NP-hard, we reduce an arbitrary satisfiability instance (given by a nonempty set of clauses C over a nonempty set of variables V, with corresponding literals $L = \{+v : v \in V\} \cup \{-v : v \in V\}$) to the following IWD-STRATEGY-ELIMINATION instance. (In this instance, we will specify that certain strategies are *uneliminable*.

A strategy σ_r can be made uneliminable, even when 0 and 1 are the only allowed utilities, by adding another strategy σ_r' and another opponent strategy σ_c, so that: 1. σ_r and σ_r' are the only strategies that give the row player a utility of 1 against σ_c. 2. σ_r and σ_r' always give the row player the same utility. 3. σ_c is the only strategy that gives the column player a utility of 1 against σ_r', but otherwise σ_c always gives the column player utility 0. This makes it impossible to eliminate any of these three strategies. We will not explicitly specify the additional strategies to make the proof more legible.)

In this proof, we will denote row player strategies by s, and column player strategies by t, to improve legibility. Let the row player's pure strategy set be given as follows. For every variable $v \in V$, the row player has corresponding strategies $s^1_{+v}, s^2_{+v}, s^1_{-v}, s^2_{-v}$. Additionally, the row player has the following 2 strategies: s^1_0 and s^2_0, where $s^2_0 = \sigma^*_r$ (that is, it is the strategy we seek to eliminate). Finally, for every clause $c \in C$, the row player has corresponding strategies s^1_c (uneliminable) and s^2_c. Let the column player's pure strategy set be given as follows. For every variable $v \in V$, the column player has a corresponding strategy t_v. For every clause $c \in C$, the column player has a corresponding strategy t_c, and additionally, for every literal $l \in L$ that occurs in c, a strategy $t_{c,l}$. For every variable $v \in V$, the column player has corresponding strategies t_{+v}, t_{-v} (both uneliminable). Finally, the column player has three additional strategies: t^1_0 (uneliminable), t^2_0, and t_1.

The utility function for the row player is given as follows:

- $u_r(s^1_{+v}, t_v) = 0$ for all $v \in V$;
- $u_r(s^2_{+v}, t_v) = 1$ for all $v \in V$;
- $u_r(s^1_{-v}, t_v) = 1$ for all $v \in V$;
- $u_r(s^2_{-v}, t_v) = 0$ for all $v \in V$;
- $u_r(s^1_{+v}, t_1) = 1$ for all $v \in V$;
- $u_r(s^2_{+v}, t_1) = 0$ for all $v \in V$;
- $u_r(s^1_{-v}, t_1) = 0$ for all $v \in V$;
- $u_r(s^2_{-v}, t_1) = 1$ for all $v \in V$;
- $u_r(s^b_{+v}, t_{+v}) = 1$ for all $v \in V$ and $b \in \{1, 2\}$;
- $u_r(s^b_{-v}, t_{-v}) = 1$ for all $v \in V$ and $b \in \{1, 2\}$;
- $u_r(s_l, t) = 0$ otherwise for all $l \in L$ and $t \in S_2$;
- $u_r(s^1_0, t_c) = 0$ for all $c \in C$;
- $u_r(s^2_0, t_c) = 1$ for all $c \in C$;
- $u_r(s^b_0, t^1_0) = 1$ for all $b \in \{1, 2\}$;
- $u_r(s^1_0, t^2_0) = 1$;
- $u_r(s^2_0, t^2_0) = 0$;
- $u_r(s^b_0, t) = 0$ otherwise for all $b \in \{1, 2\}$ and $t \in S_2$;
- $u_r(s^b_c, t) = 0$ otherwise for all $c \in C$ and $b \in \{1, 2\}$;

and the row player's utility is 0 in every other case. The utility function for the column player is given as follows:

- $u_c(s, t_v) = 0$ for all $v \in V$ and $s \in S_1$;
- $u_c(s, t_1) = 0$ for all $s \in S_1$;
- $u_c(s_l^2, t_c) = 1$ for all $c \in C$ and $l \in L$ where $l \in c$ (literal l occurs in clause c);
- $u_c(s_{l_2}^2, t_{c,l_1}) = 1$ for all $c \in C$ and $l_1, l_2 \in L, l_1 \neq l_2$ where $l_2 \in c$;
- $u_c(s_c^1, t_c) = 1$ for all $c \in C$;
- $u_c(s_c^2, t_c) = 0$ for all $c \in C$;
- $u_c(s_c^b, t_{c,l}) = 1$ for all $c \in C$, $l \in L$, and $b \in \{1, 2\}$;
- $u_c(s_2, t_c) = u_c(s_2, t_{c,l}) = 0$ otherwise for all $c \in C$ and $l \in L$;

and the column player's utility is 0 in every other case. We now show that the two instances are equivalent.

First, suppose there is a solution to the satisfiability instance: that is, a truth-value assignment to the variables in V such that all clauses are satisfied. Then, consider the following sequence of eliminations in our game: 1. For every variable v that is set to *true* in the assignment, eliminate t_v (which gives the column player utility 0 everywhere). 2. Then, for every variable v that is set to *true* in the assignment, eliminate s_{+v}^2 using s_{+v}^1 (which is possible because t_v has been eliminated, and because t_1 has not been eliminated (yet)). 3. Now eliminate t_1 (which gives the column player utility 0 everywhere). 4. Next, for every variable v that is set to *false* in the assignment, eliminate s_{-v}^2 using s_{-v}^1 (which is possible because t_1 has been eliminated, and because t_v has not been eliminated (yet)). 5. For every clause c which has the variable corresponding to one of its positive literals $l = +v$ set to *true* in the assignment, eliminate t_c using $t_{c,l}$ (which is possible because s_l^2 has been eliminated, and s_c^2 has not been eliminated (yet)). 6. For every clause c which has the variable corresponding to one of its negative literals $l = -v$ set to *false* in the assignment, eliminate t_c using $t_{c,l}$ (which is possible because s_l^2 has been eliminated, and s_c^2 has not been eliminated (yet)). 7. Because the assignment satisfied the formula, all the t_c have now been eliminated. Thus, we can eliminate $s_0^2 = \sigma_r^*$ using s_0^1. It follows that there is a solution to the IWD-STRATEGY-ELIMINATION instance.

Now suppose there is a solution to the IWD-STRATEGY-ELIMINATION instance. By Lemma 21, we can assume that all the dominances are by pure strategies. We first observe that only s_0^1 can eliminate $s_0^2 = \sigma_r^*$, because it is the only other strategy that gets the row player a utility of 1 against t_0^1, and t_0^1 is uneliminable. However, because s_0^2 performs better than s_0^1 against the t_c strategies, it follows that all of the t_c strategies must be eliminated. For each $c \in C$, the strategy t_c can only be eliminated by one of the strategies $t_{c,l}$ (with the same c), because these are the only other strategies that get the column player a utility of 1 against s_c^1, and s_c^1 is uneliminable. But, in order for some $t_{c,l}$ to eliminate t_c, s_l^2 must be eliminated first. Only s_l^1 can eliminate s_l^2, because it is the only other strategy that gets the row player a utility of 1 against t_l, and t_l is uneliminable. We next show that for every $v \in V$ only one of s_{+v}^2, s_{-v}^2 can be eliminated. This is because in order for s_{+v}^1 to eliminate s_{+v}^2, t_v needs to have been eliminated and t_1, not (so t_v must be eliminated before t_1); but in order for s_{-v}^1 to eliminate s_{-v}^2, t_1 needs to have been eliminated and t_v, not (so t_1 must be eliminated before t_v). So, set v to *true* if s_{+v}^2 is eliminated, and to *false* otherwise Because by the above, for every clause c, one of the s_l^2 with $l \in c$ must be eliminated, it follows that this is a satisfying assignment to the satisfiability instance. ■

Using Theorem 93, it is now (relatively) easy to show that IWD-UNIQUE-SOLUTION is also NP-complete under the same restrictions.

Theorem 94 *IWD-UNIQUE-SOLUTION is NP-complete, even with 2 players, and with 0 and 1 being the only utilities occurring in the matrix—whether or not dominance by mixed strategies is allowed.*

Proof: Again, the problem is in NP because we can nondeterministically choose the sequence of eliminations and verify whether it is correct. To show NP-hardness, we reduce an arbitrary IWD-STRATEGY-ELIMINATION instance to the following IWD-UNIQUE-SOLUTION instance. Let all the strategies for each player from the original instance remain part of the new instance, and let the utilities resulting from the players playing a pair of these strategies be the same. We add three additional strategies $\sigma_r^1, \sigma_r^2, \sigma_r^3$ for the row player, and three additional strategies $\sigma_c^1, \sigma_c^2, \sigma_c^3$ for the column player. Let the additional utilities be as follows:

- $u_r(\sigma_r, \sigma_c^j) = 1$ for all $\sigma_r \notin \{\sigma_r^1, \sigma_r^2, \sigma_r^3\}$ and $j \in \{2, 3\}$;
- $u_r(\sigma_r^i, \sigma_c) = 1$ for all $i \in \{1, 2, 3\}$ and $\sigma_c \notin \{\sigma_c^2, \sigma_c^3\}$;
- $u_r(\sigma_r^i, \sigma_c^2) = 1$ for all $i \in \{2, 3\}$;
- $u_r(\sigma_r^1, \sigma_c^3) = 1$;
- and the row player's utility is 0 in all other cases involving a new strategy.
- $u_c(\sigma_r^3, \sigma_c) = 1$ for all $\sigma_c \notin \{\sigma_c^1, \sigma_c^2, \sigma_c^3\}$;
- $u_c(\sigma_r^*, \sigma_c^j) = 1$ for all $j \in \{2, 3\}$ (σ_r^* is the strategy to be eliminated in the original instance);
- $u_c(\sigma_r^i, \sigma_c^1) = 1$ for all $i \in \{1, 2\}$;
- $u_r(\sigma_r^1, \sigma_c^2) = 1$;
- $u_r(\sigma_r^2, \sigma_c^3) = 1$;
- and the column player's utility is 0 in all other cases involving a new strategy.

We proceed to show that the two instances are equivalent.

First suppose there exists a solution to the original IWD-STRATEGY-ELIMINATION instance. Then, perform the same sequence of eliminations to eliminate σ_r^* in the new IWD-UNIQUE-SOLUTION instance. (This is possible because at any stage, any weak dominance for the row player in the original instance is still a weak dominance in the new instance, because the two strategies' utilities for the row player are the same when the column player plays one of the new strategies; and the same is true for the column player.) Once σ_r^* is eliminated, let σ_c^1 eliminate σ_c^2. (It performs better against σ_r^2.) Then, let σ_r^1 eliminate all the other remaining strategies for the row player. (It always performs better against either σ_c^1 or σ_c^3.) Finally, σ_c^1 is the unique best response against σ_r^1 among the column player's remaining strategies, so let it eliminate all the other remaining strategies for the column player. Thus, there exists a solution to the IWD-UNIQUE-SOLUTION instance.

Now suppose there exists a solution to the IWD-UNIQUE-SOLUTION instance. By Lemma 21, we can assume that all the dominances are by pure strategies. We will show that none of the new strategies $(\sigma_r^1, \sigma_r^2, \sigma_r^3, \sigma_c^1, \sigma_c^2, \sigma_c^3)$ can either eliminate another strategy, or be eliminated before σ_r^* is eliminated. Thus, there must be a sequence of eliminations ending in the elimination of σ_r^*, which does not involve any of the new strategies, and is therefore a valid sequence of eliminations in the original game (because all original strategies perform the same against each new strategy). We now show that this is true by exhausting all possibilities for the *first* elimination before σ_r^* is eliminated that involves a new strategy. None of the σ_r^i can be eliminated by a $\sigma_r \notin \{\sigma_r^1, \sigma_r^2, \sigma_r^3\}$, because the σ_r^i perform better against σ_c^1. σ_r^1 cannot eliminate any other strategy, because it always performs poorer against σ_c^2. σ_r^2 and σ_r^3 are equivalent from the row player's perspective (and thus cannot eliminate each other), and cannot eliminate any other strategy because they always perform poorer against σ_c^3. None of the σ_c^j can be eliminated by a $\sigma_c \notin \{\sigma_c^1, \sigma_c^2, \sigma_c^3\}$, because the σ_c^j always perform better against either σ_r^1 or σ_r^2. σ_c^1 cannot eliminate any other strategy, because it always performs poorer against either σ_r^* or σ_r^3. σ_c^2 cannot eliminate any other strategy, because it always performs poorer against σ_r^2 or σ_r^3. σ_c^3 cannot eliminate any other strategy, because it always performs poorer against σ_r^1 or σ_r^3. From this, it follows that there exists a solution to the IWD-STRATEGY-ELIMINATION instance. ■

A slightly weaker version of the part of Theorem 94 concerning dominance by pure strategies only is the main result of Gilboa *et al.* [1993]. (Besides not proving the result for dominance by mixed strategies, the original result was weaker because it required utilities $\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ rather than just $\{0, 1\}$ (and because of this, our Lemma 21 cannot be applied to the original result to get the result for mixed strategies, giving us an additional motivation to prove the result for the case where utilities are in $\{0, 1\}$).)

9.1.3 (Iterated) dominance using mixed strategies with small supports

When showing that a strategy is dominated by a mixed strategy, there are several reasons to prefer exhibiting a dominating strategy that places positive probability on as few pure strategies as possible. First, this will reduce the number of bits required to specify the dominating strategy (and thus the proof of dominance can be communicated quicker): if the dominating mixed strategy places positive probability on only k strategies, then it can be specified using k real numbers for the probabilities, plus $k \log m$ (where m is the number of strategies for the player under consideration) bits to indicate which strategies are used. Second, the proof of dominance will be "cleaner": for a dominating mixed strategy, it is typically (always in the case of strict dominance) possible to spread some of the probability onto any unused pure strategy and still have a dominating strategy, but this obscures which pure strategies are the ones that are key in making the mixed strategy dominating. Third, because (by the previous) the argument for eliminating the dominated strategy is simpler and easier to understand, it is more likely to be accepted. Fourth, the level of risk neutrality required for the argument to work is reduced, at least in the extreme case where dominance by a single pure strategy can be exhibited (no risk neutrality is required here).

This motivates the following problem.

Definition 53 (MINIMUM-DOMINATING-SET) *We are given the row player's utilities of a game in normal form, a distinguished strategy σ^* for the row player, a specification of whether the dominance should be strict or weak, and a number k (not necessarily a constant). We are asked* *whether there exists a mixed strategy σ for the row player that places positive probability on at most k pure strategies, and dominates σ^* in the required sense.*

Unfortunately, this problem is NP-complete.

Theorem 95 *MINIMUM-DOMINATING-SET is NP-complete, both for strict and for weak dominance.*

Proof: The problem is in NP because we can nondeterministically choose a set of at most k strategies to give positive probability, and decide whether we can dominate σ^* with these k strategies as described in Proposition 11. To show NP-hardness, we reduce an arbitrary SET-COVER instance (given a set S, subsets $S_1, S_2, \ldots, S_r$, and a number t, can all of S be covered by at most t of the subsets?) to the following MINIMUM-DOMINATING-SET instance. For every element $s \in S$, there is a pure strategy σ_s for the column player. For every subset S_i, there is a pure strategy σ_{S_i} for the row player. Finally, there is the distinguished pure strategy σ^* for the row player. The row player's utilities are as follows: $u_r(\sigma_{S_i}, \sigma_s) = t + 1$ if $s \in S_i$; $u_r(\sigma_{S_i}, \sigma_s) = 0$ if $s \notin S_i$; $u_r(\sigma^*, \sigma_s) = 1$ for all $s \in S$. Finally, we let $k = t$. We now proceed to show that the two instances are equivalent.

First suppose there exists a solution to the SET-COVER instance. Without loss of generality, we can assume that there are exactly k subsets in the cover. Then, for every S_i that is in the cover, let the dominating strategy σ place exactly $\frac{1}{k}$ probability on the corresponding pure strategy σ_{S_i}. Now, if we let $n(s)$ be the number of subsets in the cover containing s (we observe that that $n(s) \geq 1$), then for every strategy σ_s for the column player, the row player's expected utility for playing σ when the column player is playing σ_s is $u(\sigma, \sigma_s) = \frac{n(s)}{k}(k+1) \geq \frac{k+1}{k} > 1 = u(\sigma^*, \sigma_s)$. So σ strictly (and thus also weakly) dominates σ^*, and there exists a solution to the MINIMUM-DOMINATING-SET instance.

Now suppose there exists a solution to the MINIMUM-DOMINATING-SET instance. Consider the (at most k) pure strategies of the form σ_{S_i} on which the dominating mixed strategy σ places positive probability, and let $\mathcal{T}$ be the collection of the corresponding subsets S_i. We claim that $\mathcal{T}$ is a cover. For suppose there is some $s \in S$ that is not in any of the subsets in $\mathcal{T}$. Then, if the column player plays σ_s, the row player (when playing σ) will always receive utility 0—as opposed to the utility of 1 the row player would receive for playing σ^*, contradicting the fact that σ dominates σ^* (whether this dominance is weak or strict). It follows that there exists a solution to the SET-COVER instance. ■

On the other hand, if we require that the dominating strategy only places positive probability on a *very* small number of pure strategies, then it once again becomes easy to check whether a strategy is dominated. Specifically, to find out whether player i's strategy σ^* is dominated by a strategy that places positive probability on only k pure strategies, we can simply check, for every subset of k of player i's pure strategies, whether there is a strategy that places positive probability only on these k strategies and dominates σ^*, using Proposition 11. This requires only $O(|\Sigma_i|^k)$ such checks. Thus, if k is a constant, this constitutes a polynomial-time algorithm.

A natural question to ask next is whether *iterated* strict dominance remains computationally easy when dominating strategies are required to place positive probability on at most k pure strategies, where k is a small constant. (We have already shown in Subsection 9.1.2 that iterated weak dominance is hard even when $k = 1$, that is, only dominance by pure strategies is allowed.) Of course, if iterated strict dominance were path-independent under this restriction, computational easiness would follow as it did in Subsection 9.1.2. However, it turns out that this is not the case.

Observation 1 *If we restrict the dominating strategies to place positive probability on at most two pure strategies, iterated strict dominance becomes path-dependent.*

Proof: Consider the following game:

7, 1	0, 0	0, 0
0, 0	7, 1	0, 0
3, 0	3, 0	0, 0
0, 0	0, 0	3, 1
1, 0	1, 0	1, 0

Let (i, j) denote the outcome in which the row player plays the ith row and the column player plays the jth column. Because $(1, 1)$, $(2, 2)$, and $(4, 3)$ are all Nash equilibria, none of the column player's pure strategies will ever be eliminated, and neither will rows 1, 2, and 4. We now observe that randomizing uniformly over rows 1 and 2 dominates row 3, and randomizing uniformly over rows 3 and 4 dominates row 5. However, if we eliminate row 3 first, it becomes impossible to dominate row 5 without randomizing over at least 3 pure strategies. ■

Indeed, iterated strict dominance turns out to be hard even when $k = 3$.

Theorem 96 *If we restrict the dominating strategies to place positive probability on at most three pure strategies, it becomes NP-complete to decide whether a given strategy can be eliminated using iterated strict dominance.*

Proof: The problem is in NP because given a sequence of strategies to be eliminated, we can check in polynomial time whether this is a valid sequence of eliminations (for any strategy to be eliminated, we can check, for every subset of three other strategies, whether there is a strategy placing positive probability on only these three strategies that dominates the strategy to be eliminated, using Proposition 11). To show that the problem is NP-hard, we reduce an arbitrary satisfiability instance (given by a nonempty set of clauses C over a nonempty set of variables V, with corresponding literals $L = \{+v : v \in V\} \cup \{-v : v \in V\}$) to the following two-player game.

For every variable $v \in V$, the row player has strategies $s_{+v}, s_{-v}, s_v^1, s_v^2, s_v^3, s_v^4$, and the column player has strategies $t_v^1, t_v^2, t_v^3, t_v^4$. For every clause $c \in C$, the row player has a strategy s_c, and the column player has a strategy t_c, as well as, for every literal l occurring in c, an additional strategy t_c^l. The row player has two additional strategies s_1 and s_2. (s_2 is the strategy that we are seeking to eliminate.) Finally, the column player has one additional strategy t_1.

The utility function for the row player is given as follows (where ϵ is some sufficiently small number):

- $u_r(s_{+v}, t_v^j) = 4$ if $j \in \{1, 2\}$, for all $v \in V$;

- $u_r(s_{+v}, t_v^j) = 1$ if $j \in \{3, 4\}$, for all $v \in V$;
- $u_r(s_{-v}, t_v^j) = 1$ if $j \in \{1, 2\}$, for all $v \in V$;
- $u_r(s_{-v}, t_v^j) = 4$ if $j \in \{3, 4\}$, for all $v \in V$;
- $u_r(s_{+v}, t) = u_r(s_{-v}, t) = 0$ for all $v \in V$ and $t \notin \{t_v^1, t_v^2, t_v^3, t_v^4\}$;
- $u_r(s_v^i, t_v^i) = 13$ for all $v \in V$ and $i \in \{1, 2, 3, 4\}$;
- $u_r(s_v^i, t) = \epsilon$ for all $v \in V$, $i \in \{1, 2, 3, 4\}$, and $t \neq t_v^i$;
- $u_r(s_c, t_c) = 2$ for all $c \in C$;
- $u_r(s_c, t) = 0$ for all $c \in C$ and $t \neq t_c$;
- $u_r(s_1, t_1) = 1 + \epsilon$;
- $u_r(s_1, t) = \epsilon$ for all $t \neq t_1$;
- $u_r(s_2, t_1) = 1$;
- $u_r(s_2, t_c) = 1$ for all $c \in C$;
- $u_r(s_2, t) = 0$ for all $t \notin \{t_1\} \cup \{t_c : c \in C\}$.

The utility function for the column player is given as follows:

- $u_c(s_v^i, t_v^i) = 1$ for all $v \in V$ and $i \in \{1, 2, 3, 4\}$;
- $u_c(s, t_v^i) = 0$ for all $v \in V$, $i \in \{1, 2, 3, 4\}$, and $s \neq s_v^i$;
- $u_c(s_c, t_c) = 1$ for all $c \in C$;
- $u_c(s_l, t_c) = 1$ for all $c \in C$ and $l \in L$ occurring in c;
- $u_c(s, t_c) = 0$ for all $c \in C$ and $s \notin \{s_c\} \cup \{s_l : l \in c\}$;
- $u_c(s_c, t_c^l) = 1 + \epsilon$ for all $c \in C$;
- $u_c(s_{l'}, t_c^l) = 1 + \epsilon$ for all $c \in C$ and $l' \neq l$ occurring in c;
- $u_c(s, t_c^l) = \epsilon$ for all $c \in C$ and $s \notin \{s_c\} \cup \{s_{l'} : l' \in c, l \neq l'\}$;
- $u_c(s_2, t_1) = 1$;
- $u_c(s, t_1) = 0$ for all $s \neq s_2$.

We now show that the two instances are equivalent. First, suppose that there is a solution to the satisfiability instance. Then, consider the following sequence of eliminations in our game: 1. For every variable v that is set to *true* in the satisfying assignment, eliminate s_{+v} with the mixed strategy σ_r that places probability $1/3$ on s_{-v}, probability $1/3$ on s_v^1, and probability $1/3$ on s_v^2. (The expected utility of playing σ_r against t_v^1 or t_v^2 is $14/3 > 4$; against t_v^3 or t_v^4, it is $4/3 > 1$; and against anything else it is $2\epsilon/3 > 0$. Hence the dominance is valid.) 2. Similarly, for every variable v that is set to *false* in the satisfying assignment, eliminate s_{-v} with the mixed strategy σ_r that places probability $1/3$ on s_{+v}, probability $1/3$ on s_v^3, and probability $1/3$ on s_v^4. (The expected utility of playing σ_r against t_v^1 or t_v^2 is $4/3 > 1$; against t_v^3 or t_v^4, it is $14/3 > 4$; and against anything else it is $2\epsilon/3 > 0$. Hence the dominance is valid.) 3. For every $c \in C$, eliminate t_c with any t_c^l for which l was set to *true* in the satisfying assignment. (This is a valid dominance because t_c^l performs better than t_c against any strategy other than s_l, and we eliminated s_l in step 1 or in step 2.) 4. Finally, eliminate s_2 with s_1. (This is a valid dominance because s_1 performs better than s_2 against any strategy other than those in $\{t_c : c \in C\}$, which we eliminated in step 3.) Hence, there is an elimination path that eliminates s_2.

Now, suppose that there is an elimination path that eliminates s_2. The strategy that eventually dominates s_2 must place most of its probability on s_1, because s_1 is the only other strategy that performs well against t_1, which cannot be eliminated before s_2. But, s_1 performs significantly worse than s_2 against any strategy t_c with $c \in C$, so it follows that all these strategies must be eliminated first. Each strategy t_c can only be eliminated by a strategy that places most of its weight on the corresponding strategies t_c^l with $l \in c$, because they are the only other strategies that perform well against s_c, which cannot be eliminated before t_c. But, each strategy t_c^l performs significantly worse than t_c against s_l, so it follows that for every clause c, for one of the literals l occurring in it, s_l must be eliminated first. Now, strategies of the form t_v^j will never be eliminated because they are the unique best responses to the corresponding strategies s_v^j (which are, in turn, the best responses to the corresponding t_v^j). As a result, if strategy s_{+v} (respectively, s_{-v}) is eliminated, then its opposite strategy s_{-v} (respectively, s_{+v}) can no longer be eliminated, for the following reason. There is no other pure strategy remaining that gets a significant utility against more than one of the strategies $t_v^1, t_v^2, t_v^3, t_v^4$, but s_{-v} (respectively, s_{+v}) gets significant utility against all 4, and therefore cannot be dominated by a mixed strategy placing positive probability on at most 3 strategies. It follows that for each $v \in V$, at most one of the strategies s_{+v}, s_{-v} is eliminated, in such a way that for every clause c, for one of the literals l occurring in it, s_l must be eliminated. But then setting all the literals l such that s_l is eliminated to *true* constitutes a solution to the satisfiability instance. ■

In the next subsection, we return to the setting where there is no restriction on the number of pure strategies on which a dominating mixed strategy can place positive probability.

9.1.4 (Iterated) dominance in Bayesian games

In this subsection, we study Bayesian games. Because Bayesian games have a representation that is exponentially more concise than their normal-form representation, questions that are easy for normal-form games can be hard for Bayesian games. In fact, it turns out that checking whether a strategy is dominated by a pure strategy is hard in Bayesian games.

Theorem 97 *In a Bayesian game, it is NP-complete to decide whether a given pure strategy $\sigma_r : \Theta_r \to A_r$ is dominated by some other pure strategy (both for strict and weak dominance), even when the row player's distribution over types is uniform.*

Proof: The problem is in NP because it is easy to verify whether a candidate dominating strategy is indeed a dominating strategy. To show that the problem is NP-hard, we reduce an arbitrary satisfiability instance (given by a set of clauses C using variables from V) to the following Bayesian game. Let the row player's action set be $A_r = \{t, f, 0\}$ and let the column player's action set be $A_c = \{a_c : c \in C\}$. Let the row player's type set be $\Theta_r = \{\theta_v : v \in V\}$, with a distribution π_r that is uniform. Let the row player's utility function be as follows:

- $u_r(\theta_v, 0, a_c) = 0$ for all $v \in V$ and $c \in C$;

- $u_r(\theta_v, b, a_c) = |V|$ for all $v \in V$, $c \in C$, and $b \in \{t, f\}$ such that setting v to b satisfies c;
- $u_r(\theta_v, b, a_c) = -1$ for all $v \in V$, $c \in C$, and $b \in \{t, f\}$ such that setting v to b does not satisfy c.

Let the pure strategy to be dominated be the one that plays 0 for every type. We show that the strategy is dominated by a pure strategy if and only if there is a solution to the satisfiability instance.

First, suppose there is a solution to the satisfiability instance. Then, let σ_r^d be given by: $\sigma_r^d(\theta_v) = t$ if v is set to *true* in the solution to the satisfiability instance, and $\sigma_r^d(\theta_v) = f$ otherwise. Then, against any action a_c by the column player, there is at least one type θ_v such that either $+v \in c$ and $\sigma_r^d(\theta_v) = t$, or $-v \in c$ and $\sigma_r^d(\theta_v) = f$. Thus, the row player's expected utility against action a_c is at least $\frac{|V|}{|V|} - \frac{|V|-1}{|V|} = \frac{1}{|V|} > 0$. So, σ_r^d is a dominating strategy.

Now, suppose there is a dominating pure strategy σ_r^d. This dominating strategy must play t or f for at least one type. Thus, against any a_c by the column player, there must at least be some type θ_v for which $u_r(\theta_v, \sigma_r^d(\theta_v), a_c) > 0$. That is, there must be at least one variable v such that setting v to $\sigma_r^d(\theta_v)$ satifies c. But then, setting each v to $\sigma_r^d(\theta_v)$ must satisfy all the clauses. So a satisfying assignment exists. ■

However, it turns out that we can modify the linear programs from Proposition 11 to obtain a polynomial time algorithm for checking whether a strategy is dominated by a *mixed* strategy in Bayesian games.

Theorem 98 *In a Bayesian game, it can be decided in polynomial time whether a given (possibly mixed) strategy σ_r is dominated by some other mixed strategy, using linear programming (both for strict and weak dominance).*

Proof: We can modify the linear programs presented in Proposition 11 as follows. For strict dominance, again assuming without loss of generality that all the utilities in the game are positive, use the following linear program (in which $p_r^{\sigma_r}(\theta_r, a_r)$ is the probability that σ_r, the strategy to be dominated, places on a_r for type θ_r):

minimize $\sum_{\theta_r \in \Theta_r} \sum_{a_r \in A_r} p_r(a_r)$

such that

for all $a_c \in A_c$, $\sum_{\theta_r \in \Theta_r} \sum_{a_r \in A_r} \pi(\theta_r) u_r(\theta_r, a_r, a_c) p_r(\theta_r, a_r) \geq \sum_{\theta_r \in \Theta_r} \sum_{a_r \in A_r} \pi(\theta_r) u_r(\theta_r, a_r, a_c) p_r^{\sigma_r}(\theta_r, a_r)$;

for all $\theta_r \in \Theta_r$, $\sum_{a_r \in A_r} p_r(\theta_r, a_r) \leq 1$.

Assuming that $\pi(\theta_r) > 0$ for all $\theta_r \in \Theta_r$, this program will return an objective value smaller than $|\Theta_r|$ if and only if σ_r is strictly dominated, by reasoning similar to that done in Proposition 11.

For weak dominance, use the following linear program:

maximize $\sum_{a_c \in A_c} (\sum_{\theta_r \in \Theta_r} \sum_{a_r \in A_r} \pi(\theta_r) u_r(\theta_r, a_r, a_c) p_r(\theta_r, a_r) - \sum_{\theta_r \in \Theta_r} \sum_{a_r \in A_r} \pi(\theta_r) u_r(\theta_r, a_r, a_c) p_r^{\sigma_r}(\theta_r, a_r))$

such that

for all $a_c \in A_c$, $\sum_{\theta_r \in \Theta_r} \sum_{a_r \in A_r} \pi(\theta_r) u_r(\theta_r, a_r, a_c) p_r(\theta_r, a_r) \geq \sum_{\theta_r \in \Theta_r} \sum_{a_r \in A_r} \pi(\theta_r) u_r(\theta_r, a_r, a_c) p_r^{\sigma_r}(\theta_r, a_r)$;

for all $\theta_r \in \Theta_r$, $\sum_{a_r \in A_r} p_r(\theta_r, a_r) = 1$.

This program will return an objective value greater than 0 if and only if σ_r is weakly dominated, by reasoning similar to that done in Proposition 11. ∎

We now turn to iterated dominance in Bayesian games. Naïvely, one might argue that iterated dominance in Bayesian games *always* requires an exponential number of steps when a significant fraction of the game's pure strategies can be eliminated, because there are exponentially many pure strategies. However, this is not a very strong argument because oftentimes we can eliminate exponentially many pure strategies *in one step*. For example, if for some type $\theta_r \in \Theta_r$ we have, for all $a_c \in A_c$, that $u(\theta_r, a_r^1, a_c) > u(\theta_r, a_r^2, a_c)$, then any pure strategy for the row player which plays action a_r^2 for type θ_r is dominated (by the strategy that plays action a_r^1 for type θ_r instead)—and there are exponentially many ($|A_r|^{|\Theta_r|-1}$) such strategies. It is therefore conceivable that we need only polynomially many eliminations of *collections* of a player's strategies. However, the following theorem shows that this is not the case, by giving an example where an exponential number of *iterations* (that is, alternations between the players in eliminating strategies) is required. (We emphasize that this is not a result about computational complexity.)

Theorem 99 *Even in symmetric 3-player Bayesian games, iterated dominance by pure strategies can require an exponential number of iterations (both for strict and weak dominance), even with only three actions per player.*

Proof: Let each player $i \in \{1, 2, 3\}$ have $n + 1$ types $\theta_i^1, \theta_i^2, \ldots, \theta_i^{n+1}$. Let each player i have 3 actions a_i, b_i, c_i, and let the utility function of each player be defined as follows. (In the below, $i+1$ and $i + 2$ are shorthand for $i + 1 (\bmod 3)$ and $i + 2 (\bmod 3)$ when used as player indices. Also, $-\infty$ can be replaced by a sufficiently negative number. Finally, δ and ϵ should be chosen to be very small (even compared to $2^{-(n+1)}$), and ϵ should be more than twice as large as δ.)

- $u_i(\theta_i^1; a_i, c_{i+1}, c_{i+2}) = -1$;
- $u_i(\theta_i^1; a_i, s_{i+1}, s_{i+2}) = 0$ for $s_{i+1} \neq c_{i+1}$ or $s_{i+2} \neq c_{i+2}$;
- $u_i(\theta_i^1; b_i, s_{i+1}, s_{i+2}) = -\epsilon$ for $s_{i+1} \neq a_{i+1}$ and $s_{i+2} \neq a_{i+2}$;
- $u_i(\theta_i^1; b_i, s_{i+1}, s_{i+2}) = -\infty$ for $s_{i+1} = a_{i+1}$ or $s_{i+2} = a_{i+2}$;
- $u_i(\theta_i^1; c_i, s_{i+1}, s_{i+2}) = -\infty$ for all s_{i+1}, s_{i+2};
- $u_i(\theta_i^j; a_i, s_{i+1}, s_{i+2}) = -\infty$ for all s_{i+1}, s_{i+2} when $j > 1$;
- $u_i(\theta_i^j; b_i, s_{i+1}, s_{i+2}) = -\epsilon$ for all s_{i+1}, s_{i+2} when $j > 1$;
- $u_i(\theta_i^j; c_i, s_{i+1}, c_{i+2}) = \delta - \epsilon - 1/2$ for all s_{i+1} when $j > 1$;
- $u_i(\theta_i^j; c_i, s_{i+1}, s_{i+2}) = \delta - \epsilon$ for all s_{i+1} and $s_{i+2} \neq c_{i+2}$ when $j > 1$.

Let the distribution over each player's types be given by $p(\theta_i^j) = 2^{-j}$ (with the exception that $p(\theta_i^2) = 2^{-2} + 2^{-(n+1)}$). We will be interested in eliminating strategies of the following form: play b_i for type θ_i^1, and play one of b_i or c_i otherwise. Because the utility function is the same for any type θ_i^j with $j > 1$, these strategies are effectively defined by the total probability that they place on c_i,[7] which is $t_i^2(2^{-2}+2^{-(n+1)})+\sum_{j=3}^{n+1} t_i^j 2^{-j}$ where $t_i^j=1$ if player i plays c_i for type θ_i^j, and 0 otherwise. This probability is different for any two different strategies of the given form, and we have exponentially many different strategies of the given form. For any probability q which can be expressed as $t_2(2^{-2}+2^{-(n+1)})+\sum_{j=3}^{n+1} t_j 2^{-j}$ (with all $t_j \in \{0,1\}$), let $\sigma_i(q)$ denote the (unique) strategy of the given form for player i which places a total probability of q on c_i. Any strategy that plays c_i for type θ_i^1 or a_i for some type θ_i^j with $j>1$ can immediately be eliminated. We will show that, after that, we must eliminate the strategies $\sigma_i(q)$ with high q first, slowly working down to those with lower q.

Claim 1: If $\sigma_{i+1}(q')$ and $\sigma_{i+2}(q')$ have not yet been eliminated, and $q<q'$, then $\sigma_i(q)$ cannot yet be eliminated. *Proof:* First, we show that no strategy $\sigma_i(q'')$ can eliminate $\sigma_i(q)$. Against $\sigma_{i+1}(q'''), \sigma_{i+2}(q''')$, the utility of playing $\sigma_i(p)$ is $-\epsilon+p\cdot\delta-p\cdot q'''/2$. Thus, when $q'''=0$, it is best to set p as high as possible (and we note that $\sigma_{i+1}(0)$ and $\sigma_{i+2}(0)$ have not been eliminated), but when $q'''>0$, it is best to set p as low as possible because $\delta<q'''/2$. Thus, whether $q>q''$ or $q<q''$, $\sigma_i(q)$ will always do strictly better than $\sigma_i(q'')$ against some remaining opponent strategies. Hence, no strategy $\sigma_i(q'')$ can eliminate $\sigma_i(q)$. The only other pure strategies that could dominate $\sigma_i(q)$ are strategies that play a_i for type θ_i^1, and b_i or c_i for all other types. Let us take such a strategy and suppose that it plays c with probability p. Against $\sigma_{i+1}(q'), \sigma_{i+2}(q')$ (which have not yet been eliminated), the utility of playing this strategy is $-(q')^2/2-\epsilon/2+p\cdot\delta-p\cdot q'/2$. On the other hand, playing $\sigma_i(q)$ gives $-\epsilon+q\cdot\delta-q\cdot q'/2$. Because $q'>q$, we have $-(q')^2/2<-q\cdot q'/2$, and because δ and ϵ are small, it follows that $\sigma_i(q)$ receives a higher utility. Therefore, no strategy dominates $\sigma_i(q)$, proving the claim.

Claim 2: If for all $q'>q$, $\sigma_{i+1}(q')$ and $\sigma_{i+2}(q')$ have been eliminated, then $\sigma_i(q)$ can be eliminated. *Proof:* Consider the strategy for player i that plays a_i for type θ_i^1, and b_i for all other types (call this strategy σ_i'); we claim σ_i' dominates $\sigma_i(q)$. First, if either of the other players k plays a_k for θ_k^1, then σ_i' performs better than $\sigma_i(q)$ (which receives $-\infty$ in some cases). Because the strategies for player k that play c_k for type θ_k^1, or a_k for some type θ_k^j with $j>1$, have already been eliminated, all that remains to check is that σ_i' performs better than $\sigma_i(q)$ whenever both of the other two players play strategies of the following form: play b_k for type θ_k^1, and play one of b_k or c_k otherwise. We note that among these strategies, there are none left that place probability greater than q on c_k. Letting q_k denote the probability with which player k plays c_k, the expected utility of playing σ_i' is $-q_{i+1}\cdot q_{i+2}/2-\epsilon/2$. On the other hand, the utility of playing $\sigma_i(q)$ is $-\epsilon+q\cdot\delta-q\cdot q_{i+2}/2$. Because $q_{i+1}\leq q$, the difference between these two expressions is at least $\epsilon/2-\delta$, which is positive. It follows that σ_i' dominates $\sigma_i(q)$.

From Claim 2, it follows that all strategies of the form $\sigma_i(q)$ will eventually be eliminated. However, Claim 1 shows that we cannot go ahead and eliminate multiple such strategies for one player, unless at least one other player simultaneously "keeps up" in the eliminated strategies: every time a $\sigma_i(q)$ is eliminated such that $\sigma_{i+1}(q)$ and $\sigma_{i+2}(q)$ have not yet been eliminated, we need to eliminate one of the latter two strategies before any $\sigma_i(q')$ with $q'>q$ can be eliminated—that is, we need to alternate between players. Because there are exponentially many strategies of the form $\sigma_i(q)$, iterated elimination will require exponentially many iterations to complete. ■

[7]Note that the strategies are still pure strategies; the probability placed on an action by a strategy here is simply the sum of the probabilities of the types for which the strategy chooses that action.

It follows that an efficient algorithm for iterated dominance (strict or weak) by pure strategies in Bayesian games, if it exists, must somehow be able to perform (at least part of) many iterations in a single step of the algorithm (because if each step only performed a single iteration, we would need exponentially many steps). Interestingly, Knuth *et al.* [1988] argue that iterated dominance appears to be an inherently sequential problem (in light of their result that iterated very weak dominance is P-complete, that is, apparently not efficiently parallelizable), suggesting that aggregating many iterations may be difficult.

This concludes the part of this dissertation studying the complexity of dominance and iterated dominance. In the next section, we study the complexity of computing Nash equilibria.

9.2 Nash equilibrium

In recent years, there has been a large amount of research on computing Nash equilibria. The question of how hard it is to compute just a single Nash equilibrium especially drew attention, and was dubbed "a most fundamental computational problem whose complexity is wide open" and "together with factoring, [...] the most important concrete open question on the boundary of P today" [Papadimitriou, 2001]. A recent breakthrough series of papers [Daskalakis *et al.*, 2005; Chen and Deng, 2005a; Daskalakis and Papadimitriou, 2005; Chen and Deng, 2005b] shows that the problem is PPAD-complete, even in the two-player case. (An earlier result shows that the problem is no easier if all utilities are required to be in $\{0, 1\}$ [Abbott *et al.*, 2005].) This suggests that the problem is indeed hard, although not as much is known about the class PPAD as about (say) NP. The best-known algorithm for finding a Nash equilibrium, the *Lemke-Howson* algorithm [Lemke and Howson, 1964], has recently been shown to have a worst-case exponential running time [Savani and von Stengel, 2004]. More recent algorithms for computing Nash equilibria have focused on guessing which of the players' pure strategies receive positive probability in the equilibrium: after this guess, only a simple linear feasibility problem needs to be solved [Dickhaut and Kaplan, 1991; Porter *et al.*, 2004; Sandholm *et al.*, 2005b]. (These algorithms clearly require exponential time in the worst case, but are often quite fast in practice.) Also, there has been growing interest in computing equilibria of games with special structure that allows them to be represented concisely [Kearns *et al.*, 2001; Leyton-Brown and Tennenholtz, 2003; Blum *et al.*, 2003; Gottlob *et al.*, 2003; Bhat and Leyton-Brown, 2004; Schoenebeck and Vadhan, 2006].

In this section, we focus mostly on computing equilibria *with certain properties*: for example, computing an equilibrium with maximal social welfare, or one that places probability on a given pure strategy. We also consider the complexity of counting the number of equilibria and computing a pure-strategy Bayes-Nash equilibirium of a Bayesian game.

9.2.1 Equilibria with certain properties in normal-form games

When one analyzes the strategic structure of a game, especially from the viewpoint of a mechanism designer who tries to construct good rules for a game, finding a single equilibrium is far from satisfactory. More desirable equilibria may exist: in this case the game becomes more attractive, especially if one can coax the players into playing a desirable equilibrium. Also, less desirable equilibria may exist: in this case the game becomes less attractive (if there is some chance that these equilibria will end up being played). Before we can make a definite judgment about the quality of the game, we would like to know the answers to questions such as: What is the game's most desirable equilibrium? Is there a unique equilibrium? If not, how many equilibria are there? Algorithms that tackle these questions would be useful both to players and to the mechanism designer.

Furthermore, algorithms that answer certain existence questions may pave the way to designing algorithms that construct a Nash equilibrium. For example, if we had an algorithm that told us whether there exists any equilibrium where a certain player plays a certain strategy, this could be useful in eliminating possibilities in the search for a Nash equilibrium.

However, all the existence questions that we have investigated turn out to be NP-hard. These are not the first results of this nature; most notably, Gilboa and Zemel [1989] provide some NP-hardness results in the same spirit. We provide a single reduction which in demonstrates (sometimes stronger versions of) most of their hardness results, and interesting new results. More significantly, as we show in Subsection 9.2.2, our reduction shows that the problems of maximizing certain properties of Nash equilibria are *inapproximable* (unless ZPP=NP). Additionally, as we show in Subsection 9.2.3, the reduction shows #P-hardness of counting the number of equilibria.

We now present our reduction.[8]

Definition 54 *Let ϕ be a Boolean formula in conjunctive normal form. Let V be its set of variables (with $|V| = n$), L the set of corresponding literals (a positive and a negative one for each variable)*[9]*, and C its set of clauses. The function $v : L \to V$ gives the variable corresponding to a literal, e.g. $v(x_1) = v(-x_1) = x_1$. We define $G_\epsilon(\phi)$ to be the following symmetric 2-player game in normal form. Let $\Sigma \equiv \Sigma_1 = \Sigma_2 = L \cup V \cup C \cup \{f\}$. Let the utility functions be*

- $u_1(l^1, l^2) = u_2(l^2, l^1) = n - 1$ *for all* $l^1, l^2 \in L$ *with* $l^1 \neq -l^2$*;*
- $u_1(l, -l) = u_2(-l, l) = n - 4$ *for all* $l \in L$*;*
- $u_1(l, x) = u_2(x, l) = n - 4$ *for all* $l \in L$*,* $x \in \Sigma - L - \{f\}$*;*
- $u_1(v, l) = u_2(l, v) = n$ *for all* $v \in V$*,* $l \in L$ *with* $v(l) \neq v$*;*
- $u_1(v, l) = u_2(l, v) = 0$ *for all* $v \in V$*,* $l \in L$ *with* $v(l) = v$*;*
- $u_1(v, x) = u_2(x, v) = n - 4$ *for all* $v \in V$*,* $x \in \Sigma - L - \{f\}$*;*
- $u_1(c, l) = u_2(l, c) = n$ *for all* $c \in C$*,* $l \in L$ *with* $l \notin c$*;*
- $u_1(c, l) = u_2(l, c) = 0$ *for all* $c \in C$*,* $l \in L$ *with* $l \in c$*;*
- $u_1(c, x) = u_2(x, c) = n - 4$ *for all* $c \in C$*,* $x \in \Sigma - L - \{f\}$*;*
- $u_1(x, f) = u_2(f, x) = 0$ *for all* $x \in \Sigma - \{f\}$*;*
- $u_1(f, f) = u_2(f, f) = \epsilon$*;*

[8]The reduction presented here is somewhat different from the reduction given in the IJCAI version of this work. The reason is that the new reduction presented here implies inapproximability results that the original reduction did not.

[9]Thus, if x_1 is a variable, x_1 and $-x_1$ are literals. We make a distinction between the variable x_1 and the literal x_1.

- $u_1(f, x) = u_2(x, f) = n - 1$ *for all* $x \in \Sigma - \{f\}$.

Theorem 100 *If* $(l_1, l_2, \ldots, l_n)$ *(where* $v(l_i) = x_i$*) satisfies* ϕ*, then there is a Nash equilibrium of* $G_\epsilon(\phi)$ *where both players play* l_i *with probability* $\frac{1}{n}$*, with expected utility* $n - 1$ *for each player. The only other Nash equilibrium is the one where both players play* f*, and receive expected utility* ϵ *each.*

Proof: We first demonstrate that these combinations of mixed strategies indeed do constitute Nash equilibria. If $(l_1, l_2, \ldots, l_n)$ (where $v(l_i) = x_i$) satisfies ϕ and the other player plays l_i with probability $\frac{1}{n}$, playing one of these l_i as well gives utility $n - 1$. On the other hand, playing the negation of one of these l_i gives utility $\frac{1}{n}(n-4) + \frac{n-1}{n}(n-1) < n-1$. Playing some variable v gives utility $\frac{1}{n}(0) + \frac{n-1}{n}(n) = n - 1$ (since one of the l_i that the other player sometimes plays has $v(l_i) = v$). Playing some clause c gives utility at most $\frac{1}{n}(0) + \frac{n-1}{n}(n) = n - 1$ (since at least one of the l_i that the other player sometimes plays occurs in clause c, since the l_i satisfy ϕ). Finally, playing f gives utility $n - 1$. It follows that playing any one of the l_i that the other player sometimes plays is an optimal response, and hence that both players playing each of these l_i with probability $\frac{1}{n}$ is a Nash equilibrium. Clearly, both players playing f is also a Nash equilibrium since playing anything else when the other plays f gives utility 0.

Now we demonstrate that there are no other Nash equilibria. If the other player always plays f, the unique best response is to also play f since playing anything else will give utility 0. Otherwise, given a mixed strategy for the other player, consider a player's expected utility given that the other player does not play f. (That is, the probability distribution over the other player's strategies is proportional to the probability distribution constituted by that player's mixed strategy, except f occurs with probability 0). If this expected utility is less than $n - 1$, the player is strictly better off playing f (which gives utility $n - 1$ when the other player does not play f, and also performs better than the original strategy when the other player does play f). So this cannot occur in equilibrium.

As we pointed out, here are no Nash equilibria where one player always plays f but the other does not, so suppose both players play f with probability less than one. Consider the expected social welfare ($E[u_1 + u_2]$), given that neither player plays f. It is easily verified that there is no outcome with social welfare greater than $2n - 2$. Also, any outcome in which one player plays an element of V or C has social welfare at most $n - 4 + n < 2n - 2$. It follows that if either player ever plays an element of V or C, the expected social welfare given that neither player plays f is strictly below $2n - 2$. By linearity of expectation it follows that the expected utility of at least one player is strictly below $n - 1$ given that neither player plays f, and by the above reasoning, this player would be strictly better off playing f instead of its randomization over strategies other than f. It follows that no element of V or C is ever played in a Nash equilibrium.

So, we can assume both players only put positive probability on strategies in $L \cup \{f\}$. Then, if the other player puts positive probability on f, playing f is a strictly better response than any element of L (since f does as at least as well against any strategy in L, and strictly better against f). It follows that the only equilibrium where f is ever played is the one where both players always play f.

Now we can assume that both players only put positive probability on elements of L. Suppose that for some $l \in L$, the probability that a given player plays either l or $-l$ is less than $\frac{1}{n}$. Then the expected utility for the other player of playing $v(l)$ is strictly greater than $\frac{1}{n}(0) + \frac{n-1}{n}(n) = n - 1$, and hence this cannot be a Nash equilibrium. So we can assume that for any $l \in L$, the probability that a given player plays either l or $-l$ is precisely $\frac{1}{n}$.

If there is an element of L such that player 1 puts positive probability on it and player 2 on its negation, both players have expected utility less than $n-1$ and would be better off switching to f. So, in a Nash equilibrium, if player 1 plays l with some probability, player 2 must play l with probability $\frac{1}{n}$, and thus player 1 must play l with probability $\frac{1}{n}$. Thus we can assume that for each variable, exactly one of its corresponding literals is played with probability $\frac{1}{n}$ by both players. It follows that in any Nash equilibrium (besides the one where both players play f), literals that are sometimes played indeed correspond to an assignment to the variables.

All that is left to show is that if this assignment does not satisfy ϕ, it does not correspond to a Nash equilibrium. Let $c \in C$ be a clause that is not satisfied by the assignment, that is, none of its literals are ever played. Then playing c would give utility n, and both players would be better off playing this. ■

Example 1 *The following table shows the game $G_\epsilon(\phi)$ where $\phi = (x_1 \vee -x_2) \wedge (-x_1 \vee x_2)$.*

	x_1	x_2	$+x_1$	$-x_1$	$+x_2$	$-x_2$	$(x_1 \vee -x_2)$	$(-x_1 \vee x_2)$	f
x_1	-2,-2	-2,-2	0,-2	0,-2	2,-2	2,-2	-2,-2	-2,-2	0,1
x_2	-2,-2	-2,-2	2,-2	2,-2	0,-2	0,-2	-2,-2	-2,-2	0,1
$+x_1$	-2,0	-2,2	1,1	-2,-2	1,1	1,1	-2,0	-2,2	0,1
$-x_1$	-2,0	-2,2	-2,-2	1,1	1,1	1,1	-2,2	-2,0	0,1
$+x_2$	-2,2	-2,0	1,1	1,1	1,1	-2,-2	-2,2	-2,0	0,1
$-x_2$	-2,2	-2,0	1,1	1,1	-2,-2	1,1	-2,0	-2,2	0,1
$(x_1 \vee -x_2)$	-2,-2	-2,-2	0,-2	2,-2	2,-2	0,-2	-2,-2	-2,-2	0,1
$(-x_1 \vee x_2)$	-2,-2	-2,-2	2,-2	0,-2	0,-2	2,-2	-2,-2	-2,-2	0,1
f	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	ϵ,ϵ

The only two solutions to the SAT instance defined by ϕ is to either set both variables to true*, or both to* false*. Indeed, the only equilibria of the game $G_\epsilon(\phi)$ are those where: 1. Both players randomize uniformly over $\{+x_1, +x_2\}$; 2. Both players randomize uniformly over $\{-x_1, -x_2\}$; 3. Both players play f. So the example is consistent with Theorem 100.*

Thus, in general, there exists a Nash equilibrium in $G_\epsilon(\phi)$ where each player gets utility $n-1$ if and only if ϕ is satisfiable; otherwise, the only equilibrium is the one where both players play f and each of them gets ϵ. Suppose $n-1 > \epsilon$. Then, any sensible definition of welfare optimization would prefer the first kind of equilibrium. So, it follows that determining whether a "good" equilibrium exists is hard for any such definition. Additionally, the first kind of equilibrium is, in various senses, an optimal outcome for the game, even if the players were to cooperate, so even finding out whether such an optimal equilibrium exists is hard. The corollaries below illustrate these points.

All the corollaries show NP-completeness of a problem, meaning that the problem is both NP-hard and in NP. Technically, only the NP-hardness part is a corollary of Theorem 100 in each case. Membership in NP follows in each case because we can nondeterministically generate strategies for the players, and verify whether these constitute a Nash equilibrium with the desired property.

Alternatively, for the case of two players, we can nondeterministically generate only the supports of the players' strategies. At this point, determining whether a Nash equilibrium with the given supports exists is a simple linear feasibility program (see, for example, Dickhaut and Kaplan [1991]; Porter *et al.* [2004]), to which we can add an objective to maximize (such as, for example, social welfare). The resulting linear program can be solved in polynomial time [Khachiyan, 1979].

Corollary 9 *Even in symmetric 2-player games, it is NP-complete to determine whether there exists a NE with expected (standard) social welfare ($E[\sum_{1 \le i \le |A|} u_i]$) at least k, even when k is the maximum social welfare that could be obtained in the game.*

Proof: For any ϕ, in $G_\epsilon(\phi)$, the social welfare of a Nash equilibrium corresponding to any satisfying assignment is $2(n-1)$. On the other hand, the social welfare of the Nash equilibrium that always exists is only 2ϵ. Thus, for $\epsilon < 1$ and $n \geq 2$, $G_\epsilon(\phi)$ has a Nash equilibrium with a social welfare of at least $2(n-1)$ if and only if ϕ is satisfiable. ■

Corollary 10 *Even in symmetric 2-player games, it is NP-complete to determine whether there exists a NE where all players have expected utility at least k (that is, the* egalitarian social welfare *is at least k), even when k is the largest number such that there exists a distribution over outcomes of the game such that all players have expected utility at least k.*

Proof: For any ϕ, in $G_\epsilon(\phi)$, the egalitarian social welfare of a Nash equilibrium corresponding to any satisfying assignment is $n-1$. On the other hand, the egalitarian social welfare of the Nash equilibrium that always exists is only ϵ. Thus, for $\epsilon < 1$ and $n \geq 2$, $G_\epsilon(\phi)$ has a Nash equilibrium with an egalitarian social welfare of at least $n-1$ if and only if ϕ is satisfiable. ■

Corollary 11 *Even in symmetric 2-player games, it is NP-complete to determine whether there exists a* Pareto-optimal *NE. (A distribution over outcomes is Pareto-optimal if there is no other distribution over outcomes such that every player has at least equal expected utility, and at least one player has strictly greater expected utility.)*

Proof: For $\epsilon < 1$ and $n \geq 2$, any Nash equilibrium in $G_\epsilon(\phi)$ corresponding to a satisfying assignment is Pareto-optimal, whereas the Nash equilibrium that always exists is not Pareto-optimal. Thus, a Pareto optimal Nash equilibrium exists if and only if ϕ is satisfiable. ■

Corollary 12 *Even in symmetric 2-player games, it is NP-complete to determine whether there exists a NE where player 1 has expected utility at least k.*

Proof: For any ϕ, in $G_\epsilon(\phi)$, player 1's utility in a Nash equilibrium corresponding to any satisfying assignment is $(n-1)$. On the other hand, player 1's utility in the Nash equilibrium that always exists is only ϵ. Thus, for $\epsilon < 1$ and $n \geq 2$, $G_\epsilon(\phi)$ has a Nash equilibrium with a utility for player 1 of at least $n-1$ if and only if ϕ is satisfiable. ■

Some additional corollaries are:

Corollary 13 *Even in symmetric 2-player games, it is NP-complete to determine whether there is more than one Nash equilibrium.*

Proof: For any ϕ, $G_\epsilon(\phi)$ has additional Nash equilibria (besides the one that always exists) if and only if ϕ is satisfiable. ■

Corollary 14 *Even in symmetric 2-player games, it is NP-complete to determine whether there is an equilibrium where player 1 sometimes plays a given* $x \in \Sigma_1$.

Proof: For any ϕ, in $G_\epsilon(\phi)$, there is a Nash equilibrium where player 1 sometimes plays $+x_1$ if and only if there is a satisfying assignment to ϕ with x_1 set to *true*. But determining whether this is the case is NP-complete. ■

Corollary 15 *Even in symmetric 2-player games, it is NP-complete to determine whether there is an equilibrium where player 1 never plays a given* $x \in \Sigma_1$.

Proof: For any ϕ, in $G_\epsilon(\phi)$, there is a Nash equilibrium where player 1 never plays f if and only if ϕ is satisfiable. ■

Corollary 16 *Even in symmetric 2-player games, it is NP-complete to determine whether there is an equilibrium where player 1's strategy has at least* k *pure strategies in its support (even when* $k = 2$*).*

Proof: For any ϕ, in $G_\epsilon(\phi)$, any Nash equilibrium corresponding to a satisfying assignment uses a support of n strategies for player 1. On the other hand, the Nash equilibrium that always exists uses a support of only 1 strategy for player 1. Thus, for $n \geq 2$, $G_\epsilon(\phi)$ has a Nash equilibrium using a support of at least 2 strategies for player 1 if and only if ϕ is satisfiable. ■

Corollary 17 *Even in symmetric 2-player games, it is NP-complete to determine whether there is an equilibrium where the players' strategies together have at least* k *pure strategies in their supports (even when* $k = 3$*).*

Proof: For any ϕ, in $G_\epsilon(\phi)$, any Nash equilibrium corresponding to a satisfying assignment uses a support of n strategies for each player, for a total of $2n$ strategies. On the other hand, the Nash equilibrium that always exists uses a support of only 1 strategy for each player, for a total of only 2 strategies. Thus, for $n \geq 2$, $G_\epsilon(\phi)$ has a Nash equilibrium using at least 3 strategies in the supports of the players if and only if ϕ is satisfiable. ■

Corollary 18 *Even in symmetric 2-player games, it is NP-complete to determine whether there is an equilibrium where each player's strategy has at least* k *pure strategies in its support (even when* $k = 2$*).*

Proof: For any ϕ, in $G_\epsilon(\phi)$, any Nash equilibrium corresponding to a satisfying assignment uses a support of n strategies for each player. On the other hand, the Nash equilibrium that always exists uses a support of only 1 strategy for each player. Thus, for $n \geq 2$, $G_\epsilon(\phi)$ has a Nash equilibrium using at least 2 strategies in the supports of each player if and only if ϕ is satisfiable. ■

Definition 55 *A* strong Nash equilibrium *is a vector of mixed strategies for the players so that no nonempty subset of the players can change their strategies to make all players in the subset better off.*

Corollary 19 *Even in symmetric 2-player games, it is NP-complete to determine whether a strong Nash equilibrium exists.*

Proof: For $\epsilon < 1$ and $n \geq 2$, any Nash equilibrium in $G_\epsilon(\phi)$ corresponding to a satisfying assignment is a strong Nash equilibrium, whereas the Nash equilibrium that always exists is not strong. Thus, a strong Nash equilibrium exists if and only if ϕ is satisfiable. ■

All of these results indicate that it is hard to obtain summary information about a game's Nash equilibria. (Corollaries 13, 18, and weaker[10] versions of Corollaries 10, 14, and 15 were first proven by Gilboa and Zemel [1989].)

9.2.2 Inapproximability results

Some of the corollaries of the previous subsection state that it is NP-complete to find the Nash equilibrium that maximizes a certain property (such as social welfare). For such properties, an important question is to ask whether they can be approximated. For instance, is it possible to find, in polynomial time, a Nash equilibrium that has at least half as great a social welfare as the social-welfare maximizing Nash equilibrium? Or—the same question, asked nonconstructively—can we, in polynomial time, find a number k such that there exists a Nash equilibrium with social welfare at least k, and there is no Nash equilibrium with social welfare greater than $2k$? (The nonconstructive question does not require constructing a Nash equilibrium, so it is perhaps possible that there is a polynomial-time algorithm for this question even if it is hard to construct any Nash equilibrium.) We will not give approximation algorithms in this subsection, but we will derive certain inapproximability results from Theorem 100. In each case, we will show that even the nonconstructive question is hard (and therefore also the constructive question).

Before presenting our results, we first make one subtle technical point, namely that it is unreasonable to expect an approximation algorithm to work even when the game has some negative utilities in it. For suppose we had an approximation algorithm that approximated (say) social welfare to some positive ratio, even when there are some negative utilities in the game. Then we can "boost" its results, as follows. Suppose it returned a social welfare of $2r$ on a game, and suppose it were less than the social welfare of the best Nash equilibrium. If we subtract r from all utilities in the game, the game remains the same for all strategic purposes (it has the same set of Nash equilibria). But now the result provided by the approximation algorithm on the original game corresponds

[10]Our results prove hardness in a slightly more restricted setting. Corollaries 14 and 15 in their full strength can in fact also be obtained using Gilboa and Zemel's proof technique, even though they stated the result in a weaker form.

to a social welfare of 0, which does not satisfy the approximation ratio. It follows that running the approximation algorithm on the transformed game must give a better result (which we can easily transform back to the original game).

For this reason, we require our hardness results to only use reductions to games where 0 is the lowest possible utility in the game. To do so, we will simply use the fact that $G_\epsilon(\phi)$ satisfies this property whenever $n \geq 4$. (We recall that n is the number of variables in ϕ.)

We are now ready to present our results. The first one is a stronger version of Corollary 9.

Corollary 20 *Unless ZPP = NP, there does not exist a polynomial-time algorithm that approximates (to any positive ratio) the maximum social welfare obtained by a Nash equilibrium, even in symmetric 2-player games. (Even if the ratio is allowed to be a function of the size of the game.)*

Proof: Suppose such an algorithm did exist. For any formula ϕ (with number of variables $n \geq 4$), consider the game $G_\epsilon(\phi)$ where ϵ is set so that $2\epsilon < r(2n-2)$ (here, r is the approximation ratio that the algorithm guarantees for games of the size of $G_\epsilon(\phi)$). If ϕ is satisfiable, by Theorem 100, there exists an equilibrium with social welfare $2n-2$, and thus the approximation algorithm should return a social welfare of at least $r(2n-2) > 2\epsilon$. Otherwise, by Theorem 100, the only equilibrium has social welfare 2ϵ, and thus the approximation algorithm should return a social welfare of at most 2ϵ. Thus we can use the algorithm to solve arbitrary SAT instances. ■

The next result is a stronger version of Corollary 10.

Corollary 21 *Unless ZPP = NP, there does not exist a polynomial-time algorithm that approximates (to any positive ratio) the maximum egalitarian social welfare (minimum utility) obtained by a Nash equilibrium, even in symmetric 2-player games. (Even if the ratio is allowed to be a function of the size of the game.)*

Proof: Suppose such an algorithm did exist. For any formula ϕ (with number of variables $n \geq 4$), consider the game $G_\epsilon(\phi)$ where ϵ is set so that $\epsilon < r(n-1)$ (here, r is the approximation ratio that the algorithm guarantees for games of the size of $G_\epsilon(\phi)$). If ϕ is satisfiable, by Theorem 100, there exists an equilibrium with egalitarian social welfare $n-1$, and thus the approximation algorithm should return an egalitarian social welfare of at least $r(n-1) > \epsilon$. Otherwise, by Theorem 100, the only equilibrium has egalitarian social welfare ϵ, and thus the approximation algorithm should return an egalitarian social welfare of at most ϵ. Thus we can use the algorithm to solve arbitrary SAT instances. ■

The next result is a stronger version of Corollary 12.

Corollary 22 *Unless ZPP = NP, there does not exist a polynomial-time algorithm that approximates (to any positive ratio) the maximum utility for player* 1 *obtained by a Nash equilibrium, even in symmetric 2-player games. (Even if the ratio is allowed to be a function of the size of the game.)*

Proof: Suppose such an algorithm did exist. For any formula ϕ (with number of variables $n \geq 4$), consider the game $G_\epsilon(\phi)$ where ϵ is set so that $\epsilon < r(n-1)$ (here, r is the approximation ratio that the algorithm guarantees for games of the size of $G_\epsilon(\phi)$). If ϕ is satisfiable, by Theorem 100, there exists an equilibrium with a utility of $n-1$ for player 1, and thus the approximation algorithm should return a utility of at least $r(n-1) > \epsilon$. Otherwise, by Theorem 100, the only equilibrium has a utility of ϵ for player 1, and thus the approximation algorithm should return a utility of at most ϵ. Thus we can use the algorithm to solve arbitrary SAT instances. ■

The next result is a stronger version of Corollary 16.

Corollary 23 *Unless ZPP = NP, there does not exist a polynomial-time algorithm that approximates (to any ratio $o(|\Sigma|)$) the maximum number of pure strategies in player 1's support in a Nash equilibrium, even in symmetric 2-player games.*

Proof: Suppose such an algorithm did exist. For any formula ϕ, consider the game $G_\epsilon(\phi)$ where ϵ is set arbitrarily. If ϕ is not satisfiable, by Theorem 100, the only equilibrium has only one pure strategy in player 1's support, and thus the algorithm can return a maximum support size of at most 1. On the other hand, if ϕ is satisfiable, by Theorem 100, there is an equilibrium where player 1's support has size $\Omega(|\Sigma|)$. Because by assumption our approximation algorithm has an approximation ratio of $o(|\Sigma|)$, this means that for large enough $|\Sigma|$, the approximation ratio must return a support size strictly greater than 1. Thus we can use the algorithm to solve arbitrary SAT instances (given that the instances are large enough to produce large enough $|\Sigma|$). ■

The next result is a stronger version of Corollary 17.

Corollary 24 *Unless ZPP = NP, there does not exist a polynomial-time algorithm that approximates (to any ratio $o(|\Sigma|)$) the maximum number of pure strategies in the players' strategies' supports in a Nash equilibrium, even in symmetric 2-player games.*

Proof: Suppose such an algorithm did exist. For any formula ϕ, consider the game $G_\epsilon(\phi)$ where ϵ is set arbitrarily. If ϕ is not satisfiable, by Theorem 100, the only equilibrium has only one pure strategy in each player's support, and thus the algorithm can return a number of strategies of at most 2. On the other hand, if ϕ is satisfiable, by Theorem 100, there is an equilibrium where each player's support has size $\Omega(|\Sigma|)$. Because by assumption our approximation algorithm has an approximation ratio of $o(|\Sigma|)$, this means that for large enough $|\Sigma|$, the approximation ratio must return a support size strictly greater than 2. Thus we can use the algorithm to solve arbitrary SAT instances(given that the instances are large enough to produce large enough $|\Sigma|$). ■

The next result is a stronger version of Corollary 18.

Corollary 25 *Unless ZPP = NP, there does not exist a polynomial-time algorithm that approximates (to any ratio $o(|\Sigma|)$) the maximum number, in a Nash equilibrium, of pure strategies in the support of the player that uses fewer pure strategies than the other, even in symmetric 2-player games.*

Proof: Suppose such an algorithm did exist. For any formula ϕ, consider the game $G_\epsilon(\phi)$ where ϵ is set arbitrarily. If ϕ is not satisfiable, by Theorem 100, the only equilibrium has only one pure strategy in each player's support, and thus the algorithm can return a number of strategies of at most 1. On the other hand, if ϕ is satisfiable, by Theorem 100, there is an equilibrium where each player's support has size $\Omega(|\Sigma|)$. Because by assumption our approximation algorithm has an approximation ratio of $o(|\Sigma|)$, this means that for large enough $|\Sigma|$, the approximation ratio must return a support size strictly greater than 1. Thus we can use the algorithm to solve arbitrary SAT instances(given that the instances are large enough to produce large enough $|\Sigma|$). ■

9.2.3 Counting the number of equilibria in normal-form games

Existence questions do not tell the whole story. In general, we are interested in characterizing all the equilibria of a game. One rather weak such characterization is the number of equilibria[11]. We can use Theorem 100 to show that even determining this number in a given normal-form game is hard.

Corollary 26 *Even in symmetric 2-player games, counting the number of Nash equilibria is #P-hard.*

Proof: The number of Nash equilibria in our game $G_\epsilon(\phi)$ is the number of satisfying assignments to the variables of ϕ, plus one. Counting the number of satisfying assignments to a CNF formula is #P-hard [Valiant, 1979]. ■

It is easy to construct games where there is a continuum of Nash equilibria. In such games, it is more meaningful to ask how many distinct continuums of equilibria there are. More formally, one can ask how many maximal connected sets of equilibria a game has (a maximal connected set is a connected set which is not a proper subset of a connected set).

Corollary 27 *Even in symmetric 2-player games, counting the number of maximal connected sets of Nash equilibria is #P-hard.*

Proof: Every Nash equilibrium in $G_\epsilon(\phi)$ constitutes a maximal connected set by itself, so the number of maximal connected sets is the number of satisfying assignments to the variables of ϕ, plus one. ■

The most interesting #P-hardness results are the ones where the corresponding existence and search questions are easy, such as counting the number of perfect bipartite matchings. In the case of Nash equilibria, the existence question is trivial: it has been analytically shown (by Kakutani's fixed point theorem) that a Nash equilibrium always exists [Nash, 1950]. The complexity of the search question remains open.

9.2.4 Pure-strategy Bayes-Nash equilibria

Equilibria in pure strategies are particularly desirable because they avoid the uncomfortable requirement that players randomize over strategies among which they are indifferent [Fudenberg and

[11] The number of equilibria in normal-form games has been studied both in the worst case [McLennan and Park, 1999] and in the average case [McLennan, 1999].

Tirole, 1991]. In normal-form games with small numbers of players, it is easy to determine the existence of pure-strategy equilibria: one can simply check, for each combination of pure strategies, whether it constitutes a Nash equilibrium.[12] However, this is not feasible in Bayesian games, where the players have private information about their own preferences (represented by types). Here, players may condition their actions on their types, so the strategy space of each player is exponential in the number of types.

In this subsection, we show that the question of whether a pure-strategy Bayes-Nash equilibrium exists is in fact NP-hard even in symmetric two-player games.

We study the following computational problem.

Definition 56 (PURE-STRATEGY-BNE) *We are given a Bayesian game. We are asked whether there exists a Bayes-Nash equilibrium (BNE) where all the strategies σ_{i,θ_i} are pure.*

To show our NP-hardness result, we will reduce from the SET-COVER problem.

Definition 57 (SET-COVER) *We are given a set $S = \{s_1, \ldots, s_n\}$, subsets $S_1, S_2, \ldots, S_m$ of S with $\bigcup_{1 \leq i \leq m} S_i = S$, and an integer k. We are asked whether there exist $S_{c_1}, S_{c_2}, \ldots, S_{c_k}$ such that $\bigcup_{1 \leq i \leq k} S_{c_i} = S$.*

Theorem 101 *PURE-STRATEGY-BNE is NP-complete, even in symmetric 2-player games where the priors over types are uniform.*

Proof: To show membership in NP, we observe that we can nondeterministically choose a pure strategy for each type for each player, and verify whether these constitute a BNE.

To show NP-hardness, we reduce an arbitrary SET-COVER instance to the following PURE-STRATEGY-BNE instance. Let there be two players, with $\Theta \equiv \Theta_1 = \Theta_2 = \{\theta^1, \ldots, \theta^k\}$. The priors over types are uniform. Furthermore, $\Sigma \equiv \Sigma_1 = \Sigma_2 = \{S_1, S_2, \ldots, S_m, s_1, s_2, \ldots, s_n\}$. The utility functions we choose in fact do not depend on the types, so we omit the type argument in their definitions. They are as follows:

- $u_1(S_i, S_j) = u_2(S_j, S_i) = 1$ for all S_i and S_j;
- $u_1(S_i, s_j) = u_2(s_j, S_i) = 1$ for all S_i and $s_j \notin S_i$;
- $u_1(S_i, s_j) = u_2(s_j, S_i) = 2$ for all S_i and $s_j \in S_i$;
- $u_1(s_i, s_j) = u_2(s_j, s_i) = -3k$ for all s_i and s_j;
- $u_1(s_j, S_i) = u_2(S_i, s_j) = 3$ for all S_i and $s_j \notin S_i$;
- $u_1(s_j, S_i) = u_2(S_i, s_j) = -3k$ for all S_i and $s_j \in S_i$.

[12] Computing pure-strategy Nash equilibria for more concise representations of normal-form games has been systematically studied [Gottlob *et al.*, 2003; Schoenebeck and Vadhan, 2006].

We now show the two instances are equivalent. First suppose there exist $S_{c_1}, S_{c_2}, \ldots, S_{c_k}$ such that $\bigcup_{1 \leq i \leq k} S_{c_i} = S$. Suppose both players play as follows: when their type is θ_i, they play S_{c_i}. We claim that this is a BNE. For suppose the other player employs this strategy. Then, because for any s_j, there is at least one S_{c_i} such that $s_j \in S_{c_i}$, we have that the expected utility of playing s_j is at most $\frac{1}{k}(-3k) + \frac{k-1}{k}3 < 0$. It follows that playing any of the S_j (which gives utility 1) is optimal. So there is a pure-strategy BNE.

On the other hand, suppose that there is a pure-strategy BNE. We first observe that in no pure-strategy BNE, both players play some element of S for some type: for if the other player sometimes plays some s_j, the utility of playing some s_i is at most $\frac{1}{k}(-3k) + \frac{k-1}{k}3 < 0$, whereas playing some S_i instead guarantees a utility of at least 1. So there is at least one player who never plays any element of S. Now suppose the other player sometimes plays some s_j. We know there is some S_i such that $s_j \in S_i$. If the former player plays this S_i, this will give it a utility of at least $\frac{1}{k}2 + \frac{k-1}{k}1 = 1 + \frac{1}{k}$. Since it must do at least this well in the equilibrium, and it never plays elements of S, it must sometimes receive utility 2. It follows that there exist S_a and $s_b \in S_a$ such that the former player sometimes plays S_a and the latter sometimes plays s_b. But then, playing s_b gives the latter player a utility of at most $\frac{1}{k}(-3k) + \frac{k-1}{k}3 < 0$, and it would be better off playing some S_i instead. (Contradiction.) It follows that in no pure-strategy BNE, any element of S is ever played.

Now, in our given pure-strategy equilibrium, consider the set of all the S_i that are played by player 1 for some type. Clearly there can be at most k such sets. We claim they cover S. For if they do not cover some element s_j, the expected utility of playing s_j for player 2 is 3 (because player 1 never plays any element of S). But this means that player 2 (who never plays any element of S either) is not playing optimally. (Contradiction.) Hence, there exists a set cover. ■

If one allows for general mixed strategies, a Bayes-Nash equilibrium always exists [Fudenberg and Tirole, 1991]. Computing a single mixed-strategy Bayes-Nash equilibrium is of course at least as hard as computing a single mixed-strategy Nash equilibrium in a normal-form game (since that is the special case where each agent has a single type).

This concludes the part of this dissertation studying the complexity of computing Nash equilibria. The next section introduces a parameterized strategy eliminability criterion that generalizes both dominance and Nash equilibrium, and studies how hard it is to apply computationally.

9.3 A generalized eliminability criterion

The concept of (iterated) dominance is often too strong for the purpose of solving games: it cannot eliminate enough strategies. But, if possible, we would like a stronger argument for eliminating a strategy than (mixed-strategy) Nash equilibrium. Similarly, in mechanism design (where one gets to create the game), implementation in dominant strategies is often excessively restrictive, but implementation in (Bayes-)Nash equilibrium may not be sufficiently strong for the designer's purposes. Hence, it is desirable to have eliminability criteria that are *between* these concepts in strength. In this section, we will introduce such a criterion. This criterion considers whether a given strategy is eliminable relative to given dominator & eliminee subsets of the players' strategies. The criterion spans an entire *spectrum* of strength between Nash equilibrium and strict dominance (in terms of which strategies it can eliminate), and in the extremes can be made to coincide with either of these two concepts, depending on how the dominator & eliminee sets are set. It can also be used for iterated elimination of strategies. We will also study the computational complexity of applying the new eliminability criterion, and provide a mixed integer programming approach for it.

One of the benefits of the new criterion is that when a strategy cannot be eliminated by dominance (but it can be eliminated by the Nash equilibrium concept), the new criterion may provide a stronger argument than Nash equilibrium for eliminating the strategy, by using dominator & eliminee sets smaller than the entire strategy set. To get the strongest possible argument for eliminating a strategy, the dominator & eliminee sets should be chosen to be as small as possible while still having the strategy be eliminable relative to these sets.[13] Iterated elimination of strategies using the new criterion is also possible, and again, to get the strongest possible argument for eliminating a strategy, the sequence of eliminations leading up to it should use dominator & eliminee sets that are as small as possible.[14]

As another benefit, the algorithm that we provide for checking whether a strategy is eliminable according to the new criterion can also be used as a subroutine in the computation of Nash equilibria. Specifically, any strategy that is eliminable (even using iterated elimination) according to the criterion is guaranteed not to occur in any Nash equilibrium. Current state-of-the-art algorithms for computing Nash equilibria already use a subroutine that eliminates (conditionally) dominated strategies [Porter *et al.*, 2004]. Because the new criterion can eliminate more strategies than dominance, the algorithm we provide may speed up the computation of Nash equilibria. (For purposes of speed, it is probably desirable to only apply special cases of the criterion that can be computed fast—in particular, as we will show, eliminability according to the criterion can be computed fast when the eliminee sets are small. Even these special cases are more powerful than dominance.)

Throughout, we focus on two-player games only. The eliminability criterion itself can be generalized to more players, but the computational tools we introduce do not straightforwardly generalize to more players. Moreover, we restrict attenton to normal-form games only.

9.3.1 A motivating example

Because the definition of the new eliminability criterion is complex, we will first illustrate it with an example. Consider the following (partially specified) game.

	σ_c^1	σ_c^2	σ_c^3	σ_c^4
σ_r^1	?, ?	?, 2	?, 0	?, 0
σ_r^2	2, ?	2, 2	2, 0	2, 0
σ_r^3	0, ?	0, 2	3, 0	0, 3
σ_r^4	0, ?	0, 2	0, 3	3, 0

[13] There may be multiple minimal vectors of dominator & eliminee sets relative to which the strategy is eliminable; in this dissertation, we will not attempt to settle which of these minimal vectors, if any, constitutes the most powerful argument for eliminating the strategy.

[14] Here, there may also be a tradeoff with the length of the elimination path. For example, there may be a path of several eliminations using dominator & eliminee sets that are small, as well as a single elimination using dominator & eliminee sets that are large, both of which eliminate a given strategy. (In fact, we will *always* be confronted with this situation, as Corollary 30 will show.) Again, in this dissertation, we will not attempt to settle which argument for eliminating the strategy is stronger.

A quick look at this game reveals that strategies σ_r^3 and σ_r^4 are both *almost* dominated by σ_r^2—but they perform better than σ_r^2 against σ_c^3 and σ_c^4, respectively. Similarly, strategies σ_c^3 and σ_c^4 are both almost dominated by σ_c^2—but they perform better than σ_c^2 against σ_r^4 and σ_c^3, respectively. So we are unable to eliminate any strategies using (even weak) dominance.

Now consider the following reasoning. In order for it to be worthwhile for the row player to ever play σ_r^3 rather than σ_r^2, the column player should play σ_c^3 at least $\frac{2}{3}$ of the time. (If it is exactly $\frac{2}{3}$, then switching from σ_r^2 to σ_r^3 will cost the row player 2 exactly $\frac{1}{3}$ of the time, but the row player will gain 1 exactly $\frac{2}{3}$ of the time, so the expected benefit is 0.) But, similarly, in order for it to be worthwhile for the column player to ever play σ_c^3, the row player should play σ_r^4 at least $\frac{2}{3}$ of the time. But again, in order for it to be worthwhile for the row player to ever play σ_r^4, the column player should play σ_c^4 at least $\frac{2}{3}$ of the time. Thus, if both the row and the column player accurately assess the probabilities that the other places on these strategies, and their strategies are rational with respect to these assessments (as would be the case in a Nash equilibrium), then, if the row player puts positive probability on σ_r^3, by the previous reasoning, the column player should be playing σ_c^3 at least $\frac{2}{3}$ of the time, and σ_c^4 at least $\frac{2}{3}$ of the time. Of course, this is impossible; so, in a sense, the row player should not play σ_r^3.

It may appear that all we have shown is that σ_r^3 is not played in any Nash equilibrium. But, to some extent, our argument for not playing σ_r^3 did not make use of the full elimination power of the Nash equilibrium concept. Most notably, we only reasoned about a small part of the game: we never mentioned strategies σ_r^1 and σ_c^1, and we did not even specify most of the utilities for these strategies. (It is easy to extend this example so that the argument only uses an arbitrarily small fraction of the strategies and of the utilities in the matrix, for instance by adding many copies of σ_r^1 and σ_c^1.) The locality of the reasoning that we did is more akin to the notion of dominance, which is perhaps the extreme case of local reasoning about eliminability—only two strategies are mentioned in it. So, in this sense, the argument for eliminating σ_r^3 is somewhere between dominance and Nash equilibrium in strength.

9.3.2 Definition of the eliminability criterion

We are now ready to give the formal definition of the generalized eliminability criterion. To make the definition a bit simpler, we define its negation—when a strategy is *not* eliminable relative to certain sets of strategies. Also, we only define when one of the *row player's* strategies is eliminable, but of course the definition is analogous for the column player.

The definition, which considers when a strategy e_r^* is eliminable relative to subsets D_r, E_r of the row player's pure strategies (with $e_r^* \in E_r$) and subsets D_c, E_c of the column player's pure strategies, can be stated informally as follows. To protect e_r^* from elimination, we should be able to specify the probabilities that the players' mixed strategies place on the E_i sets in such a way that 1) e_r^* receives nonzero probability, and 2) for every pure strategy e_i that receives nonzero probability, for every mixed strategy d_i using only strategies in D_i, it is conceivable that player $-i$'s mixed strategy[15] is completed so that e_i is no worse than d_i.[16] The formal definition follows.

[15] As is common in the game theory literature, $-i$ denotes "the player other than i."

[16] This description may sound similar to the concept of *rationalizability*. However, in two-player games (the subject of this section), rationalizability is known to coincide with iterated strict dominance [Pearce, 1984].

Definition 58 *Given a two-player game in normal form, subsets D_r, E_r of the row player's pure strategies Σ_r, subsets D_c, E_c of the column player's pure strategies Σ_c, and a distinguished strategy $e_r^* \in E_r$, we say that e_r^* is* not eliminable relative to *D_r, E_r, D_c, E_c, if there exist functions (partial mixed strategies) $p_r : E_r \to [0,1]$ and $p_c : E_c \to [0,1]$ with $p_r(e_r^*) > 0$, $\sum_{e_r \in E_r} p_r(e_r) \leq 1$, and $\sum_{e_c \in E_c} p_c(e_c) \leq 1$, such that the following holds. For both $i \in \{r, c\}$, for any $e_i \in E_i$ with $p_i(e_i) > 0$, for any mixed strategy d_i placing positive probability only on strategies in D_i, there is some pure strategy $\sigma_{-i} \in \Sigma_{-i} - E_{-i}$ such that (letting $p_{-i} \diamond \sigma_{-i}$ denote the mixed strategy that results from placing the remaining probability $1 - \sum_{e_{-i} \in E_{-i}} p_{-i}(e_{-i})$ that is not used by the partial mixed strategy p_{-i} on σ_{-i}), we have: $u_i(e_i, p_{-i} \diamond \sigma_{-i}) \geq u_i(d_i, p_{-i} \diamond \sigma_{-i})$. (If p_{-i} already uses up all the probability, we simply have $u_i(e_i, p_{-i}) \geq u_i(d_i, p_{-i})$—no σ_{-i} needs to be chosen.)*[17]

In the example from the previous subsubsection, we can set $D_r = \{\sigma_r^2\}$, $D_c = \{\sigma_c^2\}$, $E_r = \{\sigma_r^3, \sigma_r^4\}$, $E_c = \{\sigma_c^3, \sigma_c^4\}$, and $e_r^* = \sigma_r^3$. Then, by the reasoning that we did, it is impossible to set p_r and p_c so that the conditions are satisfied, and hence σ_r^3 is eliminable relative to these sets.

9.3.3 The spectrum of strength

In this subsection we show that the generalized eliminability criterion we defined in in the previous subsection spans a spectrum of strength all the way from Nash equilibrium (when the sets D_r, E_r, D_c, E_c are chosen as large as possible), to strict dominance (when the sets are chosen as small as possible). First, we show that the criterion is monotonically increasing, in the sense that the larger we make the sets D_r, E_r, D_c, E_c, the more strategies are eliminable.

Proposition 12 *If e_r^* is eliminable relative to $D_r^1, E_r^1, D_c^1, E_c^1$, and $D_r^1 \subseteq D_r^2, E_r^1 \subseteq E_r^2, D_c^1 \subseteq D_c^2, E_c^1 \subseteq E_c^2$, then e_r^* is eliminable relative to $D_r^2, E_r^2, D_c^2, E_c^2$.*

Proof: We will prove this by showing that if e_r^* is not eliminable relative to $D_r^2, E_r^2, D_c^2, E_c^2$, then e_r^* is not eliminable relative to $D_r^1, E_r^1, D_c^1, E_c^1$. It is straightforward that making the D_i sets smaller only weakens the condition on strategies e_i with $p_i(e_i) > 0$ in Definition 58. Hence, if e_r^* is not eliminable relative to $D_r^2, E_r^2, D_c^2, E_c^2$, then e_r^* is not eliminable relative to $D_r^1, E_r^2, D_c^1, E_c^2$. All that remains to show is that making the E_i sets smaller will not make e_r^* eliminable. To show this, we first observe that, if in its last step Definition 58 allowed for distributing the remaining probability arbitrarily over the strategies in $\Sigma_{-i} - E_{-i}$ (rather than requiring a single one of these strategies to receive all the remaining probability), this would not change the definition, because we might as well place all the remaining probability on the strategy $\sigma_{-i} \in \Sigma_{-i} - E_{-i}$ that maximizes $u_i(e_i, \sigma_{-i}) - u_i(d_i, \sigma_{-i})$. Now, let p_r and p_c be partial mixed strategies over E_r^2 and E_c^2 that prove that e_r^* is not eliminable relative to $D_r^1, E_r^2, D_c^1, E_c^2$. Then, to show that e_r^* is not eliminable relative to $D_r^1, E_r^1, D_c^1, E_c^1$, use the partial mixed strategies p_r' and p_c', which are simply the restrictions of p_r and p_c to E_r^1 and E_c^1, respectively. For any $e_i \in E_i^1$ with $p_i'(e_i) > 0$ and for any mixed strategy d_i over D_i^1, we know that there exists some $\sigma_{-i} \in \Sigma_{-i} - E_{-i}^2$ such that $u_i(e_i, p_{-i} \diamond \sigma_{-i}) \geq u_i(d_i, p_{-i} \diamond \sigma_{-i})$ (because the p_i prove that e_r^* is not eliminable relative to $D_r^1, E_r^2, D_c^1, E_c^2$). But,

[17] We need to make this case explicit for the case $E_{-i} = \Sigma_{-i}$.

the distribution $p_{-i} \diamond \sigma_{-i}$ is a legitimate completion of the partial mixed strategy p'_{-i} as well (albeit one that distributes the remaining probability over multiple strategies), and hence the p'_i prove that e^*_r is not eliminable relative to $D^1_r, E^1_r, D^1_c, E^1_c$. ∎

Next, we show that the Nash equilibrium concept is weaker[18] than our generalized eliminability criterion—in the sense that the generalized criterion can never eliminate a strategy that is in some Nash equilibrium. So, if a strategy can be eliminated by the generalized criterion, it can be eliminated by the Nash equilibrium concept.

Proposition 13 *If there is some Nash equilibrium that places positive probability on pure strategy σ^*_r, then σ^*_r is not eliminable relative to any D_r, E_r, D_c, E_c.*

Proof: Let σ'_r be the row player's (mixed) strategy in the Nash equilibrium (which places positive probability on σ^*_r), and let σ'_c be the column player's (mixed) strategy in the Nash equilibrium. For any D_r, E_r, D_c, E_c with $\sigma^*_r \in E_r$, to prove that σ^*_r is not eliminable relative to these sets, simply let p_r coincide with σ'_r on E_r—that is, let p_r be the probabilities that the row player places on the strategies in E_r in the equilibrium. (Thus, $p_r(\sigma^*_r) > 0$). Similarly, let p_c coincide with σ'_c on E_c. We will prove that the condition on strategies with positive probability is satisfied for the row player; the case of the column player follows by symmetry. For any $e_r \in E_r$ with $p_r(e_r) > 0$, for any mixed strategy d_r, we have $u_r(e_r, \sigma'_c) - u_r(d_r, \sigma'_c) \geq 0$, by the Nash equilibrium condition. Now, let pure strategy $\sigma_c \in \arg\max_{\sigma \in \Sigma_c - E_c}(u_r(e_r, p_c \diamond \sigma) - u_r(d_r, p_c \diamond \sigma))$. Then we must have $u_r(e_r, p_c \diamond \sigma_c) - u_r(d_r, p_c \diamond \sigma_c) \geq u_r(e_r, \sigma'_c) - u_r(d_r, \sigma'_c) \geq 0$ (because $p_c \diamond \sigma_c$ and σ'_c coincide on E_c, and for the former, the remainder of the distribution is chosen to maximize this expression). It follows that σ^*_r is not eliminable relative to any D_r, E_r, D_c, E_c. ∎

We next show that by choosing the sets D_r, E_r, D_c, E_c as large as possible, we can make the generalized eliminability criterion coincide with the Nash equilibrium concept.[19]

Proposition 14 *Let $D_r = E_r = \Sigma_r$ and $D_c = E_c = \Sigma_c$. Then e^*_r is eliminable relative to these sets if and only if there is no Nash equilibrium that places positive probability on e^*_r.*

Proof: The "only if" direction follows from Proposition 13. For the "if" direction, suppose e^*_r is not eliminable relative to $D_r = E_r = \Sigma_r$ and $D_c = E_c = \Sigma_c$. The partial distributions p_r and p_c with $p_r(e^*_r) > 0$ that show that e^*_r is not eliminable must use up all the probability (the probabilities must sum to one), because there are no strategies outside $E_c = \Sigma_c$ and $E_r = \Sigma_r$ to place any remaining probability on. Hence, we must have, for any strategy $e_r \in E_r = \Sigma_r$ with $p_r(e_r) > 0$, that for any mixed strategy d_r, $u_r(e_r, p_c) \geq u_r(d_r, p_c)$ (and the same for the column player). But these are precisely the conditions for p_r and p_c to constitute a Nash equilibrium. It follows that there is a

[18] When discussing elimination of strategies, it is tempting to say that the stronger criterion is the one that can eliminate more strategies. However, when discussing solution concepts, the convention is that the stronger concept is the one that implies the other. Therefore, the criterion that can eliminate fewer strategies is actually the stronger one. For example, strict dominance is stronger than weak dominance, even though weak dominance can eliminate more strategies.

[19] Unlike Nash equilibrium, the generalized eliminability criterion does not discuss what probabilities should be placed on strategies that are not eliminated, so it only "coincides" with Nash equilibrium in terms of what it can eliminate.

Nash equilibrium with positive probability on e_r^*. ■

Moving to the other side of the spectrum, we now show that the concept of strict dominance is stronger than the generalized eliminability criterion—in the sense that the generalized eliminability criterion can always eliminate a strictly dominated strategy (as long as the dominating strategy is in D_r).

Proposition 15 *If pure strategy σ_r^* is strictly dominated by some mixed strategy d_r, then σ_r^* is eliminable relative to any D_r, E_r, D_c, E_c such that 1) $\sigma_r^* \in E_r$, and 2) all the pure strategies on which d_r places positive probability are in D_r.*

Proof: To show that σ_r^* is not eliminable relative to these sets, we must set $p_r(\sigma_r^*) > 0$, and thus we must demonstrate that for some pure strategy $\sigma_c \in \Sigma_c - E_c$, $u_r(\sigma_r^*, p_c \diamond \sigma_c) \geq u_r(d_r, p_c \diamond \sigma_c)$ (or, if all the probability is used up, $u_r(\sigma_r^*, p_c) \geq u_r(d_r, p_c)$), because d_r only places positive probability on strategies in D_r. But this is impossible, because by strict dominance, $u_r(\sigma_r^*, \sigma_c) < u_r(d_r, \sigma_c)$ for any mixed strategy σ_c. ■

Finally, we show that by choosing the sets E_r, E_c as small as possible, we can make the generalized eliminability criterion coincide with the strict dominance concept.

Proposition 16 *Let $E_c = \{\}$ and $E_r = \{e_r\}$. Then e_r is eliminable relative to D_r, E_r, D_c, E_c if and only if it is strictly dominated by some mixed strategy that places positive probability only on elements of D_r.*

Proof: The "if" direction follows from Proposition 15. For the "only if" direction, suppose that e_r is eliminable relative to these sets. That means that there exists a mixed strategy d_r that places positive probability only on strategies in D_r such that for any pure strategy $\sigma_c \in \Sigma_c - E_c = \Sigma_c$, $u(e_r, \sigma_c) < u(d_r, \sigma_c)$ (because $E_c = \{\}$ and $E_r = \{e_r\}$, this is the only way in which an attempt to prove that e_r is not eliminable could fail). But this is precisely the condition for d_r to strictly dominate e_r. ■

We are now ready to turn to computational aspects of the new eliminability criterion.

9.3.4 Applying the new eliminability criterion can be computationally hard

In this subsection, we demonstrate that applying the eliminability criterion can be computationally hard, in the sense of worst-case complexity.[20] We show that applying the eliminability criterion is coNP-complete in two key special cases (subclasses of the problem). The first case is the one in which the D_r, E_r, D_c, E_c sets are set to be as large as possible. Here, the hardness follows directly from Proposition 14 and a result from Section 9.2.

Theorem 102 *Deciding whether a given strategy is eliminable relative to $D_r = E_r = \Sigma_r$ and $D_c = E_c = \Sigma_c$ is coNP-complete, even when the game is symmetric.*

[20]Because we only show hardness in the worst case, it is possible that many (or even most) instances are in fact easy to solve.

Proof: By Proposition 14, this is the converse of asking whether there exists a Nash equilibrium with positive probability on the given strategy. As we saw in Section 9.2, this is NP-complete. ■

While this shows that the eliminability criterion is, in general, computationally hard to apply, we may wonder if there are special cases in which it is computationally easy to apply. Natural special cases to look at include those in which some of the sets D_r, E_r, D_c, E_c are small. The next theorem shows that applying the eliminability criterion remains coNP-complete even when $|D_r| = |D_c| = 1$.

Theorem 103 *Deciding whether a given strategy is eliminable relative to given D_r, E_r, D_c, E_c is coNP-complete, even when $|D_r| = |D_c| = 1$.*

Proof: We will show later (Corollary 28) that the problem is in coNP. To show that the problem is coNP-hard, we reduce an arbitrary KNAPSACK instance (given by m cost-value pairs (c_i, v_i), a cost constraint C and a value target V; we assume without loss of generality that $C = 1 - \epsilon$, for some ϵ small enough that it is impossible for a subset of the c_i to sum to a value strictly between C and 1,[21] that $c_i > 0$ for all i, and that $\sum_{i=1}^{m} v_i \leq 1$) to the following eliminability question. Let the game be as follows. The row player has $m+2$ distinct pure strategies: $e_r^1, e_r^2, \ldots, e_r^m, e_r^*, d_r$ (where $E_r = \{e_r^1, e_r^2, \ldots, e_r^m, e_r^*\}$ and $D_r = \{d_r\}$). The column player has $m + 1$ distinct pure strategies: $e_c^1, e_c^2, \ldots, e_c^m, d_c$ (where $E_c = \{e_c^1, e_c^2, \ldots, e_c^m\}$ and $D_c = \{d_c\}$). Let the utilities be as follows:

- $u_r(e_r^i, e_c^j) = 1$ for all $i \neq j$;
- $u_r(e_r^i, e_c^i) = 1 - \frac{1}{v_i}$ for all i;
- $u_r(e_r^i, d_c) = 1$ for all i;
- $u_r(e_r^*, e_c^i) = \frac{1}{V} - 1$ for all i;
- $u_r(e_r^*, d_c) = -1$;
- $u_r(d_r, e_c^i) = 0$ for all i;
- $u_r(d_r, d_c) = 0$;
- $u_c(e_r^i, e_c^j) = 0$ for all $i \neq j$;
- $u_c(e_r^i, e_c^i) = \frac{1}{c_i}$ for all i;
- $u_c(e_r^i, d_c) = 1$ for all i;
- $u_c(e_r^*, e_c^i) = 0$ for all i;
- $u_c(e_r^*, d_c) = 1$;
- $u_c(d_r, e_c^i) = 0$ for all i;

[21] Because we may assume that the c_i and C are all integers divided by some number K, it is sufficient if $\epsilon < \frac{1}{K}$.

- $u_c(d_r, d_c) = 1$.

Thus, the matrix is as follows:

	e_c^1	e_c^2	$\cdots$	e_c^m	d_c
e_r^1	$1-\frac{1}{v_1}, \frac{1}{c_1}$	$1, 0$	$\cdots$	$1, 0$	$1, 1$
e_r^2	$1, 0$	$1-\frac{1}{v_2}, \frac{1}{c_2}$	$\cdots$	$1, 0$	$1, 1$
$\vdots$					
e_r^m	$1, 0$	$1, 0$	$\cdots$	$1-\frac{1}{v_m}, \frac{1}{c_m}$	$1, 1$
e_r^*	$\frac{1}{V}-1, 0$	$\frac{1}{V}-1, 0$	$\cdots$	$\frac{1}{V}-1, 0$	$-1, 1$
d_r	$0, 0$	$0, 0$	$\cdots$	$0, 0$	$0, 1$

We now show that e_r^* is eliminable relative to D_r, E_r, D_c, E_c if and only if there is no solution to the KNAPSACK instance.

First suppose there is a solution to the KNAPSACK instance. Then, for every i such that (c_i, v_i) is included in the KNAPSACK solution, let $p_r(e_r^i) = c_i$; for every i such that (c_i, v_i) is not included in the KNAPSACK solution, let $p_r(e_r^i) = 0$. Also, let $p_r(e_r^*) = 1 - \sum\limits_{i=1}^{m} p_r(e_r^i)$. (We note that $\sum\limits_{i=1}^{m} p_r(e_r^i) \leq C = 1 - \epsilon$, so that $p_r(e_r^*) \geq \epsilon > 0$.) Also, for every i such that (c_i, v_i) is included in the KNAPSACK solution, let $p_c(e_c^i) = v_i$. We now show that p_r and p_c satisfy the conditions of Definition 58. If the column player places the remaining probability on d_c, then the utility for the row player of playing any e_r^i with $p_r(e_r^i) > 0$ is $1 - \frac{v_i}{v_i} = 0$; the utility of playing e_r^* is $-1 + \frac{1}{V}\sum\limits_{i=1}^{m} p_c(e_c^i) \geq -1 + \frac{V}{V} = 0$; and the utility of playing d_r is also 0. Thus, the condition is satisfied for all elements of E_r that have positive probability. As for E_c, we note that all of the row player's probability has already been used up. The utility of playing any e_c^i with $p_c(e_c^i) > 0$ is $\frac{c_i}{c_i} = 1$, whereas the utility for playing d_c is also 1. Thus, the condition is satisfied for all elements of E_c that have positive probability. It follows that p_r and p_c satisfy the conditions of Definition 58 and e_r^* is not eliminable relative to D_r, E_r, D_c, E_c.

Now suppose that e_r^* is not eliminable relative to D_r, E_r, D_c, E_c. Let p_r and p_c be partial mixed strategies on E_r and E_c satisfying the conditions of Definition 58. We must have that $p_r(e_r^*) > 0$. The utility for the row player of playing e_r^* is $-1 + \frac{1}{V}\sum\limits_{i=1}^{m} p_c(e_c^i)$, which must be at least 0 (the utility of playing d_r); hence $\sum\limits_{i=1}^{m} p_c(e_c^i) \geq V$. The utility for the column player of playing e_c^i is $\frac{p_r(e_r^i)}{c_i}$, which must be at least 1 (the utility of playing d_c) if $p_c(e_c^i) > 0$; hence $p_r(e_r^i) \geq c_i$ if $p_c(e_c^i) > 0$. Finally, the utility for the row player of playing e_r^i is $1 - \frac{p_c(e_c^i)}{v_i}$, which must be at least 0 (the utility of playing d_r) if $p_r(e_r^i) > 0$; hence $p_c(e_c^i) \leq v_i$ if $p_r(e_r^i) > 0$. Because we must have $p_r(e_r^i) \geq c_i > 0$ if $p_c(e_c^i) > 0$, it follows that we must always have $p_c(e_c^i) \leq v_i$. Let $S = \{i : p_c(e_c^i) > 0\}$. We must have $\sum\limits_{i \in S} v_i \geq \sum\limits_{i \in S} p_c(e_c^i) \geq V$. Also, we must have $\sum\limits_{i \in S} c_i \leq \sum\limits_{i \in S} p_r(e_r^i) < 1$ (because we must have $p_r(e_r^*) > 0$). Because it is impossible that $C < \sum\limits_{i \in S} c_i < 1$, it follows that $\sum\limits_{i \in S} c_i \leq C$. But then, S is a solution to the KNAPSACK instance. ■

However, we will show later that the eliminability criterion can be applied in polynomial time if the E_i sets are small (regardless of the size of the D_i sets). To do so, we first need to introduce an alternative version of the definition.

9.3.5 An alternative, equivalent definition of the eliminability criterion

In this subsection, we will give an alternative definition of eliminability, and we will show it is equivalent to the one presented in Definition 58. While the alternative definition is slightly less intuitve than the original one, it is easier to work with computationally, as we will show in the next subsection. Informally, the alternative definition differs from the original one as follows: in the alternative definition, the completion of player $-i$'s mixed strategy has to be chosen *before* player i's strategy d_i is chosen (but after player i's strategy e_i with $p_i(e_i) > 0$ is chosen). The formal definition follows.

Definition 59 *Given a two-player game in normal form, subsets D_r, E_r of the row player's pure strategies Σ_r, subsets D_c, E_c of the column player's pure strategies Σ_c, and a distinguished strategy $e_r^* \in E_r$, we say that e_r^* is* not eliminable relative to D_r, E_r, D_c, E_c, *if there exist functions (partial mixed strategies) $p_r : E_r \to [0,1]$ and $p_c : E_c \to [0,1]$ with $p_r(e_r^*) > 0$, $\sum_{e_r \in E_r} p_r(e_r) \leq 1$, and $\sum_{e_c \in E_c} p_c(e_c) \leq 1$, such that the following holds. For both $i \in \{r, c\}$, for any $e_i \in E_i$ with $p_i(e_i) > 0$, there exists some completion of the probability distribution over $-i$'s strategies, given by $p_{-i}^{e_i} : \Sigma_{-i} \to [0,1]$ (with $p_{-i}^{e_i}(e_{-i}) = p_{-i}(e_{-i})$ for all $e_{-i} \in E_{-i}$, and $\sum_{\sigma_{-i} \in \Sigma_{-i}} p_{-i}^{e_i}(\sigma_{-i}) = 1$), such that for any pure strategy $d_i \in D_i$, we have $u_i(e_i, p_{-i}^{e_i}) \geq u_i(d_i, p_{-i}^{e_i})$.*

We now show that the two definitions are equivalent.

Theorem 104 *The notions of eliminability put forward in Definitions 58 and 59 are equivalent. That is, e_r^* is eliminable relative to D_r, E_r, D_c, E_c according to Definition 58 if and only if e_r^* is eliminable relative to (the same) D_r, E_r, D_c, E_c according to Definition 59.*

Proof: The definitions are identical up to the condition that each strategy with positive probability (each $e_r \in E_r$ with $p_r(e_r) > 0$ and each $e_c \in E_c$ with $p_c(e_c) > 0$) must satisfy. We will show that these conditions are equivalent across the two definitions, thereby showing that the definitions are equivalent.

To show that the conditions are equivalent, we introduce another, zero-sum game that is a function of the original game, the sets D_r, E_r, D_c, E_c, the chosen partial probability distributions p_r and p_c, and the strategy e_i for which we are checking whether the conditions are satisfied. (Without loss of generality, assume that we are checking it for some strategy $e_r \in E_r$ with $p_r(e_r) > 0$.)

The zero-sum game has two players, 1 and 2 (not to be confused with the row and column players of the original game). Player 1 chooses some $d_r \in D_r$, and player 2 chooses some $\sigma_c \in \Sigma_c - E_c$. The utility to player 1 is $u_r(d_r, p_c \diamond \sigma_c) - u_r(e_r, p_c \diamond \sigma_c)$ (and the utility to player 2 is the negative of this). (We assume without loss of generality that p_c does not already use up all the probability, because in this case the conditions are trivially equivalent across the two definitions.)

First, suppose that player 1 must declare her probability distribution (mixed strategy) over D_r first, after which player 2 best-responds. Then, letting $\Delta(X)$ denote the set of probability distributions over set X, player 1 will receive $\max_{\delta_r \in \Delta(D_r)} \min_{\sigma_c \in \Sigma_c - E_c} \sum_{d_r \in D_r} \delta_r(d_r)(u_r(d_r, p_c \diamond \sigma_c) - u_r(e_r, p_c \diamond \sigma_c)) = \max_{\delta_r \in \Delta(D_r)} \min_{\sigma_c \in \Sigma_c - E_c} u_r(\delta_r, p_c \diamond \sigma_c) - u_r(e_r, p_c \diamond \sigma_c)$. This expression is at most 0 if and only if the condition in Definition 58 is satisfied.

Second, suppose that player 2 must declare his probability distribution (mixed strategy) over $\Sigma_c - E_c$ first, after which player 1 best-responds. Then, player 1 will receive $\min_{\delta_c \in \Delta(\Sigma_c - E_c)} \max_{d_r \in D_r} \sum_{\sigma_c \in \Sigma_c - E_c} \delta_c(\sigma_c)(u_r(d_r, p_c \diamond \sigma_c) - u_r(e_r, p_c \diamond \sigma_c)) = \min_{\delta_c \in \Delta(\Sigma_c - E_c)} \max_{d_r \in D_r} \sum_{e_c \in E_c} p_c(e_c)(u_r(d_r, e_c) - u_r(e_r, e_c)) + \sum_{\sigma_c \in \Sigma_c - E_c} (1 - \sum_{e_c \in E_c} p_c(e_c)) \delta_c(\sigma_c)(u_r(d_r, \sigma_c) - u_r(e_r, \sigma_c)) = \min_{\delta_c \in \Delta(\Sigma_c - E_c)} \max_{d_r \in D_r} u_r(d_r, p_c \diamond \delta_c) - u_r(e_r, p_c \diamond \delta_c)$. This expression is at most 0 if and only if the condition in Definition 59 is satisfied.

However, by the Minimax Theorem [von Neumann, 1927], the two expressions must have the same value, and hence the two conditions are equivalent. ■

Informally, the reason that Definition 59 is easier to work with computationally is that all of the continuous variables (the values of the functions $p_r, p_c, p_c^{e_r}, p_r^{e_c}$) are set by the party that is trying to prove that the strategy is not eliminable; whereas in Definition 58, some of the continuous variables (the probabilities defining the mixed strategies d_r, d_c) are set by the party trying to refute the proof that the strategy is not eliminable. This will become more precise in the next subsection.

9.3.6 A mixed integer programming approach

In this subsection, we show how to translate Definition 59 into a mixed integer program that determines whether a given strategy e_r^* is eliminable relative to given sets D_r, E_r, D_c, E_c. The variables in the program, which are all restricted to be nonnegative, are the $p_i(e_i)$ for all $e_i \in E_i$; the $p_i^{e_{-i}}(\sigma_i)$ for all $e_{-i} \in E_{-i}$ and all $\sigma_i \in \Sigma_i - E_i$; and *binary* indicator variables $b_i(e_i)$ for all $e_i \in E_i$ which can be set to zero if and only if $p_i(e_i) = 0$. The program is the following:

maximize $p_r(e_r^*)$ **subject to**

(probability constraints): for both $i \in \{r, c\}$, for all $e_i \in E_i$, $\sum_{e_{-i} \in E_{-i}} p_{-i}(e_{-i}) + \sum_{\sigma_{-i} \in \Sigma_{-i} - E_{-i}} p_{-i}^{e_i}(\sigma_{-i}) = 1$

(binary constraints): for both $i \in \{r, c\}$, for all $e_i \in E_i$, $p_i(e_i) \leq b_i(e_i)$

(main constraints): for both $i \in \{r, c\}$, for all $e_i \in E_i$ and all $d_i \in D_i$, $\sum_{e_{-i} \in E_{-i}} p_{-i}(e_{-i})(u_i(e_i, e_{-i}) - u_i(d_i, e_{-i})) + \sum_{\sigma_{-i} \in \Sigma_{-i} - E_{-i}} p_{-i}^{e_i}(\sigma_{-i})(u_i(e_i, \sigma_{-i}) - u_i(d_i, \sigma_{-i})) \geq (b_i(e_i) - 1)U_i$

In this program, the constant U_i is the maximum difference between two different utilities that player i may receive in the game, that is, $U_i = \max_{\sigma_r, \sigma_r' \in \Sigma_r, \sigma_c, \sigma_c' \in \Sigma_c} u_i(\sigma_r, \sigma_c) - u_i(\sigma_r', \sigma_c')$.

Theorem 105 *The mixed integer program has a solution with objective value greater than zero if and only if e_r^* is not eliminable relative to D_r, E_r, D_c, E_c.*

Proof: For any $e_i \in E_i$ with $p_i(e_i) > 0$, $b_i(e_i)$ must be 1, and thus the corresponding main constraints become: for any $d_i \in D_i$, $\sum_{e_{-i} \in E_{-i}} p_{-i}(e_{-i})(u_i(e_i, e_{-i}) - u_i(d_i, e_{-i})) + \sum_{\sigma_{-i} \in \Sigma_{-i} - E_{-i}} p_{-i}^{e_i}(\sigma_{-i})(u_i(e_i, \sigma_{-i}) - u_i(d_i, \sigma_{-i})) \geq 0$. These are equivalent to the constraints given on strategies $e_i \in E_i$ with $p_i(e_i) > 0$ in Definition 59. On the other hand, for any $e_i \in E_i$ with $p_i(e_i) = 0$, $b_i(e_i)$ can be set to 0, in which case the constraints become: for any $d_i \in D_i$, $\sum_{e_{-i} \in E_{-i}} p_{-i}(e_{-i})(u_i(e_i, e_{-i}) - u_i(d_i, e_{-i})) + \sum_{\sigma_{-i} \in \Sigma_{-i} - E_{-i}} p_{-i}^{e_i}(\sigma_{-i})(u_i(e_i, \sigma_{-i}) - u_i(d_i, \sigma_{-i})) \geq -U_i$. Because the probabilities in each of these constraints must sum to one by the probability constraints, and U_i is the maximum difference between two different utilities that player i may receive in the game, these constraints are vacuous. Therefore the main constraints correspond exactly to those in Definition 59. ■

We obtain the following corollaries:

Corollary 28 *Checking whether a given strategy can be eliminated relative to given D_r, E_r, D_c, E_c is in coNP.*

Proof: To see whether the strategy can be protected from elimination, we can nondeterministically choose the values for the binary variables $b_r(e_r)$ and $b_c(e_c)$. After this, only a linear program remains to be solved, which can be done in polynomial time [Khachiyan, 1979]. ■

Corollary 29 *Using the mixed integer program above, the time required to check whether a given strategy can be eliminated relative to given D_r, E_r, D_c, E_c is exponential only in $|E_r| + |E_c|$ (and not in $|D_r|, |D_c|, |\Sigma_r|$, or $|\Sigma_c|$).*

Proof: Any mixed integer program whose only integer variables are binary variables can be solved in time exponential only in its number of binary variables (for example, by searching over all settings of its binary variables and solving the remaining linear program in each case). The number of binary variables in this program is $|E_r| + |E_c|$. ■

9.3.7 Iterated elimination

In this subsection, we study what happens when we eliminate strategies *iteratively* using the new criterion. The criterion can be iteratively applied by removing an eliminated strategy from the game, and subsequently checking for new eliminabilities in the game with the strategy removed, *etc.* (as in the more elementary, conventional notion of iterated dominance). First, we show that this procedure is, in a sense, sound.

Theorem 106 *Iterated elimination according to the generalized criterion will never remove a strategy that is played with positive probability in some Nash equilibrium of the original game.*

Proof: We will prove this by induction on the elimination round (that is, the number of strategies eliminated so far). The claim is true for the first round by Proposition 13. Now suppose it is true up to and including round k; we must show it is true for round $k+1$. Suppose that the claim is false for round $k+1$, that is, there exists some game G and some pure strategy σ such that 1) σ is played with positive probability in some Nash equilibrium of G, and 2) using k elimination rounds, G can be reduced to G^{k+1}, in which σ is eliminable. Now consider the game G^k which preceded G^{k+1} in the elimination sequence, that is, the game obtained by undoing the last elimination before G^{k+1}. Also, let σ' be the strategy removed from G^k to obtain G^{k+1}. Now, in G^k, σ cannot be eliminated by the induction assumption. However, by Proposition 14, any strategy that is not played with positive probability in any Nash equilibrium can be eliminated, so it follows that there is some Nash equilibrium of G^k in which σ is played with positive probability. Moreover, this Nash equilibrium cannot place positive probability on σ' (because otherwise, by Proposition 13, we would not be able to eliminate it). But then, this Nash equilibrium must also be a Nash equilibrium of G^{k+1}: it does not place any probability on strategies that are not in G^{k+1}, and the set of strategies that the players can switch to in G^{k+1} is a subset of those in G^k. Hence, by Proposition 13, we cannot eliminate σ from G^{k+1}, and we have achieved the desired contradiction. ■

Because (the single-round version of) the eliminability criterion extends all the way to Nash equilibrium by Proposition 14, we get the following corollary.

Corollary 30 *Any strategy that can be eliminated using iterated elimination can also be eliminated in a single round (that is, without iterated application of the criterion).*

Proof: By Proposition 14, all strategies that are not played with positive probability in any Nash equilibrium can be eliminated in a single round; but by Theorem 106, this is the only type of strategy that iterated elimination can eliminate. ■

Interestingly, iterated elimination is in a sense incomplete:

Proposition 17 *Removing an eliminated strategy from a game sometimes decreases the set of strategies that can be eliminated.*

Proof: Consider the following game:

	L	M	R
U	2, 2	0, 1	0, 5
D	1, 0	1, 1	1, 0

The unique Nash equilibrium of this game is (D, M), for the following reasons. In order for it to be worthwhile for the row player to play U with positive probability, the column player should play L with probability at least $1/2$. But, in order for it to be worthwhile for the column player to play L with positive probability (rather than M), the row player should play U with probability at least $1/2$. However, if the row player plays U with probability at least $1/2$, then the column player's unique best response is to play R. Hence, the row player must play D in any Nash equilibrium, and the unique best response to D is M.

Thus, by Proposition 14, all strategies besides D and M can be eliminated. In particular, R can be eliminated. However, if we remove R from the game, the remaining game is:

	L	M
U	2, 2	0, 1
D	1, 0	1, 1

In this game, (U, L) is also a Nash equilibrium, and hence U and L can no longer be eliminated, by Proposition 13. ■

This example highlights an interesting issue with respect to using this eliminability criterion as a preprocessing step in the computation of Nash equilibria: it does not suffice to simply throw out eliminated strategies and compute a Nash equilibrium for the remaining game. Rather, we need to use the criterion more carefully: if we know that a strategy is eliminable according to the criterion we can restrict our attention to supports for the player that do not include this strategy.

The example also directly implies that iterated elimination according to the generalized criterion is path-dependent (the choice of which strategy to remove first affects which strategies can/will be removed later). As we discussed in Section 9.1, the same phenomenon occurs with iterated weak dominance. There is a sizeable literature on path (in)dependence for various notions of dominance [Gilboa *et al.*, 1990; Borgers, 1993; Osborne and Rubinstein, 1994; Marx and Swinkels, 1997, 2000; Apt, 2004].

In light of these results, it may appear that there is not much reason to do iterated elimination using the new criterion, because it never increases and sometimes even decreases the set of strategies that we can eliminate. However, we need to keep in mind that Theorem 106, Corollary 30, and Proposition 17 do not pose any restrictions on the sets D_r, E_r, D_c, E_c, and therefore (by Propositions 13 and 14) are effectively results about iteratively removing strategies based on whether they are played in a Nash equilibrium. However, the new criterion is more informative and useful when there are restrictions on the sets D_r, E_r, D_c, E_c. Of particular interest is the restriction $|E_r| + |E_c| \leq k$, because by Corollary 29 this quantity determines the (worst-case) runtime of the mixed integer programming approach that we presented in the previous subsection. Under this restriction, it turns out that iterated elimination can eliminate strategies that single-round elimination cannot.

Proposition 18 *Under a restriction of the form $|E_r| + |E_c| \leq k$, iterated elimination can eliminate strategies that single-round elimination cannot (even when $k = 1$).*

Proof: By Proposition 16, when $k = 1$ the eliminability criterion coincides with strict dominance (and hence iterated application of the criterion coincides with iterated strict dominance). So, consider the following game:

	L	R
U	1, 0	1, 1
D	0, 1	0, 0

Strict dominance cannot eliminate L, but iterated strict dominance (which can remove D first) can eliminate L. ■

Of course, even under this (or any other) restriction iterated elimination remains sound in the sense of Theorem 106. Therefore, one sensible approach to eliminating strategies is the following. Iteratively apply the eliminability criterion (with whatever restrictions are desired to increase the strength of the argument, or are necessary to make it computationally manageable, such as $|E_r| + |E_c| \leq k$), removing each eliminated strategy, until the process gets stuck. Then, start again with the original game, and take a different path of iterated elimination (which may eliminate strategies that could no longer be eliminated after the first path of elimination, as described in Proposition 17), until the process gets stuck—*etc.* In the end, any strategy that was eliminated in any one of the elimination paths can be considered "eliminated", and this is safe by Theorem 106.[22]

Interestingly, here the analogy with iterated weak dominance breaks down. Because there is no soundness theorem such as Theorem 106 for iterated weak dominance, considering all the strategies that are eliminated in some iterated weak dominance elimination path to be simultaneously "eliminated" can lead to senseless results. Consider for example the following game:

	L	M	R
U	1, 1	0, 0	1, 0
D	1, 1	1, 0	0, 0

U can be eliminated by removing R first, and D can be eliminated by removing M first—but these are the row player's only strategies, so considering both of them to be eliminated makes little sense.

9.4 Summary

A theory of mechanism design for bounded agents cannot rest on game-theoretic solution concepts that are too hard for agents to compute. To assess to what extent this eliminates existing solution concepts from consideration, the first two sections of this chapter were devoted to studying how hard it is to compute solutions according to some of these concepts.

In Section 9.1, we studied computational aspects of dominance and iterated dominance. We showed that checking whether a given strategy is dominated (weakly or strictly) by some mixed strategy can be done in polynomial time using a single linear program solve. We then showed that determining whether there is some path that eliminates a given strategy is NP-complete with iterated weak dominance. This allowed us to also show that determining whether there is a path that leads to a unique solution is NP-complete. Both of these results hold both with and without dominance by mixed strategies. Iterated strict dominance, on the other hand, is path-independent (both with and without dominance by mixed strategies) and can therefore be done in polynomial time. We then studied what happens when the dominating strategy is allowed to place positive probability on only a few pure strategies. First, we showed that finding the dominating strategy with minimum support size is NP-complete (both for strict and weak dominance). Then, we showed that iterated strict

[22] This procedure is reminiscent of iterative sampling.

dominance becomes path-dependent when there is a limit on the support size of the dominating strategies, and that deciding whether a given strategy can be eliminated by iterated strict dominance under this restriction is NP-complete (even when the limit on the support size is 3). We also studied dominance and iterated dominance in Bayesian games. We showed that, unlike in normal-form games, deciding whether a given pure strategy is dominated by another pure strategy in a Bayesian game is NP-complete (both with strict and weak dominance); however, deciding whether a strategy is dominated by some mixed strategy can still be done in polynomial time with a single linear program solve (both with strict and weak dominance). Finally, we showed that iterated dominance using pure strategies can require an exponential number of iterations in a Bayesian game (both with strict and weak dominance).

In Section 9.2 we provided a single reduction that demonstrates that 1) it is NP-complete to determine whether Nash equilibria with certain natural properties exist, 2) more significantly, the problems of maximizing certain properties of a Nash equilibrium are inapproximable (unless ZPP=NP), and 3) it is #P-hard to count the Nash equilibria (or connected sets of Nash equilibria). We also showed that determining whether a pure-strategy Bayes-Nash equilibrium exists is NP-complete.

Since these (and other) results suggest that dominance is a more tractable solution concept than (Bayes)-Nash equilibrium, but is often too strict for mechanism design (and other) purposes, one may wonder whether it is possible to strike a compromise between dominance and Nash equilibrium, obtaining intermediate solution concepts that combine good aspects of both. The last section in this chapter, Section 9.3, did precisely that. We defined a generalized eliminability criterion for bimatrix games that considers whether a given strategy is eliminable relative to given dominator & eliminee subsets of the players' strategies. We showed that this definition spans a spectrum of eliminability criteria from strict dominance (when the subsets are as small as possible) to Nash equilibrium (when the subsets are as large as possible). We showed that checking whether a strategy is eliminable according to this criterion is coNP-complete (both when all the sets are as large as possible and when the dominator sets each have size 1). We then gave an alternative definition of the eliminability criterion and showed that it is equivalent using the Minimax Theorem. We showed how this alternative definition can be translated into a mixed integer program of polynomial size with a number of (binary) integer variables equal to the sum of the sizes of the eliminee sets, implying that checking whether a strategy is eliminable according to the criterion can be done in polynomial time if the eliminee sets are small. Finally, we studied using the criterion for iterated elimination of strategies.

The results in this chapter provide an initial step towards building a theory of mechanism design for bounded agents. For such a theory to be complete, it would also require methods for predicting how agents will act in strategic situations where standard game-theoretic solutions are too hard for them to compute. Ideally, these methods would not assume any detailed knowlege of the algorithms available to the agents, but this will undoubtedly be a difficult feat to accomplish. Fortunately, we do not have to wait for the entire theory of mechanism design for bounded agents to be developed before we create some initial techniques for designing such mechanisms nonetheless. (The only downside of not having the general theory is that we will not be able to evaluate how close to optimal these techniques are.) The next chapter provides one such technique, which can in fact also be used to generate mechanisms automatically (albeit in a very different way from that proposed in Chapter 6).

Chapter 10

Automated Mechanism Design for Bounded Agents

A long-term goal of this research is to combine automated mechanism design and mechanism design for bounded agents, so that mechanisms that take advantage of the agents' limited computational capacities are automatically designed. However, the approaches that we have seen in previous chapters cannot be straightforwardly combined to achieve this, for several reasons.

As we saw in Chapter 6, the work on automated mechanism design (ours as well as others') so far has restricted itself to producing truthful mechanisms, and appealed to the revelation principle to justify this. The advantage of this restriction is that it is easy to evaluate the quality of a truthful mechanism, because the agents' behavior is perfectly predictable (they will tell the truth). This is what allows us to come up with a clear formulation of the optimization problem.

However, settings in which the bounded rationality of agents can be exploited are necessarily ones in which the revelation principle does *not* meaningfully apply. Specifically, a mechanism that is truthful (in the sense that no manipulation can be beneficial) can never exploit the agents' bounded rationality, because in such a mechanism agents would reveal the truth regardless of their computational sophistication. It follows that we must extend the search space for automated mechanism design to include non-truthful mechanisms.[1] Finding an optimal mechanism in this extended search space would require us to have some method for evaluating the quality of non-truthful mechanisms, which needs to be based on a model of how the agents behave in non-truthful mechanisms. Moreover, this model *must* take into account the agents' bounded rationality: if we assume that the agents will play optimally (in a game-theoretic sense), then by the revelation principle there is still never a reason to prefer non-truthful mechanisms over truthful mechanisms.[2]

[1] Another aspect of mechanism design for bounded agents is that we may not wish to immediately ask each agent to provide all of its preferences, as these may be difficult for the agent to compute. Rather, we could consider *multistage* mechanisms that selectively query the agent only for the needed preferences. These mechanisms can still be truthful in the sense that it is always strategically optimal to answer queries truthfully. However, as explained before, the efficient elicitation of agents' preferences is a topic that is orthogonal to this dissertation.

[2] Some non-truthful mechanisms may be just as good as truthful mechanisms when agents behave in a game-theoretically optimal way, so it is possible that our search would fortuitously return a non-truthful mechanism. However, if this happens, there is still no guarantee that this non-truthful mechanism will perform better than the best truthful mechanism when agents are actually bounded, for one of two reasons. First, it may be easy to act optimally in this particular

Creating a good model of the behavior of boundedly rational agents for this purpose is by no means easy. For one, it is difficult to guarantee that agents will not adapt their reasoning algorithms to the specific mechanism that they face. The results on hardness of manipulation in Chapter 8 avoided this difficulty, by using the standard complexity-theoretic approach of showing that there are infinite *families* of instances that are hard. This was necessary because in standard formulations of complexity theory, any *individual* instance of a problem is easy to solve, for example by the algorithm that has the solution to that particular instance precomputed. Unfortunately, the purpose of automated mechanism design is precisely to design a mechanism *for the instance at hand* only! Thus we cannot refer to hardness over infinite classes of instances for this purpose.

These difficulties prevent us from formulating automated mechanism design for bounded agents as a clean optimization problem. Nevertheless, in Section 10.1, we do propose a more heuristic approach by which mechanisms for bounded agents can be designed automatically [Conitzer and Sandholm, 2006e]. In light of the issues discussed above, it should not come as a surprise that this approach is very different from the approaches described earlier in this dissertation. Rather than optimizing the entire mechanism in a single step (as in Chapter 6), the idea is to incrementally make the mechanism *more* strategy-proof over time, by finding potential beneficial manipulations, and changing outcomes locally so that these manipulations are no longer beneficial. Computationally, this is a much more scalable approach. The mechanism may eventually become (completely) strategy-proof, in which case it will not take advantage of agents' bounded rationality; however, it may be that some manipulations remain in the end. Intuitively, one should expect such remaining manipulations to be more difficult to discover for the agents, as the algorithm for designing the mechanism has not discovered them yet. Thus, instead of using a complexity-theoretic argument as in Chapter 8, here the argument for hardness of manipulation depends on the agents not being able to "outcompute" the designer. (Nevertheless, we will also give a complexity-theoretic argument.)

10.1 Incrementally making mechanisms more strategy-proof

In the approach that we propose in this section, we start with a naïvely designed mechanism that is not strategy-proof (for example, the mechanism that would be optimal in the absence of strategic behavior), and we attempt to make it *more* strategy-proof. Specifically, the approach systematically identifies situations in which an agent has an incentive to manipulate, and corrects the mechanism locally to take away this incentive. This is done iteratively, and the mechanism may or may not become (completely) strategy-proof eventually.

One can conceive of this as being a new approach to automated mechanism design, insofar as the updates to the mechanism to make it more strategy-proof can be executed automatically (by a computer). Indeed, we will provide algorithms for doing so. (These algorithms are computationally much more efficient than the optimization algorithms proposed in Chapter 6, because to ensure strategy-proofness, those algorithms had to simultaneously decide on the outcome that the mechanism chooses *for every possible input* of revealed preferences, and the strategy-proofness constraints interrelated these decisions.) It is also possible to think about the results of this approach theoretically, and use them as a guide in more "traditional" mechanism design. We pursue this as well,

non-truthful mechanism. Second (worse), it may the case that it is not easy, and that this difficulty actually leads the agents to act in such a way that *worse* outcomes are obtained.

giving various examples. Finally, we will argue that if the mechanism that the approach produces remains manipulable, then any remaining manipulations will be computationally hard to find.

This approach bears some similarity to how mechanisms are designed in the real world. Real-world mechanisms are often initially naïve, leading to undesirable strategic behavior; once this is recognized, the mechanism is somehow amended to disincent the undesirable behavior. For example, some naïvely designed mechanisms give bidders incentives to postpone submitting their bids until just before the event closes (*i.e.*, sniping); often this is (partially) fixed by adding an *activity rule*, which prevents bidders that do not bid actively early from winning later. As another example, in the 2003 Trading Agent Competition Supply Chain Management (TAC/SCM) game, the rules of the game led the agents to procure most of their components on day 0. This was deemed undesirable, and the designers tried to modify the rules for the 2004 competition to disincent this behavior [Kiekintveld *et al.*, 2005].[3]

As we will see, there are many variants of the approach, each with its own merits. We will not decide which variant is the best in this dissertation; rather, we will show for a few different variants that they can result in desirable mechanisms.

10.1.1 Definitions

In this chapter, we will consider payments (if they are possible) to be part of the outcome. Because of this, we can identify a mechanism with its outcome selection function. Given a mechanism $M : \Theta \rightarrow O$ mapping type vectors to outcomes, a *beneficial manipulation*[4] consists of an agent i, a type vector $\langle \theta_1, \ldots, \theta_n \rangle \in \Theta$, and an alternative type report $\hat{\theta}_i$ for agent i such that $u_i(\theta_i, M(\langle \theta_1, \ldots, \theta_n \rangle)) < u_i(\theta_i, M(\langle \theta_1, \ldots, \theta_{i-1}, \hat{\theta}_i, \theta_{i+1}, \ldots, \theta_n \rangle))$. In this case we say that i manipulates *from* $\langle \theta_1, \ldots, \theta_n \rangle$ *into* $\langle \theta_1, \ldots, \theta_{i-1}, \hat{\theta}_i, \theta_{i+1}, \ldots, \theta_n \rangle$. We note that a mechanism is strategy-proof or (dominant-strategies) incentive compatible if and only if there are no beneficial manipulations. (We will not consider Bayes-Nash equilibrium incentive compatibility in this chapter.)

In settings with payments, we will enforce an *ex-post individual rationality* constraint: we should not make an agent worse off than he would have been if he had not participated in the mechanism. That is, we cannot charge an agent more than he reported the outcome (disregarding payments) was worth to him.

10.1.2 Our approach and techniques

In this subsection, we explain the approach and techniques that we consider in this chapter. We recall that our goal is not to (immediately) design a strategy-proof mechanism; rather, we start with some manipulable mechanism, and attempt to incrementally make it "more" strategy-proof. Thus, the basic template of our approach is as follows:

1. Start with some (manipulable) mechanism M;

[3] Interestingly, these *ad-hoc* modifications failed to prevent the behavior, and even an extreme modification during the 2004 competition failed. Later research suggests that in fact all reasonable settings for a key parameter would have failed [Vorobeychik *et al.*, 2006].

[4] "Beneficial" here means beneficial to the manipulating agent.

2. Find some set F of manipulations (where a manipulation is given by an agent i, a type vector $\langle \theta_1, \ldots, \theta_n \rangle$, and an alternative type report $\hat{\theta}_i$ for agent i);

3. If possible, change the mechanism M to prevent (many of) these manipulations from being beneficial;

4. Repeat from step 2 until termination.

This is merely a template; at each one of the steps, something remains to be filled in. Which initial mechanism do we choose in step 1? Which set of manipulations do we consider in step 2? How do we “fix” the mechanism in step 3 to prevent these manipulations? And how do we decide to terminate in step 4? In this dissertation, we will not resolve what is the best way to fill in these blanks (it seems unlikely that there is a single, universal best way), but rather we will provide a few instantiations of the technique, illustrate them with examples, and show some interesting properties.

One natural way of instantiating step 1 is to choose a *naïvely optimal* mechanism, that is, a mechanism that would give the highest objective value for each type vector *if* every agent would always reveal his type truthfully. For instance, if we wish to maximize social welfare, we simply always choose an outcome that maximizes social welfare for the reported types; if we wish to maximize revenue, we choose an outcome that maximizes social welfare for the reported types, and make each agent pay his entire valuation.

In step 2, there are many possible options: we can choose the set of *all* manipulations; the set of all manipulations for a single agent; the set of all manipulations from or to a particular type or type vector; or just a single manipulation. Which option we choose will affect the difficulty of step 3.

Step 3 is the most complex step. Let us first consider the case where we are only trying to prevent a single manipulation, from $\theta = \langle \theta_1, \ldots, \theta_n \rangle$ to $\theta' = \langle \theta_1, \ldots, \theta_{i-1}, \hat{\theta}_i, \theta_{i+1}, \ldots, \theta_n \rangle$. We can make this manipulation undesirable in one of three ways: **(a)** make the outcome that M selects for θ more desirable for agent i (when he has type θ_i), **(b)** make the outcome that M selects for θ' less desirable for agent i (when he has type θ_i), or **(c)** a combination of the two. For the most part, we will focus on **(a)** in this chapter. There may be multiple ways to make the outcome that M selects for θ sufficiently desirable to prevent the manipulation; a natural way to select from among these outcomes is to choose the one that maximizes the designer's original objective. (Note that any one of these modifications may introduce other beneficial manipulations.)

When we are trying to prevent a set of manipulations, we are confronted with an additional problem: after we have prevented one manipulation in the set, we may reintroduce the incentive for this manipulation when we try to prevent another manipulation. As a simple example, suppose that we are selling a single item to a single bidder, who may value the item at 1, 2, or 3. Suppose that we start with the (naïve) mechanism in which we always sell the item to the bidder at the bid that he places (if he bids $x \in \{1, 2, 3\}$, we sell the item to him at price $\pi(x) = x$). Of course, the bidder has an incentive to shade his valuation. Now suppose that (for some reason) in step 2 we choose the set of the following two beneficial manipulations: report 2 when the true value is 3, and report 1 when the true value is 2. Also suppose that we take approach **(a)** above (for a type vector from which there is a beneficial manipulation, make its outcome more desirable to the manipulating agent). We can fix the first manipulation by setting $\pi(3) = 2$, but then if we fix the second manipulation by setting $\pi(2) = 1$, the incentive to report 2 when the true value is 3 returns. This can be prevented by updating all of the type vectors simultaneously in such a way that none of the manipulations remain beneficial: in the example, this would lead us to find that we should set $\pi(3) = \pi(2) = 1$. However, in general settings, this may require solving a potentially large constrained optimization problem, which would constitute an approach similar to standard automated mechanism design—reintroducing some of the scalability problems that we wish to avoid.[5] Instead, we will be less ambitious: when addressing the manipulations from one type vector, we will act as if we will not change the outcomes for any other type vector. Thus, in the example above, we will indeed choose $\pi(3) = 2$, $\pi(2) = 1$. (Of course, if we had included the manipulation of reporting 1 when the true value is 3, we would set $\pi(3) = \pi(2) = 1$, and there would be no problem. This is effectively what will happen in some of the examples that we will give later.)

Formally, for this particular instantiation of our approach, if M is the mechanism at the beginning of the iteration and M' is the mechanism at the end of the iteration (after the update), and F is the set of manipulations under consideration, we have $M'(\theta) \in \arg\max_{o \in O(M,\theta,F)} g(\theta, o)$ (here, $\theta = \langle \theta_1, \ldots, \theta_n \rangle$), where $O(M, \theta, F) \subseteq O$ is the set of all outcomes o such that for any beneficial manipulation $(i, \hat{\theta}_i)$ (with $(i, \theta, \hat{\theta}_i) \in F$), $u_i(\theta_i, o) \geq u_i(\theta_i, M(\langle \theta_1, \ldots, \theta_{i-1}, \hat{\theta}_i, \theta_{i+1}, \ldots, \theta_n \rangle))$. It may happen that $O(M, \theta, F) = \emptyset$ (no outcome will prevent all manipulations). In this case, there are various ways in which we can proceed. One is not to update the outcome at all, *i.e.* set $M'(\theta) = M(\theta)$. Another is to minimize the number of agents that will have an incentive to manipulate from θ after the change, that is, to choose $M'(\theta) \in \arg\min_{o \in O} |\{i : (\exists (i, \theta, \hat{\theta}_i) \in F : u_i(\theta_i, o) < u_i(\theta_i, M(\langle \theta_1, \ldots, \theta_{i-1}, \hat{\theta}_i, \theta_{i+1}, \ldots, \theta_n \rangle)))\}|$ (and ties can be broken to maximize the objective g).

Many other variants are possible. For example, instead of choosing from the set of all possible outcomes O when we update the outcome of the mechanism for some type vector θ, we can limit ourselves to the set of all outcomes that would result from some beneficial manipulation in F from θ—that is, the set $\{o \in O : ((\exists (i, \hat{\theta}_i) : (i, \theta, \hat{\theta}_i) \in F) : o = M(\langle \theta_1, \ldots, \theta_{i-1}, \hat{\theta}_i, \theta_{i+1}, \ldots, \theta_n \rangle))\}$—in addition to the current outcome $M(\theta)$. The motivation for this is that rather than have to consider all possible outcomes every time, we may wish to simplify our job by considering only the ones that cause the failure of strategy-proofness in the first place. (We may, however, get better results by considering all outcomes.)

In the last few paragraphs, we have been focusing on approach **(a)** above (for a type vector from which there is a beneficial manipulation, make its outcome more desirable to the manipulating agent); approach **(b)** (for a type vector *into* which there is a beneficial manipulation, make its outcome *less* desirable to the manipulating agent) can be instantiated with similar techniques. For example, we can redefine $O(M, \theta, F) \subseteq O$ as the set of all outcomes o such that for any manipulation in F *into* θ, choosing $M'(\theta) = o$ prevents this manipulation from being beneficial.

Next, we present examples of all of the above-mentioned variants.

10.1.3 Instantiating the methodology

In this subsection, we illustrate the potential benefits of the approach by exhibiting mechanisms that it can produce in various standard mechanism design settings. We will demonstrate settings in

[5] Although, if the set of manipulations that we are considering is small, this approach may still scale better than standard automated mechanism design (in which *all* manipulations are considered simultaneously).

which the approach ends up producing strategy-proof mechanisms, as well as a setting in which the produced mechanism is still vulnerable to manipulation (but in some sense "more" strategy-proof than naïve mechanisms). We emphasize that our goal in this subsection is not necessarily to come up with spectacularly novel mechanisms, but rather to show that the approach advocated in this chapter produces sensible results. Therefore, for now, we will consider the approach successful if it produces a well-known mechanism. In future research, we hope to use the technique to help us design novel mechanisms as well.

Deriving the VCG mechanism

In this subsubsection, we show the following result: in general preference aggregation settings in which the agents can make payments (*e.g.* combinatorial auctions), (one variant of) our technique yields the VCG mechanism after a single iteration. We recall from Chapter 4 that the VCG mechanism chooses an outcome that maximizes social welfare (not counting payments), and imposes the following tax on an agent: consider the total utility (not counting payments) of the other agents given the chosen outcome, and subtract this from the total utility (not counting payments) that the other agents *would have obtained* if the given agent's preferences had been ignored in choosing the outcome. Specifically, we will consider the following variant of our technique (perhaps the most basic one):

- Our objective g is to try maximize some (say, linear) combination of allocative social welfare (*i.e.* social welfare not taking payments into account) and revenue. (It does not matter what the combination is.)
- The set F of manipulations that we consider is that of all possible misreports (by any single agent).
- We try to prevent manipulations according to **(a)** above (for a type vector from which there is a beneficial manipulation, make its outcome desirable enough to the manipulating agents to prevent the manipulation). Among outcomes that achieve this, we choose one maximizing the objective function g.

Because we consider payments part of the outcome in this section, we will use the term "allocation" to refer to the part of the outcome that does not concern payments, even though the result is not restricted to allocation settings such as auctions. Also, we will refer to the utility that agent i with type θ_i gets from allocation s (not including payments) as $u_i(\theta_i, s)$. The following simple observation shows that the naïvely optimal mechanism is the *first-price* mechanism, which chooses an allocation that maximizes social welfare, and makes every agent pay his valuation for the allocation.

Observation 2 *The first-price mechanism naïvely maximizes both revenue and allocative social welfare.*

Proof: That the mechanism (naïvely) maximizes allocative social welfare is clear. Moreover, due to the individual rationality constraint, we can never extract more than the allocative social welfare; and the first-price mechanism (naïvely) extracts all the allocative social welfare, for an outcome that (naïvely) maximizes allocative social welfare. ■

Before we show the main result of this subsubsection, we first characterize optimal manipulations for the agents under the first-price mechanism.

Lemma 22 *The following is an optimal manipulation $\hat{\theta}_i$ from $\theta \in \Theta$ for agent i under the first-price mechanism:*

- *for the allocation s^* that would be chosen under the first-price mechanism for θ, report a value equal to i's VCG payment under the true valuations ($u(\hat{\theta}_i(s^*)) = VCG_i(\theta_i, \theta_{-i})$);*
- *for any other allocation $s \neq s^*$, report a valuation of 0.*[6]

The utility of this manipulation is $u(\theta_i, s^*) - VCG_i(\theta_i, \theta_{-i})$. (This assumes ties will be broken in favor of allocation s^*.)

Without the tie-breaking assumption, the lemma does not hold: for example, in a single-item first-price auction, bidding exactly the second price for the item is not an optimal manipulation for the bidder with the highest valuation if the tie is broken in favor of the other bidder. However, increasing the bid by any amount will guarantee that the item is won (and in general, increasing the value for s^* by any amount will guarantee that outcome).

Proof: First, we show that this manipulation will still result in s^* being chosen. Suppose that allocation $s \neq s^*$ is chosen instead. Given the tie-breaking assumption, it follows that $\sum\limits_{j \neq i} u_j(\theta_j, s) > u_i(\hat{\theta}_i, s^*) + \sum\limits_{j \neq i} u_j(\theta_j, s^*)$, or equivalently, $VCG_i(\theta_i, \theta_{-i}) < \sum\limits_{j \neq i} u_j(\theta_j, s) - u_j(\theta_j, s^*)$. However, by definition, $VCG_i(\theta_i, \theta_{-i}) = \max_{s^{**}} \sum\limits_{j \neq i} u_j(\theta_j, s^{**}) - u_j(\theta_j, s^*) \geq \sum\limits_{j \neq i} u_j(\theta_j, s) - u_j(\theta_j, s^*)$, so we have the desired contradiction. It follows that agent i's utility under the manipulation is $u_i(\theta_i, s^*) - VCG_i(\theta_i, \theta_{-i})$.

Next, we show that agent i cannot obtain a higher utility with any other manipulation. Suppose that manipulation $\hat{\theta}_i$ results in allocation s being chosen. Because utilities cannot be negative under truthful reporting, it follows that $u_i(\hat{\theta}_i, s) + \sum\limits_{j \neq i} u_j(\theta_j, s) \geq \max_{s^{**}} \sum\limits_{j \neq i} u_j(\theta_j, s^{**})$. Using the fact that $VCG_i(\theta_i, \theta_{-i}) = \max_{s^{**}} \sum\limits_{j \neq i} u_j(\theta_j, s^{**}) - u_j(\theta_j, s^*)$, we can rewrite the previous inequality as $u_i(\hat{\theta}_i, s) + \sum\limits_{j \neq i} u_j(\theta_j, s) \geq VCG_i(\theta_i, \theta_{-i}) + \sum\limits_{j \neq i} u_j(\theta_j, s^*)$, or equivalently $u_i(\hat{\theta}_i, s) \geq VCG_i(\theta_i, \theta_{-i}) + \sum\limits_{j \neq i} u_j(\theta_j, s^*) - u_j(\theta_j, s)$. Because $\sum\limits_{j} u_j(\theta_j, s^*) \geq \sum\limits_{j} u_j(\theta_j, s)$, we can rewrite the previous inequality as $u_i(\hat{\theta}_i, s) \geq VCG_i(\theta_i, \theta_{-i}) - u_i(\theta_i, s^*) + u_i(\theta_i, s) + \sum\limits_{j} u_j(\theta_j, s^*) - u_j(\theta_j, s) \geq VCG_i(\theta_i, \theta_{-i}) - u_i(\theta_i, s^*) + u_i(\theta_i, s)$, or equivalently, $u_i(\theta_i, s) - u_i(\hat{\theta}_i, s) \leq u_i(\theta_i, s^*) - VCG_i(\theta_i, \theta_{-i})$, as was to be shown. ■

[6]There may be constraints on the reported utility function that prevent this—for example, in a (combinatorial) auction, perhaps only monotone valuations are allowed (winning more items never hurts an agent). If so, the agent should report valuations for these outcomes that are as small as possible, which will still lead to s^* being chosen.

Theorem 107 *Under the variant of our approach described above, the mechanism resulting after a single iteration is the VCG mechanism.*

Proof: By Observation 2, the naïvely optimal mechanism is the first-price mechanism. When updating the outcome for θ, by Lemma 22, each agent i must receive a utility of at least $u_i(\theta_i, s^*) - VCG_i(\theta_i, \theta_{-i})$, where s^* is the allocation that maximizes allocative social welfare for type vector θ. One way of achieving this is to choose allocation s^*, and to charge agent i exactly $VCG_i(\theta_i, \theta_{-i})$—that is, simply run the VCG mechanism. Clearly this maximizes allocative social welfare. But, under the constraints on the agents' utilities, it also maximizes revenue, for the following reason. For any allocation s, the most revenue that we can hope to extract is the allocative social welfare of s, that is, $\sum_i u_i(\theta_i, s)$, minus the sum of the utilities that we must guarantee the agents, that is, $\sum_i u_i(\theta_i, s^*) - VCG_i(\theta_i, \theta_{-i})$. Because $s = s^*$ maximizes $\sum_i u_i(\theta_i, s)$, this means that the most revenue we can hope to extract is $\sum_i VCG_i(\theta_i, \theta_{-i})$, and the VCG mechanism achieves this. ■

Deriving an equal cost sharing mechanism for a nonexcludable public good

We now return to the problem of designing a mechanism for deciding on whether or not a single nonexcludable public good is built. (We studied the automated design of such mechanisms in Chapter 6, Subsection 6.5.2.) Specifically, every agent i has a type v_i (the value of the good to him), and the mechanism decides whether the good is produced, as well as each agent's payment π_i. If the good is produced, we must have $\sum_i \pi_i \geq c$, where c is the cost of producing the good. An individual rationality constraint applies: for each i, $\pi_i \leq \hat{v}_i$ (nobody pays more than his reported value for the good).

In this subsection, rather than use a single variant of our approach, we actually use two distinct phases. Throughout, our objective is to maximize the efficiency of the decision (the good should be produced if and only if the total utility that it generates exceeds the cost of the good); as a secondary objective, we try to maximize revenue.[7] Thus, the naïvely optimal mechanism is to produce the good if and only if the sum of the reported valuations exceeds c, and if so, to charge every agent his entire reported valuation.

An iteration in the first phase proceeds almost exactly as in the previous subsubsection. We try to prevent manipulations according to **(a)** above: for a type vector from which there are beneficial manipulations, make its outcome desirable enough to the manipulating agents to prevent the manipulations, and among outcomes that achieve this, choose the one that maximizes our objective. If there is no such outcome, then we do not change the outcome selected for this type vector. However, in each iteration, we consider only a limited set of manipulations: in the first iteration, we consider only manipulations *to the highest possible type* (*i.e.* the highest possible value that an agent may have for the public good); in the second, we also consider manipulations to the second highest possible type; *etc.*, up until the last iteration, in which we consider manipulations all the way down to a value of 0.

[7]It is not necessary to have this secondary objective for the result to go through, but it simplifies the analysis.

Technically speaking, this approach only works on a finite type space. If the type space is $\mathbb{R}^{\geq 0}$ (all nonnnegative valuations), we encounter two problems: first, there is no highest valuation to start with; and second, there are uncountably infinitely many valuations, leading to infinitely many iterations. Thus, it would not be possible to run the approach automatically in this case. However, for the purpose of theoretical analysis, we can (and will) still consider the case where the type space is $\mathbb{R}^{\geq 0}$: the first problem is overcome by the fact that manipulating to a *higher* type is not beneficial in this domain (free-riders pretend to have a lower valuation); the second problem is overcome by conceiving of this process as the limit of a sequence of similar processes corresponding to finer and finer discretizations of the nonnegative numbers. (If we were to actually run on a discretization, the final resulting mechanism would be close to the mechanism that results in the limit case—and the finer the discretization, the closer the result will be.)

Before we describe the second phase, we will first analyze what happens in the first phase.

Lemma 23 *After considering manipulations to value r, the mechanism will take the following form ($\hat{v}_i$ is agent i's reported value):*

1. *The good will be produced if and only if* $\sum_i \hat{v}_i \geq c$;
2. *If the good is produced, and* $\sum_i \min\{r, \hat{v}_i\} > c$, *then every agent i will pay* $\min\{r, \hat{v}_i\}$;
3. *If the good is produced, and* $\sum_i \min\{r, \hat{v}_i\} \leq c$, *then, letting $t \geq r$ be the number such that* $\sum_i \min\{t, \hat{v}_i\} = c$, *every agent i will pay* $\min\{t, \hat{v}_i\}$.

Proof: Suppose we have proved the result for manipulations to r; let us prove the result for manipulations to an infinitesimally smaller r'. Consider an arbitrary type vector $v = \langle v_1, \ldots, v_n \rangle$ with $\sum_i v_i \geq c$. In the mechanism M that results after considering manipulations to r only, any agent i with $v_i \leq r'$ has no incentive to manipulate to r' (after the manipulation, the agent will be made to pay at least $r' \geq v_i$), so we will not change such an agent's payments. An agent with $v_i > r'$, however, will have an incentive to manipulate to r', *if* this manipulation does not prevent the production of the good (the agent will pay r' rather than the at least $\min\{r, v_i\} > r'$ that he would have paid without manipulation). If the total payment under M given v exceeds c (that is, 2. above applies), then manipulating to r' in fact does not prevent the production of the good, so all such agents have an incentive to manipulate; but, on the other hand, we can reduce the payment of such agents from r to r' in the new mechanism for type vector v, which will prevent the manipulation. However, if the total payment under M given v is exactly c (that is, 3. above applies), then it is impossible to reduce the payments of such agents to r', because we cannot collect any more money from the remaining agents and hence we would not be able to afford the good. ■

Corollary 31 *After considering all manipulations (including to $r = 0$), the mechanism will take the following form:*

1. *The good will be produced if and only if* $\sum_i \hat{v}_i \geq c$;

2. *If the good is produced, then, letting t be the number such that $\sum_i \min\{t, \hat{v}_i\} = c$, every agent i will pay* $\min\{t, \hat{v}_i\}$.

Figure 10.1: Example of a public good setting in which there are 3 agents; the public good costs 9 to produce. The horizontal lines represent the agents' true valuations (8, 1, and 6), which are sufficient to produce the good. The circles represent the payments that the agents make for this type vector after considering manipulations to 7; the crosses represent the payments that the agents make for this type vector after considering manipulations to 4. At this stage the payments sum exactly to 9, so the payments remain at this level even after considering manipulations to even lower values.

The mechanism from Corollary 31 (call it M_1) is still not strategy-proof. For example, in the example in Figure 10.1, suppose that agent 2's valuation for the good is 3 instead. Then, M_1 will charge agent 2 a payment of 3 instead of 1. Thus, agent 2 will be better off reporting 1 instead. However, the next phase will make the mechanism strategy-proof.

In phase two, we take approach **(b)** above: for a type vector *into* which there are beneficial manipulations, make its outcome *un*desirable enough to the manipulating agents to prevent the manipulations. We will do so by not producing the good at all for such type vectors. We will perform a single iteration of this, considering all possible manipulations.

Lemma 24 *A type vector $\langle v_1, \ldots, v_n \rangle$ for which the mechanism M_1 produces the good has a manipulation into it if and only if for some i, $v_i < c/n$.*

Proof: If $v_i < c/n$, consider the modified type vector $\langle v_1, \ldots, v_{i-1}, v_i', v_{i+1}, \ldots, v_n \rangle$ with $v_i' = c/n$. For this modified type vector, i must pay at least c/n (because it must always be the case that

$t \geq c/n$), and thus i would be better off manipulating into the original type vector. This proves the "if" part of the lemma.

On the other hand, if for all i, $v_i \geq c/n$, then $t = c/n$, and all agents pay c/n. Suppose there exists a beneficial manipulation by some agent i into this type vector, from some modified type vector $\langle v_1, \ldots, v_{i-1}, v'_i, v_{i+1}, \ldots, v_n \rangle$ for which the good is also produced. (It is never beneficial to manipulate from a type vector for which the good is not produced, as the manipulating agent would have to pay more than his value for the good.) We must have $v'_i > c/n$, for otherwise i would be at least as well off reporting truthfully. But then, everyone pays c/n in the modified type vector as well, contradicting the incentive to manipulate. This proves the "only if" part of the lemma. ■

Thus, we have the following theorem for the mechanism M_2 that results after one iteration of the second phase:

Theorem 108 *M_2 produces the good if and only if for all i, $\hat{v}_i \geq c/n$; and if so, M_2 charges every agent c/n.*

Thus, our approach has produced a very simple mechanism that most of us have encountered at some point in life: the agents are to share the costs of producing the good equally, and if one of them refuses to do so, the good will not be produced (and nobody will have to pay). This mechanism may seem somewhat disappointing, especially if it is unlikely that all the agents will value the good at at least c/n. However, it turns out that *this is in fact the best possible anonymous strategy-proof mechanism* (that satisfies the individual rationality constraint). (Moulin [1994] has already shown a similar result in a more general setting in which multiple levels of the public good can be produced; however, he required *coalitional* strategy-proofness, and he explicitly posed as an open question whether the result would continue to hold for the weaker notion of (individual) strategy-proofness.)

Deriving the plurality-with-runoff rule for voting

In this subsection, we address voting (social choice) settings. Recall that in such a setting, every agent (voter) i's type is a complete ranking $\succ_i$ over the outcomes (candidates). The mechanism (voting rule) takes as input the agents' type reports (votes), consisting of complete rankings of the candidates, and chooses an outcome.

Recall that under the commonly used plurality rule, we only consider every voter's highest-ranked candidate, and the winner is simply the candidate with the highest number of votes ranking it first (its plurality score). The plurality rule is very manipulable: a voter voting for a candidate that is not close to winning may prefer to attempt to get the candidate that currently has the second-highest plurality score to win, by voting for that candidate instead. In the real world, one common way of "fixing" this is to add a runoff round, resulting in the plurality-with-runoff rule. Recall that under this rule, we take the two candidates with the highest plurality scores, and declare as the winner the one that is ranked higher by more voters. By the Gibbard-Satterthwaite theorem (Chapter 4), this is still not a strategy-proof mechanism (it is neither dictatorial nor does it preclude any candidate from winning)—indeed, a voter may change his vote to change which candidates are in the runoff. Still, the plurality with runoff rule is, in an intuitive sense, "less" manipulable than the plurality rule (and certainly more desirable than a strategy-proof rule, since it would either be dictatorial or preclude some candidate from winning).

In this subsubsection, we will show that the following variant of our approach will produce the plurality-with-runoff rule when starting with the plurality rule as the initial mechanism.

- The set F of manipulations that we consider is that of all manipulations in which a voter changes which candidate he ranks first.
- We try to prevent manipulations as follows: for a type (vote) vector from which there is a beneficial manipulation, consider all the outcomes that may result from such a manipulation (in addition to the current outcome), and choose as the new outcome the one that minimizes the number of agents that still have an incentive to manipulate from this vote vector.
- We will change the outcome for each vote vector at most once (but we will have multiple iterations, for vote vectors whose outcome did not change in earlier iterations).

Theorem 109 *For a given type vector θ, suppose that candidate b is ranked first the most often, and a is ranked first the second most often ($s(b) > s(a) > \ldots$, where $s(o)$ is the number of times o is ranked first). Moreover, suppose that the number of votes that prefers a to b is greater than or equal to the number of votes that prefers b to a. Then, starting with the plurality rule, after exactly $s(b) - s(a)$ iterations of the approach described above, the outcome for θ changes for the first time, to a (the outcome of the plurality with runoff rule).*[8]

Proof: We will prove the result by induction on $s(b) - s(a)$. First note that there must be some voter that prefers a to b but did not rank a first. Now, if $s(b) - s(a) = 1$, a beneficial manipulation for this voter is to rank a first, which will make a win. No other candidate can be made to win with a beneficial manipulation (no voter ranking b first has an incentive to change his vote, hence no beneficial manipulation will reduce b's score; any other candidate's score can be increased by at most 1 by a manipulation; and every candidate besides a is at least two votes behind b). Thus, in the first iteration, we must decide whether to keep b as the winner, or change it to a. If we keep b as the winner, all the voters that prefer a to b but do not rank a first have an incentive to change their vote (and rank a first). On the other hand, if we change the winner to a, all the voters that prefer b to a but do not rank b first have an incentive to change their vote (and rank b first), so that b leads by two votes and wins. So, in which case do we have more voters with an incentive to change their vote? In the first case, because there are at least as many voters preferring a to b as b to a, and there are fewer voters among those preferring a to b that rank a first than there are voters preferring b to a that rank b first. Hence, we will change the outcome to a.

Now suppose that we have proven the result for $s(b) - s(a) = k - 1$; let us prove it for $s(b) - s(a) = k$. First, we note that in prior iterations, there were no beneficial manipulations from the type vector that we are considering (no voter ranking b first has an incentive to change his vote, thus any beneficial manipulation can only reduce the difference between $s(b)$ and the score of another candidate by 1, and by the induction assumption no vote vector that results from such a manipulation has had its outcome changed in earlier iterations—*i.e.* it is still b), and thus the outcome has not yet changed in prior iterations. But, by the induction assumption, any vote vector

[8] This is assuming that ties in the plurality rule are broken in favor of a; otherwise, one more iteration is needed. (Some assumption on tie-breaking must always be made for voting rules.)

that can be obtained from the vote vector that we are considering by changing one of the votes ranking a higher than b but not first, to one that ranks a first, must have had its outcome changed to a in the previous round. Thus, now there is a beneficial manipulation that will make a the winner. On the other hand, no other candidate can be made to win by such a beneficial manipulation, since they are too far behind b given the current number of iterations. The remainder of the analysis is similar to the base case ($s(b) - s(a) = 1$). ■

10.1.4 Computing the outcomes of the mechanism

In this subsection, we discuss how to automatically compute the outcomes of the mechanisms that are generated by this approach in general. It will be convenient to think about settings in which the set of possible type vectors is finite (so that the mechanism can be represented as a finite table), although these techniques can be extended to (some) infinite settings as well. One potential upside relative to standard automated mechanism design techniques (as presented in Chapter 6) is that we do not need to compute the entire mechanism (the outcomes for all type vectors) here; rather, we only need to compute the outcome for the type vector that is actually reported.

Let M_0 denote the (naïve) mechanism from which we start, and let M_t denote the mechanism after t iterations. Let F_t denote the set of beneficial manipulations that we are considering (and are trying to prevent) in the tth iteration. Thus, M_t is a function of F_t and M_{t-1}. What this function is depends on the specific variant of the approach that we are using. When we try to prevent manipulations by making the outcome for the type vector from which the agent is manipulating more desirable for that agent, we can be more specific, and say that, for type vector θ, $M_t(\theta)$ is a function of the subset $F_t^\theta \subseteq F_t$ that consists of manipulations that start from θ, and of the outcomes that M_{t-1} selects on the subset of type vectors that would result from a manipulation in F_t^θ. Thus, to compute the outcome that M_t produces on θ, we only need to consider the outcomes that M_{t-1} chooses for type vectors *that differ from θ in at most one type* (and possibly even fewer, if F_t^θ does not consider all possible manipulations). As such, we need to consider M_{t-1}'s outcomes on at most $\sum_{i=1}^{n} |\Theta_i|$ type vectors to compute $M_t(\theta)$ (for any given θ), which is much smaller than the set of all type vectors ($\prod_{i=1}^{n} |\Theta_i|$). Of course, to compute $M_{t-1}(\theta')$ for some type vector θ', we need to consider M_{t-2}'s outcomes on up to $\sum_{i=1}^{n} |\Theta_i|$ type vectors, *etc.*

Because of this, a simple recursive approach for computing $M_t(\theta)$ for some θ will require $O((\sum_{i=1}^{n} |\Theta_i|)^t)$ time. This approach may, however, spend a significant amount of time recomputing values $M_j(\theta')$ many times. Another approach is to use dynamic programming, computing and storing mechanism M_{j-1}'s outcomes on *all* type vectors before proceeding to compute outcomes for M_j. This approach will require $O(t \cdot (\prod_{i=1}^{n} |\Theta_i|) \cdot (\sum_{i=1}^{n} |\Theta_i|))$ time (for every iteration, for every type vector, we must investigate all possible manipulations). We note that when we use this approach, we may as well compute the entire mechanism M_t (we already have to compute the entire mechanism M_{t-1}). If n is large and t is small, the recursive approach is more efficient; if n is small and t is large, the dynamic programming approach is more efficient.

All of this is for fully general (finite) domains; it is likely that these techniques can be sped up considerably for specific domains. Moreover, as we have already seen, some domains can simply be solved analytically.

10.1.5 Computational hardness of manipulation

We have already demonstrated that our approach can change naïve mechanisms into mechanisms that are less (sometimes not at all) manipulable. In this subsection, we will argue that in addition, if the mechanism remains manipulable, *the remaining manipulations are computationally difficult to find*. This is especially valuable because, as we argued earlier, if it is computationally too difficult to discover beneficial manipulations, the revelation principle ceases to meaningfully apply, and a manipulable mechanism can sometimes actually outperform all truthful mechanisms.

In this subsection, we first present an informal, but general, argument for the claim that any manipulations that remain after a large number of iterations of our approach are hard to find. This argument depends on the assumption that the agents only use the most straightforward possible algorithm for finding manipulations. Second, we show that if we add a random component to our approach for updating the mechanism, then we can prove formally that detecting whether there is a beneficial manipulation becomes #P-hard.

An informal argument for hardness of manipulation

Suppose that the only thing that an agent knows about the mechanism is the variant of our approach by which the designer obtains it (the initial naïve mechanism, the manipulations that the designer considers, how she tries to eliminate these opportunities for manipulations, how many iterations she performs, *etc.*). Given this, one natural algorithm for an agent to find a beneficial manipulation is to simulate our approach for the relevant type vectors, perhaps using the algorithms presented earlier. However, this approach is computationally infeasible if the agent does not have the computational capabilities to simulate as many iterations as the designer will actually perform.

Of course, this argument fails if the agent actually has greater computational abilities or better algorithms than the designer. In the next subsubsection, we will give a different, formal argument for hardness of manipulation for one particular instantiation of our approach.

Random sequential updating leads to #P-hardness

So far, we have only discussed updating the mechanism in a deterministic fashion. When the mechanism is updated deterministically, any agent that is computationally powerful enough to simulate this updating process can determine the outcome that the mechanism will choose, for any vector of revealed types. Hence, that agent can evaluate whether he would benefit from misrepresenting his preferences. However, this is not the case if we add random choices to our approach (and the agents are not told about the random choices until after they have reported their types).

The hardness result that we show in this subsubsection holds even for a single agent; therefore we will only specify the variant of our approach used in this subsubsection for a single agent. Any generalization of the variant to more than one agent will have the same property.

First, we take the set Θ of all of the agent's types, and organize the types as a sequence of $|\Theta|/2$ pairs of types:[9] $((\theta_{11}, \theta_{12}), (\theta_{21}, \theta_{22}), \ldots, (\theta_{|\Theta|1}, \theta_{|\Theta|2}))$. In the ith iteration, we randomly choose between θ_{i1} and θ_{i2}, and consider all the beneficial manipulations out of the chosen type (and no other manipulations). (We will only need $|\Theta|/2$ iterations.) Then, as before, we try to prevent these manipulations by making the outcome for the chosen type more appealing to an agent with that type (and if there are multiple ways of doing so, we choose the one that maximizes the designer's objective).

We are now ready to present our #P-hardness result. We emphasize that, similarly to the hardness results in Chapter 8, this is only a worst-case notion of hardness, which may not prevent manipulation in all cases.

Theorem 110 *Evaluating whether there exists a manipulation that increases an agent's expected utility is #P-hard*[10] *under the variant of our technique described above.*

Proof: We reduce an arbitrary #SAT instance, together with a number K, to the following setting. Let there be a single agent with the following types. For every variable $v \in V$, we have types θ_{+v} and θ_{-v}; for every clause $c \in C$, we have types θ_c^1 and θ_c^2; finally, we have four additional types $\theta_1, \theta_2, \theta_3, \theta_4$. The sequence of pairs of types is as follows: $((\theta_{+v_1}, \theta_{-v_1}), \ldots, (\theta_{+v_{|V|}}, \theta_{-v_{|V|}}), (\theta_{c_1}^1, \theta_{c_1}^2), \ldots, (\theta_{c_{|C|}}^1, \theta_{c_{|C|}}^2), (\theta_1, \theta_2), (\theta_3, \theta_4))$. Let the outcome set be as follows: for every variable $v \in V$, we have outcomes o_{+v} and o_{-v}; for every clause $c \in C$, we have an outcome o_c; finally, we have outcomes o_1, o_2, o_3, o_4. The utility function is zero everywhere, with the following exceptions: for every literal l, $u(\theta_l, o_l) = 2, u(\theta_l, o_2) = 1$; for every clause c, $u(\theta_c, o_c) = 4, u(\theta_c, o_l) = 3$ if l occurs in c, $u(\theta_c, o_3) = 2, u(\theta_c, o_2) = 1$; for $\theta \in \{\theta_1, \theta_2\}$, $u(\theta, o_4) = \frac{2^{|V|+1}}{2^{|V|}-K}, u(\theta, o_1) = 1$; for $\theta \in \{\theta_3, \theta_4\}$, $u(\theta, o_4) = 2, u(\theta, o_3) = 1$. The designer's objective function is zero everywhere, with the following exceptions: for all $\theta \in \Theta$, $g(\theta, o_1) = 3$; $g(\theta_3, o_2) = g(\theta_4, o_2) = 4$; for every literal l, $g(\theta_l, o_l) = 2$; for every clause c, $g(\theta_c, o_3) = 2, g(\theta_c, o_c) = 1$; $g(\theta_3, o_4) = g(\theta_4, o_4) = 2$. The initial mechanism, which naïvely maximizes the designer's objective, chooses o_1 for all types, with the exception of θ_3 and θ_4, for which it chooses o_2.

The mechanism will first update exactly one of θ_{+v} and θ_{-v}, for every $v \in V$. Specifically, if θ_l (where l is a literal) is updated, the new outcome chosen for that type will be o_l (which is more desirable to the agent that o_2, and better for the designer than o_2). Subsequently, exactly one of θ_c^1 and θ_c^2 is updated. Specifically, if θ_c^i is updated, the new outcome chosen for that type will be o_3 if no type θ_l with $l \in c$ has been updated (and therefore no o_l with $l \in c$ is ever chosen by the mechanism), and o_c otherwise. (o_3 is more desirable to the agent than o_2, but less desirable than some o_l with $l \in c$; however, o_c is even more desirable than such an o_l. The designer would prefer to prevent the manipulation with o_3, but o_c is the next best way of preventing the manipulation if o_3 will not suffice.) Then, one of θ_1 and θ_2 is updated, but the outcome will not be changed for either of them (the only outcome that the agent would prefer to o_1 for these types is o_4, which the mechanism does not yet choose for any type); finally, one of θ_3 and θ_4 is updated, and the outcome for this

[9]The hardness result will hold even if the number of types is restricted to be even, so it does not matter how this is generalized to situations in which the number of types is odd.

[10]Technically, we reduce from a decision variant of #SAT ("Are there fewer than K solutions?"). An algorithm for this decision variant can be used to solve the original problem using binary search.

type will change to o_4 if and only if outcome o_3 is now chosen by the mechanism for some type θ_c^i (that outcome would be preferred to o_2 by the agent for these types; o_4 is the next best outcome for the designer). We note that this update to o_4 will *not* happen if and only if, for every clause c, for at least one literal $l \in c$, θ_l was updated (rather than θ_{-l})—because in this case (and only in this case), whenever we updated one of θ_c^1, θ_c^2, o_c was chosen (rather than o_3). In other words, it will not happen if and only if the literals l that were chosen constitute a satisfying assignment for the formula. The probability that this happens is $n/(2^{|V|})$, where n is the number of solutions to the SAT formula.

Now, let us consider whether the agent has an incentive to manipulate if his type is θ_1. Reporting truthfully will lead to outcome o_1 being chosen, giving the agent a utility of 1. It only makes sense to manipulate to a type for which o_4 may be chosen, because o_1 is preferred to all other outcomes—hence, any beneficial manipulation would be to θ_3 or θ_4. Without loss of generality, let us consider a manipulation to θ_3. What is the chance (given that we have done exactly $|\Theta|/2$ updates) that we choose o_4 for θ_3? It is $1/2$ (the chance that θ_3 was in fact updated) times $1 - n/(2^{|V|})$ (the chance that o_4 was chosen), which is $(2^{|V|} - n)/(2^{|V|+1})$. Otherwise, o_2 is still chosen for θ_3, and manipulating to θ_3 from θ_1 would give utility 0. Thus, the expected utility of the manipulation is $\frac{2^{|V|}-n}{2^{|V|+1}} \frac{2^{|V|+1}}{2^{|V|}-K}$. This is greater than 1 if and only if $n < K$. ■

10.2 Summary

In this chapter, we suggested an approach for (automatically) designing mechanisms for bounded agents. Under this approach, we start with a naïve (manipulable) mechanism, and incrementally make it *more* strategy-proof over a sequence of iterations.

We gave various examples of mechanisms that (variants of) our approach generate, including: the VCG mechanism in general settings with payments; an equal cost sharing mechanism for public goods settings; and the plurality-with-runoff voting rule. We also provided several basic algorithms for automatically executing our approach in general settings, including a recursive algorithm and a dynamic programming algorithm, and analyzed their running times. Finally, we discussed how computationally hard it is for agents to find any remaining beneficial manipulation. We argued that agents using a straightforward manipulation algorithm will not be able to compute manipulations if the designer has greater computational power (and can thus execute more iterations). We also showed that if we add randomness to how the outcomes are computed, then detecting whether a manipulation that is beneficial in expectation exists becomes #P-hard for an agent.

Chapter 11

Conclusions and Future Research

This dissertation set out to investigate the role that computation plays in various aspects of preference aggregation, and to use computation to improve the resulting outcomes. In this final chapter, we will review the research contributions of this dissertation, as well as discuss directions for future research.

11.1 Contributions

The following are the main research contributions of this dissertation. (Some minor contributions are omitted.)

- **A hierarchical framework** that categorizes various ways in which computational tools can improve preference aggregation (Chapter 1). This framework provides an organized view of much of the work on computational aspects of preference aggregation (and, in particular, of the research in this dissertation); it also provides a natural guide towards future research (as we will discuss in Section 11.2).

- **New settings for expressive preference aggregation** (Chapter 2). Specifically, we introduced expressive preference aggregation for *donations to (charitable) causes*, which allows agents to make their donations conditional on how much, and to which charities, other agents donate. We also introduced expressive preference aggregation in settings with *externalities*, where each agent controls variables that affect the welfare of the other agents.

- **New techniques for solving outcome optimization problems** in expressive preference aggregation settings (Chapter 3). In the context of *voting*, we introduced a powerful preprocessing technique for computing Slater rankings: a set of similar candidates can be solved recursively and replaced with a single super-candidate in the original problem. For *combinatorial auctions*, we showed that if the items are vertices in a graph, and each bid is on a connected component of the graph, then the winner determination problem can be solved efficiently if the graph is known and has bounded treewidth, or if the graph is a tree (in which case we can discover the graph). For the setting of expressive preference aggregation for *donations to charities*, we showed that the outcome optimization problem is inapproximable, but we also provided mixed integer programming techniques for solving this problem in general, and exhibited special cases that are easy to solve. Finally, for the setting of expressive preference aggregation in settings with *externalities*, we showed that it is typically NP-hard to find a nontrivial feasible solution, but we also exhibited two special cases in which this is not the case (in one, the feasible solution with the maximal concessions can be found efficiently, and in the other, the social-welfare maximizing solution).

- **An analysis of classical mechanism design techniques** in expressive preference aggregation settings (Chapter 5). We showed that the *VCG mechanism* is extremely vulnerable to collusion, and provides poor revenue guarantees, in combinatorial auctions and exchanges. We also gave conditions that characterize when these vulnerabilities occur, and studied how computationally hard it is to decide if these conditions hold. For aggregating preferences over *donations to charities*, we showed a fundamental impossibility result that precludes the existence of an ideal mechanism (even with only a single charity), but we also gave some positive results that show that negotiating over multiple charities simultaneously can be helpful in designing good mechanisms.

- **Automated mechanism design** (Chapter 6). We defined the basic problem of automated mechanism design and determined its computational complexity for a number of special cases. We introduced a linear programming (mixed integer programming) approach for designing randomized (deterministic) mechanisms in general, and a special-purpose algorithm for a special case. We gave various applications and presented scalability results. Finally, we studied a more concise representation of problem instances, and showed how this changes the complexity of the problem.

- **Mechanisms that make manipulation computationally hard** (Chapter 8). We demonstrated that *the revelation principle fails when agents are computationally bounded*. Specifically, we exhibited a family of settings where a non-truthful mechanism is easier to execute and harder to manipulate than the best truthful mechanism; moreover, the non-truthful mechanism will perform just as well as the optimal truthful mechanism if it is manipulated nonetheless, and otherwise it will perform strictly better. We then showed that in voting, *adding a preround* can make manipulation (by an individual voter) significantly harder—NP-hard, #P hard, or PSPACE-hard, depending on whether the scheduling of the preround precedes, follows, or is interleaved with the voting, respectively. We also showed that coalitional weighted manipulation can be hard even for voting settings *with few candidates*, and that these results also imply hardness of manipulation by an individual if there is uncertainty about the others' votes. Finally, we gave an impossibility result showing that voting rules that are *usually hard to manipulate* do not exist, if we require the rule and the distribution over instances to satisfy some other sensible properties.

- **An analysis of the complexity of computing game-theoretic solutions** (Chapter 9). We showed that the basic computational problems related to *dominance* can be solved efficiently, with a few exceptions: for iterated weak dominance in normal-form games, and dominance by pure strategies in Bayesian games, these questions are NP-complete; and iterated dominance in Bayesian games can even require an exponential number of iterations. We also gave a single reduction that shows that finding (even approximately) optimal *Nash equilibria* (for various notions of optimal), or Nash equilibria with certain other properties, is NP-complete; and that counting the number of (connected sets of) Nash equilibria is #P-hard. In addition, we showed that determining whether a pure-strategy Bayes-Nash equilibrium exists is NP-complete. Finally, we introduced a new *parameterized definition of strategy eliminability* and showed that it generalizes both strict dominance and Nash equilibrium eliminability (in that these are obtained for some settings of the parameters). We also showed that strategy eliminability is efficiently computable if and only if the parameter settings are close to those corresponding to dominance.

- **A methodology for incrementally making mechanisms more strategy-proof** (Chapter 10). This work can be interpreted as one methodology for automatically designing mechanisms for bounded agents, and as such is a first step towards our long-term goal of bringing automated mechanism design and mechanism design for bounded agents together. The idea of this methodology is to start with a naïve mechanism, such as the one that maximizes social welfare without any incentive compatibility constraints, and subsequently to identify opportunities for manipulation and locally change the mechanism to prevent these. We showed that this approach can generate certain known mechanisms (including non-truthful ones). We introduced algorithms for automatically computing outcomes according to this approach, and argued that the resulting mechanisms are hard to manipulate.

11.2 Future research

The hierarchy introduced in this dissertation provides a natural guide to future research. Typically, a new domain for expressive preference aggregation will initially be studied at the shallow levels of the hierarchy, after which research on the domain will gradually move to deeper levels. For example, the allocation of tasks and resources (using combinatorial auctions and exchanges) was initially studied at the shallowest node (outcome optimization); in recent years, most research on this domain has focused on (algorithmic) mechanism design, the second node in the hierarchy; and most recently, automated mechanism design has started to be applied to these settings. In contrast, domains that have only recently started receiving serious attention, such as expressive negotation in settings with externalities, are still being studied exclusively at the level of outcome optimization. Hence, natural directions for future research include pushing existing domains deeper down the hierarchy, as well as introducing new domains—or formalizing domains that already exist in the real world—and (presumably) studying them at the shallowest levels first. Additionally, in the context of mechanism design for bounded agents (and especially automated mechanism design for bounded agents), it is not yet completely clear how mechanisms should be evaluated. Thus, future research at these nodes will also involve developing a general theory for such evaluation. Domain-specific studies, such as the ones we did on voting, may help in doing so.

Much research also remains to be done on topics orthogonal to the hierarchy, such as preference elicitation and distributed computation of outcomes (see Section 1.4). These topics can be studied at each node in the hierarchy, for any domain. However, typically, doing so requires that research on that domain at that node has already reached a basic level of maturity—specifically, it requires that we have a good grasp on how outcomes should be chosen, and how these outcomes can be efficiently computed given the preferences of the agents. Because of this, these orthogonal topics will usually not be the first to receive attention, but this is certainly not because they are unimportant.

While the hierarchy proposed in this dissertation provides a high-level guideline to future research, the research contributions at the individual nodes of the hierarchy suggest many more specific open questions and directions. The remainder of this section will lay out some of these more immediately accessible avenues for future research.

11.2.1 Node (1): Outcome optimization

In Section 3.1, we saw how the preprocessing technique of finding and aggregating sets of similar candidates into super-candidates can drastically speed up the search for optimal Slater rankings (which are NP-hard to find). One may ask whether similar techniques can be applied to other hard-to-execute voting rules. We already discussed how the technique can be extended to apply to computing Kemeny rankings, but we do not yet have experimental results on the efficacy of doing so. It would also be interesting to characterize restrictions on the votes that have the effect of making the preprocessing technique sufficient to solve the Slater problem in polynomial time. (One such restriction that we discussed is that of having a hierarchical pairwise election graph, but there may be other restrictions with this property.) Another possibility is to look for entirely different preprocessing techniques, or to try to generalize the similar-candidates technique. Finally, given a good understanding of what makes a voting rule amenable to the use of such techniques, one may use this understanding in the design of new voting rules—by ensuring that these new rules allow for the application of such techniques and can therefore be executed fast.

In Section 3.2, we showed how the winner determination problem in combinatorial auctions can be solved in polynomial time, *if* we know an item graph for the instance that has bounded treewidth; additionally, we saw how to *find* an item *tree* in polynomial time (if it exists). This left us with a very specific open question: can we find item graphs with small treewidth (but treewidth greater than 1) in polynomial time if they exist? For example, can we find an item graph of treewidth 2 in polynomial time (if it exists)? Alternatively, given that an item graph of treewidth k exists, can we find an item graph of treewidth at most (say) $2k$? If we can, then the mere fact that an item graph of bounded treewidth *exists* for the winner determination problem at hand will guarantee that it can be solved in polynomial time (whereas now, we additionally require that we *know* the graph). As another specific open question (perhaps of less significance), we showed that in the case where bids are allowed to contain multiple components, constructing a line item graph is NP-complete when 5 components per bid are allowed; but we left open whether this is so for fewer (say, 4) components. Additional future research directions include comparing the item-graph based algorithms to other winner determination algorithms (*e.g.* using a solver such as CPLEX), as well as *integrating* the item-graph based algorithms into search algorithms (where they can be applied at individual nodes in the search tree).

As for the framework and algorithms for expressive preference aggregation for donations to charities (Sections 2.3 and 3.3), one possible future direction is to build a web-based implementation of these techniques that will allow them to be used in practice. Another direction is to experimentally test the scalability of the mixed integer and linear programming formulations of the clearing problem that we proposed. One can also try to characterize other restrictions on the bids that make the clearing problem easy to solve. Another possibility is to consider the elicitation problem in this setting, and to design good iterative mechanisms for addressing this problem. Finally, one can consider other bidding languages—for example, languages that allow donations to be conditional on which other donors are donating (and on how much they are donating).

For the outcome optimization problem in settings with externalities (Sections 2.4 and 3.4), we showed mostly negative (NP-hardness) results. Future research can potentially address this in several ways. First, algorithms could be given that require exponential time in the worst case, but run fast in practice; one possibility for this could be the use of mixed integer programming techniques. Another possibility is to try to design approximation algorithms, although this does not look very promising given that even the problem of finding a nontrivial feasible solution is NP-hard in most settings that we studied. Finally, we may try to simplify the problem, possibly by changing or restricting the language in which agents express their preferences.

11.2.2 Node (2): Mechanism design

Our research on vulnerability of the VCG mechanism to collusion and low revenue in combinatorial auctions and exchanges (Section 5.1) suggests a number of future research directions. First, it would be desirable to create new mechanisms that do not share these weaknesses. These mechanisms may be truthful mechanisms—perhaps even other Groves mechanisms—but they may also be non-truthful mechanisms. For example, we saw that (even under strategic behavior by the agents) the first-price mechanism does not run into trouble in some of the instances that caused problems for the VCG mechanism, and it would certainly be interesting to characterize more generally how first-price mechanisms perform in terms of collusion and revenue. As another direction, we only characterized when certain worst-case scenarios can occur under the VCG mechanism—but there are certainly other instances in which bad (albeit not worst-case) outcomes can occur. Providing a more complete characterization that also classifies these instances would give us an even better understanding of the VCG mechanism's vulnerabilities.

Mechanisms for expressive preference aggregation in the setting of donations to charities (Section 5.2) (or, more generally, for expressive preference aggregation in settings with externalities/public goods) still leave much to be desired. In part, this is due to fundamental impossibility results such as the one that we presented. Nevertheless, our results also suggest that such impossibilities can sometimes be mitigated by using a single mechanism to decide on the donations to multiple charities—although it is not yet clear how to do this in the general case. While difficulties for mechanism design in these settings occur even when restricting our attention to the case of quasilinear preferences, it is important that eventually mechanisms will address the case of more general preferences as well. This is especially so because typically, when donating money to a large charity, the marginal benefit to the charity of another dollar remains roughly constant even when donations are large. Thus, the main reason why donors give only limited amounts is that larger donations will prevent them from buying more basic goods for themselves (*e.g.* food, clothing)—that is, as they donate more, the marginal utility of keeping a dollar for themselves becomes larger, and thus their utility is not linear in money. Automated mechanism design may be helpful in creating mechanisms in the face of non-quasilinear utilities.

11.2.3 Node (3a): Automated mechanism design

Automated mechanism design is a relatively new research area, and because of this much remains to be done. Perhaps the most important direction for future research is getting automated mechanism design to scale to larger instances. There are numerous ways in which this can be achieved. First of all, better algorithms for the general problem can be developed. This can be done either by improving the mixed integer/linear programming formulations, or by developing new algorithms altogether. As examples of the former, we have observed (1) that some of the constraints in our formulation imply some of the others (similar observations have been made by others [Gui *et al.*, 2004; Lovejoy, 2006]), and omitting these implied constraints (perhaps surprisingly) significantly decreases the time that CPLEX requires to solve instances; and (2) in settings that are symmetric with respect to the agents, formulations that are much more concise (and easier to solve) can be given. As an example of the latter, in Section 6.7 we introduced a special-purpose algorithm for the case of designing deterministic algorithms without payments for a single agent, and perhaps this algorithm can be extended to apply to the general problem.

On the other hand, rather than address the general problem, one can also choose to focus on specific domains. For example, one can restrict attention to the automated design of combinatorial auction mechanisms, as has been done by Likhodedov and Sandholm [2003, 2004, 2005]. By focusing on such a specific domain, it is possible to make use of theoretical results that characterize (optimal) truthful mechanisms in that domain, which can significantly reduce the space of mechanisms that must be searched. It is also possible to restrict the space of mechanisms under consideration without such a characterization result. For example, Sandholm and Gilpin [2006] restrict attention to sequences of take-it-or-leave-it offers for selling items. Such a restriction will potentially exclude all optimal mechanisms from the search space, but typically there are many reasonable restrictions of the search space that one would not expect to come at too much of a cost. For example, one can require that the lowest k bids never win anything. (Such restrictions, besides improving scalability, also allow us to rule out mechanisms that are intuitively unappealing.) If there is a formal guarantee that optimal mechanisms in the restricted search space always have objective values that are close to those of optimal mechanisms in the unrestricted search space, then an algorithm that identifies an optimal mechanism in the restricted search space constitutes an approximation algorithm for the unrestricted setting.

There are many other important future directions on automated mechanism design besides improving its scalability. New domains in which AMD can be applied continue to be found (for example, recommender systems [Jurca and Faltings, 2006]). One can also use AMD as a tool in traditional mechanism design. For example, by letting the software compute the optimal mechanism for a number of sample instances in the domain of interest, one can gain intuition about what the characterization of the optimal mechanism for the general case should look like. It is also possible to use automated mechanism design software to help disprove general conjectures, by solving randomly sampled instances until a counterexample is found. For example, a conjecture that optimal mechanisms in a certain domain need never use randomization can be disproved by finding an instance in the domain where the optimal randomized mechanism performs strictly better than the optimal deterministic mechanism.

Finally, one can seek to expand the automated mechanism design toolbox, for instance by studying how to model additional solution concepts. For example, one can add constraints to the problem to prevent collusion from being beneficial. One can also try to modify the Bayes-Nash equilibrium constraints so that truthful reporting remains optimal even if the designer's prior over agents' types is slightly inaccurate. Additionally, it may be possible to create tools for cases where the designer has only partial information about the prior over agents' types. Finally, it would be useful to have techniques for the case where the type space (and perhaps also the outcome space) is continuous rather than discrete. Even if techniques for solving such continuous instances directly remain elusive, it would help to at least know how to discretize them well.

11.2.4 Node (3b): Mechanism design for bounded agents

In Section 8.1 we showed that there exist settings in which there are non-truthful mechanisms that perform at least as well as any truthful mechanism (and strictly better if agents are computationally bounded), and that are also computationally easier to execute. Future research should investigate whether this result can be generalized to other settings of interest, such as combinatorial auctions. One may also consider non-truthful mechanisms that do not have all three of these properties, *i.e.* non-truthful mechanisms that are not easier to execute, or are not guaranteed to perform strictly better in the face of computational boundedness, or are not guaranteed to perform at least as well when agents are strategic. In the last case, it would be risky to run this non-truthful mechanism instead of the optimal truthful one because the outcome may be worse, and hence it would be good to have some way of assessing this risk, *i.e.* being able to estimate what the odds are that the resulting outcome will be worse (or better), and how much worse (or better) it will be.

There are also various avenues for future research on voting rules that are computationally hard to manipulate (Sections 8.2, 8.3, and 8.4). Most significantly, it is important to see whether the impossibility result that we presented can be circumvented to create a voting rule that is in some sense usually hard to manipulate. We offered a few approaches at the end of Section 8.4 that could potentially create such a rule (without contradicting the impossibility result). Another direction for future research is to create new ways to tweak existing voting rules to make them harder to manipulate. (Elkind and Lipmaa [2005a] have already generalized the technique of adding a preround, showing that voting rules can be *hybridized* with each other to make manipulation harder.) It would be especially interesting to create tweaks that make the manipulation problem hard even with few candidates. Finally, it is important to study in more detail how hard the manipulation problem is when the nonmanipulators' votes are not exactly known, as this is typically the situation that manipulators are confronted with.

Much work also remains to be done on computing game-theoretic solutions (Chapter 9). In Section 9.2 we showed that computing Nash equilibria with certain properties is NP-hard. Are there any interesting properties for which this is not the case? To illustrate this, consider the following computational problem: find a Nash equilibrium with the property that there does not exist another Nash equilibrium in which each player's support is reduced by one strategy (relative to the former equilibrium). An equilibrium with this property can be found by finding any one Nash equilibrium, and then trying each possible way of reducing each player's support by one strategy (there are only polynomially many ways of doing so, and given the supports, finding a Nash equilibrium only requires solving a linear feasibility program). If we are successful in finding such a reduced Nash equilibrium, then we repeat the process with that equilibrium, until we fail. (It should be kept in mind that computing any one Nash equilibrium is still PPAD-complete.) Discovering properties of this kind can also help address the equilibrium selection problem (if there are multiple equilibria, which one should be played?), since one can argue that it is more natural to play an easy-to-compute equilibrium than to play a hard-to-compute equilibrium.

Various questions also remain on the generalized eliminability criterion introduced in Section 9.3. Are there other special cases (besides the case of small E sets) where it can be computed in polynomial time whether a strategy is eliminable? Are there alternative characterizations of this eliminability criterion (or restricted versions thereof)? Such alternative characterizations may make the criterion easier to understand, and may also lead to new approaches to computing whether a strategy can be eliminated. Perhaps it is also possible to create altogether different eliminability criteria. Such a criterion may be strictly weaker or stronger than the one presented here, so that eliminability in the one sense implies eliminability in the other sense; or the criteria may not be comparable. Another future research direction is to apply the eliminability technique in practice, testing it on random distributions of games such as those generated by GAMUT [Nudelman *et al.*, 2004], and using it to speed up computation of Nash equilibria (possibly on similar distributions of games).

Finally, other directions for future research that can be taken for any one of these solution concepts include computing solutions for restricted classes of games, as well as computing solutions under different game representations (including games of imperfect information, graphical games [Kearns *et al.*, 2001], action-graph games [Bhat and Leyton-Brown, 2004], *etc.*).

11.2.5 Node (4): Automated mechanism design for bounded agents

Automated mechanism design for bounded agents is in its infancy, and future research will likely create new and more comprehensive approaches. However, even the initial approach that we proposed—starting with a naïve mechanism and incrementally making it more strategy-proof—raises many questions. Most significantly, while we have given a general template for the technique, many choices must be made to fully instantiate it. For instance, we must choose the set of manipulations that we try to prevent, the way in which we try to prevent them, and when we terminate the updating process. We gave examples of various instantiations of the technique and the mechanisms that they generate, but ideally we would have a single instantiation of the technique that dominates all others. Even if this is not possible, it would be very helpful if we could provide some guidance as to which instantiation is likely to work best for a given application.

Another avenue for future research is to use this approach to help us create new general mechanisms. Whereas automated mechanism design in the sense of Chapter 6 can only help us conjecture truthful mechanisms, this approach potentially can also help us create new non-truthful mechanisms—for example, new voting rules. (Recall that all truthful voting rules are unsatisfactory by the Gibbard-Satterthwaite impossibility theorem.)

On the matter of designing algorithms for automatically executing the approach, it is important to find algorithms that scale to larger numbers of iterations. Not only will this allow us to design mechanisms with fewer possibilities for beneficial manipulation, but it will also make the remaining beneficial manipulations more difficult to find, because agents must reason through more iterations to find them. (Of course, if the agents have access to the more efficient algorithms as well, this benefit disappears. Nevertheless, if we as designers do not find more efficient algorithms, we run the risk that agents will find these algorithms themselves, and will be able to out-compute us and find beneficial manipulations.)

Bibliography

Tim Abbott, Daniel Kane, and Paul Valiant. On the complexity of two-player win-lose games. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2005.

Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, 2005.

K. Akcoglu, J. Aspnes, B. DasGupta, and M. Y. Kao. Opportunity cost algorithms for combinatorial auctions. *Applied Optimization: Computational Methods in Decision-Making, Economics and Finance*, pages 455–479, 2002.

Noga Alon. Ranking tournaments. *SIAM Journal of Discrete Mathematics*, 20:137–142, 2006.

Martin Andersson and Tuomas Sandholm. Time-quality tradeoffs in reallocative negotiation with combinatorial contract types. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 3–10, Orlando, FL, 1999.

Martin Andersson and Tuomas Sandholm. Contract type sequencing for reallocative negotiation. In *Proceedings of the Twentieth International Conference on Distributed Computing Systems*, Taipei, Taiwan, April 2000.

Krzysztof R. Apt. Uniform proofs of order independence for various strategy elimination procedures. *Contributions to Theoretical Economics*, 4(1), 2004.

Aaron Archer and Eva Tardos. Frugal path mechanisms. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 991–999, 2002.

Aaron Archer, Christos Papadimitriou, K Tawar, and Eva Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2003.

S. Areborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a K-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8:277–284, 1987.

Mark Armstrong. Optimal multi-object auctions. *Review of Economic Studies*, 67:455–481, 2000.

Kenneth Arrow. *Social choice and individual values*. New Haven: Cowles Foundation, 2nd edition, 1963. 1st edition 1951.

Kenneth Arrow. The property rights doctrine and demand revelation under incomplete information. In M Boskin, editor, *Economics and human welfare*. New York Academic Press, 1979.

Lawrence Ausubel and Paul Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1, 2002. No. 1, Article 1.

Lawrence M. Ausubel and Paul Milgrom. The lovely but lonely Vickrey auction. In Peter Cramton, Yoav Shoham, and Richard Steinberg, editors, *Combinatorial Auctions*, chapter 1. MIT Press, 2006.

Christopher Avery and Terrence Hendershott. Bundling and optimal auctions of multiple products. *Review of Economic Studies*, 67:483–497, 2000.

Jorgen Bang-Jensen and Carsten Thomassen. A polynomial algorithm for the 2-path problem for semicomplete digraphs. *SIAM Journal of Discrete Mathematics*, 5(3):366–376, 1992.

Yair Bartal, Rica Gonen, and Noam Nisan. Incentive compatible multi-unit combinatorial auctions. In *Theoretical Aspects of Rationality and Knowledge (TARK)*, Bloomington, Indiana, USA, 2003.

Yair Bartal, Rica Gonen, and Pierfrancesco La Mura. Negotiation-range mechanisms: Exploring the limits of truthful efficient markets. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 1–8, New York, NY, 2004.

John Bartholdi, III and James Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

John Bartholdi, III, Craig Tovey, and Michael Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.

John Bartholdi, III, Craig Tovey, and Michael Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989.

Nivan A. R. Bhat and Kevin Leyton-Brown. Computing Nash equilibria of action-graph games. In *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, Banff, Canada, 2004.

Darse Billings, Neil Burch, Aaron Davidson, Robert Holte, Jonathan Schaeffer, Terence Schauenberg, and Duane Szafron. Approximating game-theoretic optimal strategies for full-scale poker. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

Ben Blum, Christian R. Shelton, and Daphne Koller. A continuation method for Nash equilibria in structured games. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

Avrim Blum, Jeffrey Jackson, Tuomas Sandholm, and Martin Zinkevich. Preference elicitation and query learning. *Journal of Machine Learning Research*, 5:649–667, 2004. Early version in COLT-03.

Tilman Borgers. Pure strategy dominance. *Econometrica*, 61(2):423–430, 1993.

Craig Boutilier. Solving concisely expressed combinatorial auction problems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 359–366, Edmonton, Canada, 2002.

Sylvain Bouveret and Jérôme Lang. Efficiency and envy-freeness in fair division of indivisible goods: logical representation and complexity. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 935–940, Edinburgh, UK, 2005.

Felix Brandt and Tuomas Sandholm. (Im)Possibility of unconditionally privacy-preserving auctions. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 810–817, New York, NY, USA, 2004.

Felix Brandt and Tuomas Sandholm. On correctness and privacy in distributed mechanisms. In *Agent-Mediated Electronic Commerce (AMEC) workshop*, pages 1–14, New York, NY, 2004.

Felix Brandt and Tuomas Sandholm. Decentralized voting with unconditional privacy. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Utrecht, The Netherlands, 2005.

Felix Brandt and Tuomas Sandholm. Efficient privacy-preserving protocols for multi-unit auctions. In *Proceedings of the Financial Cryptography and Data Security conference(FC), Springer LNCS*, 2005.

Felix Brandt and Tuomas Sandholm. Unconditional privacy in social choice. In *Theoretical Aspects of Rationality and Knowledge (TARK)*, Singapore, 2005.

Andrew Byde. Applying evolutionary game theory to auction mechanism design. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 192–193, San Diego, CA, 2003. Poster paper.

Ruggiero Cavallo. Optimal decision-making with minimal waste: Strategyproof redistribution of VCG payments. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Hakodate, Japan, 2006.

Xi Chen and Xiaotie Deng. 3-Nash is PPAD-complete. *Electronic Colloquium on Computational Complexity*, Report No. 134, 2005.

Xi Chen and Xiaotie Deng. Settling the complexity of 2-player Nash equilibrium. *Electronic Colloquium on Computational Complexity*, Report No. 150, 2005.

Ed H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

Dave Cliff. Evolution of market mechanism through a continuous space of auction-types. Technical Report HPL-2001-326, HP Labs, 2001.

William Cohen, Robert Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:213–270, 1999.

Wolfram Conen and Tuomas Sandholm. Preference elicitation in combinatorial auctions: Extended abstract. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 256–259, Tampa, FL, October 2001. A more detailed description of the algorithmic aspects appeared in the IJCAI-2001 Workshop on Economic Agents, Models, and Mechanisms, pp. 71–80.

Wolfram Conen and Tuomas Sandholm. Partial-revelation VCG mechanism for combinatorial auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 367–372, Edmonton, Canada, 2002.

Vincent Conitzer and Nikesh Garera. Learning algorithms for online principal-agent problems (and selling goods online). In *International Conference on Machine Learning (ICML)*, Pittsburgh, PA, USA, 2006.

Vincent Conitzer and Tuomas Sandholm. Complexity of manipulating elections with few candidates. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 314–319, Edmonton, Canada, 2002.

Vincent Conitzer and Tuomas Sandholm. Complexity of mechanism design. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 103–110, Edmonton, Canada, 2002.

Vincent Conitzer and Tuomas Sandholm. Vote elicitation: Complexity and strategy-proofness. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 392–397, Edmonton, Canada, 2002.

Vincent Conitzer and Tuomas Sandholm. Applications of automated mechanism design. In *UAI-03 workshop on Bayesian Modeling Applications*, Acapulco, Mexico, 2003.

Vincent Conitzer and Tuomas Sandholm. Automated mechanism design: Complexity results stemming from the single-agent setting. In *Proceedings of the 5th International Conference on Electronic Commerce (ICEC-03)*, pages 17–24, Pittsburgh, PA, USA, 2003.

Vincent Conitzer and Tuomas Sandholm. Automated mechanism design with a structured outcome space, 2003.

Vincent Conitzer and Tuomas Sandholm. BL-WoLF: A framework for loss-bounded learnability in zero-sum games. In *International Conference on Machine Learning (ICML)*, pages 91–98, Washington, DC, USA, 2003.

Vincent Conitzer and Tuomas Sandholm. Complexity results about Nash equilibria. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 765–771, Acapulco, Mexico, 2003.

Vincent Conitzer and Tuomas Sandholm. Definition and complexity of some basic metareasoning problems. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1099–1106, Acapulco, Mexico, 2003.

Vincent Conitzer and Tuomas Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 781–788, Acapulco, Mexico, 2003.

Vincent Conitzer and Tuomas Sandholm. An algorithm for automatically designing deterministic mechanisms without payments. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 128–135, New York, NY, USA, 2004.

Vincent Conitzer and Tuomas Sandholm. Communication complexity as a lower bound for learning in games. In *International Conference on Machine Learning (ICML)*, pages 185–192, Banff, Alberta, Canada, 2004.

Vincent Conitzer and Tuomas Sandholm. Computational criticisms of the revelation principle. In *The Conference on Logic and the Foundations of Game and Decision Theory (LOFT-04)*, Leipzig, Germany, 2004. Earlier versions appeared as a short paper at ACM-EC-04, and in the workshop on Agent-Mediated Electronic Commerce (AMEC-03).

Vincent Conitzer and Tuomas Sandholm. Computing Shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 219–225, San Jose, CA, 2004. Earlier version: IJCAI-03 workshop on Distributed Constraint Reasoning (DCR-03), Acapulco, Mexico.

Vincent Conitzer and Tuomas Sandholm. Expressive negotiation over donations to charities. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 51–60, New York, NY, 2004.

Vincent Conitzer and Tuomas Sandholm. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 132–141, New York, NY, 2004. Early version appeared as a poster in the ACM Conference on Electronic Commerce, 2003, pp. 232–233.

Vincent Conitzer and Tuomas Sandholm. Common voting rules as maximum likelihood estimators. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 145–152, Edinburgh, UK, 2005.

Vincent Conitzer and Tuomas Sandholm. Communication complexity of common voting rules. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 78–87, Vancouver, Canada, 2005.

Vincent Conitzer and Tuomas Sandholm. Complexity of (iterated) dominance. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 88–97, Vancouver, Canada, 2005.

Vincent Conitzer and Tuomas Sandholm. Expressive negotiation in settings with externalities. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 255–260, Pittsburgh, PA, 2005.

Vincent Conitzer and Tuomas Sandholm. A generalized strategy eliminability criterion and computational methods for applying it. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 483–488, Pittsburgh, PA, 2005.

Vincent Conitzer and Tuomas Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 2006. Short version in ICML-03.

Vincent Conitzer and Tuomas Sandholm. Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence*, 170(6-7):607–619, 2006. Earlier version appeared in IJCAI-03, pages 613–618.

Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, Ann Arbor, MI, 2006.

Vincent Conitzer and Tuomas Sandholm. Failures of the VCG mechanism in combinatorial auctions and exchanges. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 521–528, Hakodate, Japan, 2006. Early versions appeared at the AAMAS-04 Agent-Mediated Electronic Commerce (AMEC) workshop, and (as a short paper) at ACM-EC-04.

Vincent Conitzer and Tuomas Sandholm. Incrementally making mechanisms more strategy-proof. In *Multidisciplinary Workshop on Advances in Preference Handling*, Riva del Garda, Italy, 2006.

Vincent Conitzer and Tuomas Sandholm. Nonexistence of voting rules that are usually hard to manipulate. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Boston, MA, 2006.

Vincent Conitzer and Tuomas Sandholm. A technique for reducing normal-form games to compute a Nash equilibrium. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 537–544, Hakodate, Japan, 2006. Early version appeared in IJCAI Workshop on Game Theoretic and Decision Theoretic Agents (GTDT), 2005.

Vincent Conitzer, Jérôme Lang, and Tuomas Sandholm. How many candidates are needed to make elections hard to manipulate? In *Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 201–214, Bloomington, Indiana, USA, 2003.

Vincent Conitzer, Jonathan Derryberry, and Tuomas Sandholm. Combinatorial auctions with structured item graphs. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 212–218, San Jose, CA, 2004.

Vincent Conitzer, Tuomas Sandholm, and Paolo Santi. Combinatorial auctions with k-wise dependent valuations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 248–254, Pittsburgh, PA, 2005. Draft in Oct., 2003.

Vincent Conitzer, Andrew Davenport, and Jayant Kalagnanam. Improved bounds for computing Kemeny rankings. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Boston, MA, 2006. Early version presented at INFORMS-05.

Vincent Conitzer. Computing Slater rankings using similarities among candidates. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Boston, MA, 2006. Early version appeared as IBM RC 23748, 2005.

Don Coppersmith, Lisa Fleischer, and Atri Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction to Algorithms*. MIT Press, 1990.

George Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

R. K. Dash, S. D. Ramchurn, and N. R. Jennings. Trust-based mechanism design. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 748–755, New York, NY, USA, 2004.

Constantinos Daskalakis and Christos Papadimitriou. Three-player games are hard. *Electronic Colloquium on Computational Complexity*, Report No. 139, 2005.

Constantinos Daskalakis, Paul Goldberg, and Christos Papadimitriou. The complexity of computing a Nash equilibrium. *Electronic Colloquium on Computational Complexity*, Report No. 115, 2005.

Claude d'Aspremont and Louis-Andre Gérard-Varet. Incentives and incomplete information. *Journal of Public Economics*, 11:25–45, 1979.

Andrew Davenport and Jayant Kalagnanam. A computational study of the Kemeny rule for preference aggregation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 697–702, San Jose, CA, 2004.

Marie Jean Antoine Nicolas de Caritat (Marquis de Condorcet). Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix. 1785. Paris: L'Imprimerie Royale.

Sven de Vries and Rakesh Vohra. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, 15(3):284–309, 2003.

Sven de Vries, James Schummer, and Rakesh V. Vohra. On ascending auctions for heterogeneous objects, 2003. Draft, Nov.

Christine DeMartini, Anthony Kwasnica, John Ledyard, and David Porter. A new and improved design for multi-object iterative auctions. Technical Report 1054, California Institute of Technology, Social Science, September 1999.

John Dickhaut and Todd Kaplan. A program for finding Nash equilibria. *The Mathematica Journal*, pages 87–93, 1991.

Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th World Wide Web Conference*, pages 613–622, 2001.

eBay UK. Proxy bidding. 2004. http://pages.ebay.co.uk/help/buyerguide/bidding-prxy.html.

Edith Elkind and Helger Lipmaa. Hybrid voting protocols and hardness of manipulation. In *Annual International Symposium on Algorithms and Computation (ISAAC)*, 2005.

Edith Elkind and Helger Lipmaa. Small coalitions cannot manipulate voting. In *Proceedings of the Financial Cryptography and Data Security conference(FC)*, 2005.

Elaine Eschen and Jeremy Spinrad. An $O(n^2)$ algorithm for circular-arc graph recognition. In *Annual SIAM-ACM Symposium on Discrete Algorithms (SODA)*, pages 128–137, 1993.

Joan Feigenbaum, Christos Papadimitriou, and Scott Shenker. Sharing the cost of mulicast transmissions. *Journal of Computer and System Sciences*, 63:21–41, 2001. Early version in STOC-00.

Lance Fortnow, Joe Kilian, David M. Pennock, and Michael P. Wellman. Betting boolean-style: a framework for trading in securities based on logical formulas. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 144–155, San Diego, CA, 2003.

Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, 1991.

Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 548–553, Stockholm, Sweden, August 1999.

David Gale, Harold W. Kuhn, and Albert W. Tucker. Linear programming and the theory of games. In Tjalling Koopmans, editor, *Activity Analysis of Allocation and Production*. 1951.

Michael Garey and David Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.

Michael Garey, David Johnson, and Larry Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

Allan Gibbard. Manipulation of voting schemes. *Econometrica*, 41:587–602, 1973.

Allan Gibbard. Manipulation of schemes that mix voting with chance. *Econometrica*, 45:665–681, 1977.

Itzhak Gilboa and Eitan Zemel. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1:80–93, 1989.

Itzhak Gilboa, Ehud Kalai, and Eitan Zemel. On the order of eliminating dominated strategies. *Operations Research Letters*, 9:85–89, 1990.

Itzhak Gilboa, Ehud Kalai, and Eitan Zemel. The complexity of eliminating dominated strategies. *Mathematics of Operation Research*, 18:553–565, 1993.

Andrew Gilpin and Tuomas Sandholm. A competitive Texas Hold'em poker player via automated abstraction and real-time equilibrium computation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Boston, MA, 2006.

Andrew Gilpin and Tuomas Sandholm. Finding equilibria in large sequential games of imperfect information. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, Ann Arbor, MI, 2006.

Andrew Goldberg and Jason Hartline. Envy-free auctions for digital goods. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 29–35, San Diego, CA, 2003.

Devorah Goldburg and Shannon McElligott. Red cross statement on official donation locations. 2001. Press release, http://www.redcross.org/press/disaster/ds_pr/011017legitdonors.html.

Rica Gonen and Daniel Lehmann. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 13–20, Minneapolis, MN, October 2000.

Georg Gottlob, Gianluigi Greco, and Francesco Scarcello. Pure Nash equilibria: hard and easy games. In *Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 215–230, Bloomington, Indiana, USA, 2003.

J Green and J-J Laffont. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica*, 45:427–438, 1977.

Theodore Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

Hongwei Gui, Rudolf Müller, and Rakesh Vohra. Characterizing dominant strategy mechanisms with multi-dimensional types, 2004. Working Paper.

E. Hemaspaandra and L. Hemaspaandra. Dichotomy for voting systems. Technical Report 861, University of Rochester, Department of Computer Science, 2005.

E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM*, 44(6):806–825, 1997.

John Hershberger and Subhash Suri. Vickrey prices and shortest paths: What is an edge worth? In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2001.

Douglas Hofstadter. *Metamagical Themas*. Basic Books, 1985.

Bengt Holmström. Groves' scheme on restricted domains. *Econometrica*, 47(5):1137–1144, 1979.

Holger Hoos and Craig Boutilier. Solving combinatorial auctions using stochastic local search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 22–29, Austin, TX, August 2000.

Benoit Hudson and Tuomas Sandholm. Effectiveness of query types and policies for preference elicitation in combinatorial auctions. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 386–393, New York, NY, USA, 2004. Early versions: CMU tech report CMU-CS-02-124, AMEC-02, SITE-02.

Nathanaël Hyafil and Craig Boutilier. Regret-based incremental partial revelation mechanisms. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Boston, MA, 2006.

Takayuki Ito, Makoto Yokoo, and Shigeo Matsubara. Designing an auciton protocol under asymmetric information on nature's selection. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 61–68, Bologna, Italy, 2002.

Takayuki Ito, Makoto Yokoo, and Shigeo Matsubara. Toward a combinatorial auction protocol among experts and amateurs: The case of single-skilled experts. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 481–488, Melbourne, Australia, 2003.

Takayuki Ito, Makoto Yokoo, and Shigeo Matsubara. A combinatorial auction among versatile experts and amateurs. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 378–385, New York, NY, USA, 2004.

Sergei Izmalkov, Matt Lepinski, and Silvio Micali. Universal mechanism design. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2005.

Philippe Jehiel and Benny Moldovanu. How (not) to sell nuclear weapons. *American Economic Review*, 86(4):814–829, 1996.

Radu Jurca and Boi Faltings. Minimum payments that reward honest reputation feedback. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, Ann Arbor, MI, 2006.

Richard Karp. Reducibility among combinatorial problems. In Raymond E Miller and James W Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, NY, 1972.

Michael Kearns, Michael Littman, and Satinder Singh. Graphical models for game theory. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2001.

John Kemeny. Mathematics without numbers. In *Daedalus*, volume 88, pages 571–591. 1959.

Leonid Khachiyan. A polynomial algorithm in linear programming. *Soviet Math. Doklady*, 20:191–194, 1979.

Christopher Kiekintveld, Yevgeniy Vorobeychik, and Michael Wellman. An analysis of the 2004 supply chain management trading agent competition. In *IJCAI-05 Workshop on Trading Agent Design and Analysis*, Edinburgh, UK, 2005.

Donald E. Knuth, Christos H. Papadimitriou, and John N Tsitsiklis. A note on strategy elimination in bimatrix games. *Operations Research Letters*, 7(3):103–107, 1988.

R Kohli, R Krishnamurthi, and P Mirchandani. The minimum satisfiability problem. *SIAM Journal of Discrete Mathematics*, 7(2):275–283, 1994.

Daphne Koller and Avi Pfeffer. Representations and solutions for game-theoretic problems. *Artificial Intelligence*, 94(1):167–215, July 1997.

Norbert Korte and Rolf Mohring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM Journal on Computing*, 18(1):68–81, February 1989.

Anshul Kothari, David Parkes, and Subhash Suri. Approximately-strategyproof and tractable multi-unit auctions. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 166–175, San Diego, CA, 2003.

Sebastién Lahaie and David Parkes. Applying learning algorithms to preference elicitation. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, New York, NY, 2004.

Kate Larson and Tuomas Sandholm. Bargaining with limited computation: Deliberation equilibrium. *Artificial Intelligence*, 132(2):183–217, 2001. Short early version appeared in the Proceedings of the National Conference on Artificial Intelligence (AAAI), pp. 48–55, Austin, TX, 2000.

Kate Larson and Tuomas Sandholm. Costly valuation computation in auctions. In *Theoretical Aspects of Rationality and Knowledge (TARK VIII)*, pages 169–182, Siena, Italy, July 2001.

Kate Larson and Tuomas Sandholm. Mechanism design for deliberative agents. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Utrecht, The Netherlands, 2005. Early versions appeared as *Designing Auctions for Deliberative Agents* at AMEC-04, and *Strategic Deliberation and Truthful Revelation: An Impossibility Result* at ACM-EC-04 (short paper).

Ron Lavi, Ahuva Mu'Alem, and Noam Nisan. Towards a characterization of truthful combinatorial auctions. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 574–583, 2003.

Daniel Lehmann, Lidian Ita O'Callaghan, and Yoav Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002. Early version appeared in ACMEC-99.

Carlton Lemke and J. Howson. Equilibrium points of bimatrix games. *Journal of the Society of Industrial and Applied Mathematics*, 12:413–423, 1964.

Kevin Leyton-Brown and Moshe Tennenholtz. Local-effect games. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

Anton Likhodedov and Tuomas Sandholm. Auction mechanism for optimally trading off efficiency and revenue. In *Agent-Mediated Electronic Commerce (AMEC) workshop*, Melbourne, Australia, 2003. A short version also appeared in the ACM Conference on Electronic Commerce, 2003. The extension to multi-unit auctions has been accepted as a short paper in the ACM Conference on Electronic Commerce, 2004.

Anton Likhodedov and Tuomas Sandholm. Methods for boosting revenue in combinatorial auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 232–237, San Jose, CA, 2004.

Anton Likhodedov and Tuomas Sandholm. Approximating revenue-maximizing combinatorial auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Pittsburgh, PA, 2005.

Richard Lipton, Evangelos Markakis, Elchanan Mossel, and Amin Saberi. On approximately fair allocations of indivisible goods. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 125–131, New York, NY, 2004.

William S. Lovejoy. Optimal mechanisms with finite agent types. *Management Science*, 53(5):788–803, 2006.

Leslie M. Marx and Jeroen M. Swinkels. Order independence for iterated weak dominance. *Games and Economic Behavior*, 18:219–245, 1997.

Leslie M. Marx and Jeroen M. Swinkels. Corrigendum, order independence for iterated weak dominance. *Games and Economic Behavior*, 31:324–329, 2000.

Andreu Mas-Colell, Michael Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

Eric Maskin and John Riley. Optimal multi-unit auctions. In Frank Hahn, editor, *The Economics of Missing Markets, Information, and Games*, chapter 14, pages 312–335. Clarendon Press, Oxford, 1989.

Andrew McLennan and In-Uck Park. Generic 4x4 two person games have at most 15 Nash equilibria. *Games and Economic Behavior*, pages 26–1,111–130, 1999.

Andrew McLennan. The expected number of Nash equilibria of a normal form game. *Econometrica*, 1999.

Dov Monderer and Moshe Tennenholtz. Asymptotically optimal multi-object auctions for risk-averse agents. Technical report, Faculty of Industrial Engineering and Management, Technion, Haifa, Israel, February 1999.

Hervé Moulin. Serial cost-sharing of excludable public goods. *Review of Economic Studies*, 61:305–325, 1994.

Ahuva Mu'alem and Noam Nisan. Truthful approximate mechanisms for restricted combinatorial auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 379–384, Edmonton, Canada, July 2002.

Roger Myerson and Mark Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 28:265–281, 1983.

Roger Myerson. Incentive compatibility and the bargaining problem. *Econometrica*, 41(1), 1979.

Roger Myerson. Optimal auction design. *Mathematics of Operation Research*, 6:58–73, 1981.

Roger Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, Cambridge, 1991.

John Nash. Equilibrium points in n-person games. *Proc. of the National Academy of Sciences*, 36:48–49, 1950.

George Nemhauser and Laurence Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, 1999. Section 4, page 11.

Noam Nisan and Amir Ronen. Computationally feasible VCG mechanisms. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 242–252, Minneapolis, MN, 2000.

Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001. Early version in STOC-99.

Noam Nisan and Ilya Segal. The communication requirements of efficient allocations and supporting prices. *Journal of Economic Theory*, 2005. Forthcoming.

Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 1–12, Minneapolis, MN, 2000.

Robert Nozick. Newcomb's problem and two principles of choice. In Nicholas Rescher et al., editor, *Essays in Honor of Carl G. Hempel*, pages 114–146. Synthese Library (Dordrecht, the Netherlands: D. Reidel), 1969.

Eugene Nudelman, Jennifer Wortman, Kevin Leyton-Brown, and Yoav Shoham. Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, New York, NY, USA, 2004.

Naoki Ohta, Atsushi Iwasaki, Makoto Yokoo, Kohki Maruono, Vincent Conitzer, and Tuomas Sandholm. A compact representation scheme for coalitional games in open anonymous environments. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Boston, MA, 2006.

Martin J Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

Christos Papadimitriou and Tim Roughgarden. Equilibria in symmetric games. 2003. Available at http://www.cs.berkeley.edu/˜ christos/papers/sym.ps.

Christos Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31:288–301, 1985.

Christos Papadimitriou. NP-completeness: A retrospective. In *Proceedings of the International Conference on Automata, Languages, and Programming (ICALP)*, 1997.

Christos Papadimitriou. Algorithms, games and the Internet. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 749–753, 2001.

David Parkes and Grant Schoenebeck. GROWRANGE: Anytime VCG-based mechanisms. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 34–41, San Jose, CA, 2004.

David Parkes and Jeffrey Shneidman. Distributed implementations of generalized Vickrey-Clarke-Groves auctions. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 261–268, New York, NY, USA, 2004.

David Parkes, Jayant Kalagnanam, and Marta Eso. Achieving budget-balance with Vickrey-based payment schemes in exchanges. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1161–1168, Seattle, WA, 2001.

David Parkes, Ruggiero Cavallo, Nick Elprin, Adam Juda, Sebastien Lahaie, Benjamin Lubin, Loizos Michael, Jeffrey Shneidman, and Hassan Sultan. ICE: An iterative combinatorial exchange. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, Vancouver, Canada, 2005.

David Parkes. iBundle: An efficient ascending price bundle auction. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 148–157, Denver, CO, November 1999.

David Parkes. Optimal auction design for agents with hard valuation problems. In *Agent-Mediated Electronic Commerce Workshop at the International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.

David G. Pearce. Rationalizable strategic behavior and the problem of perfection. *Econometrica*, 52:1029–1050, 1984.

Michal Penn and Moshe Tennenholtz. Constrained multi-object auctions and b-matching. *Information Processing Letters*, 75(1–2):29–34, July 2000.

Adrian Petcu, Boi Faltings, and David Parkes. Mdpop: Faithful distributed implementation of efficient social choice problems. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Hakodate, Japan, 2006.

Steve Phelps, Peter McBurnley, Simon Parsons, and Elizabeth Sklar. Co-evolutionary auction mechanism design. *Lecture Notes in AI*, 2531, 2002.

M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Aggregating partially ordered preferences: possibility and impossibility results. In *Theoretical Aspects of Rationality and Knowledge (TARK)*, Singapore, 2005.

Ryan Porter, Amir Ronen, Yoav Shoham, and Moshe Tennenholtz. Mechanism design with execution uncertainty. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, Edmonton, Canada, 2002.

Ryan Porter, Eugene Nudelman, and Yoav Shoham. Simple search methods for finding a Nash equilibrium. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 664–669, San Jose, CA, 2004.

Ryan Porter. Mechanism design for online real-time scheduling. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 61–70, New York, NY, 2004.

Ariel D. Procaccia and Jeffrey S. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 497–504, Hakodate, Japan, 2006.

F. Rossi, M. S. Pini, K. B. Venable, and T. Walsh. Strategic voting when aggregating partially ordered preferences. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 685–687, Hakodate, Japan, 2006.

J. Rothe, H. Spakowski, and J. Vogel. Exact complexity of the winner problem for Young elections. In *Theory of Computing Systems*, volume 36(4), pages 375–386. Springer-Verlag, 2003.

Michael Rothkopf, Thomas Teisberg, and Edward Kahn. Why are Vickrey auctions rare? *Journal of Political Economy*, 98(1):94–109, 1990.

Michael Rothkopf, Aleksandar Pekeč, and Ronald Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2003.

Tuomas Sandholm and Andrew Gilpin. Sequences of take-it-or-leave-it offers: Near-optimal auctions without full valuation revelation. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1127–1134, Hakodate, Japan, 2006.

Tuomas Sandholm and Victor R Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, pages 328–335, San Francisco, CA, June 1995. Reprinted in *Readings in Agents*, Huhns and Singh, eds., pp. 66–73, 1997.

Tuomas Sandholm and Victor R Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1):99–137, 1997. Special issue on Economic Principles of Multiagent Systems. Early version appeared at the International Joint Conference on Artificial Intelligence (IJCAI), pages 662–669, 1995.

Tuomas Sandholm and Victor Lesser. Leveled commitment contracting: A backtracking instrument for multiagent systems. *AI Magazine*, 23(3):89–100, 2002.

Tuomas Sandholm and Subhash Suri. BOB: Improved winner determination in combinatorial auctions and generalizations. *Artificial Intelligence*, 145:33–58, 2003. Early version: Improved Algorithms for Optimal Winner Determination in Combinatorial Auctions and Generalizations. National Conference on Artificial Intelligence (AAAI-00), pp. 90–97, Austin, TX, July 31 – August 2.

Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Winner determination in combinatorial auction generalizations. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 69–76, Bologna, Italy, July 2002. Early version appeared at the AGENTS-01 Workshop on Agent-Based Approaches to B2B, pp. 35–41, Montreal, Canada, May 2001.

Tuomas Sandholm, Vincent Conitzer, and Craig Boutilier. Automated design of multistage mechanisms. In *First International Workshop on Incentive Based Computing, at the IEEE / WIC / ACM International Conference on Web Intelligence (WI)*, Compiegne, France, 2005.

Tuomas Sandholm, Andrew Gilpin, and Vincent Conitzer. Mixed-integer programming methods for finding Nash equilibria. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 495–501, Pittsburgh, PA, 2005.

Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. CABOB: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science*, 51(3):374–390, 2005. Special issue on Electronic Markets. Early version in IJCAI-01.

Tuomas Sandholm, David Levine, Michael Concordia, Paul Martyn, Rick Hughes, Jim Jacobs, and Dennis Begg. Changing the game in strategic sourcing at Procter & Gamble: Expressive competition enabled by optimization. *Interfaces*, 36(1):55–68, 2006. Edelman award competition finalist writeup.

Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 256–262, Washington, D.C., July 1993.

Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 256–262, Washington, D.C., 1993.

Tuomas Sandholm. Necessary and sufficient contract types for optimal task allocation. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI)*, page 87, Nagoya, Japan, 1997. Poster session abstracts.

Tuomas Sandholm. Issues in computational Vickrey auctions. *International Journal of Electronic Commerce*, 4(3):107–129, 2000. Special Issue on Applying Intelligent Agents for Electronic Commerce. A short, early version appeared at the Second International Conference on Multi–Agent Systems (ICMAS), pages 299–306, 1996.

Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, January 2002. First appeared as an invited talk at the First International Conference on Information and Computation Economies, Charleston, SC, Oct. 25–28, 1998. Extended version appeared as Washington Univ., Dept. of Computer Science, tech report WUCS-99-01, January 28th, 1999. Conference version appeared at the International Joint Conference on Artificial Intelligence (IJCAI), pp. 542–547, Stockholm, Sweden, 1999.

Tuomas Sandholm. eMediator: A next generation electronic commerce server. *Computational Intelligence*, 18(4):656–676, 2002. Special issue on Agent Technology for Electronic Commerce. Early versions appeared in the Conference on Autonomous Agents (AGENTS-00), pp. 73–96, 2000; AAAI-99 Workshop on AI in Electronic Commerce, Orlando, FL, pp. 46–55, July 1999; and as a Washington University, St. Louis, Dept. of Computer Science technical report WU-CS-99-02, Jan. 1999.

Tuomas Sandholm. Expressive commerce and its application to sourcing. In *Conference on Innovative Applications of Artificial Intelligence*, Boston, MA, July 2006.

Paolo Santi, Vincent Conitzer, and Tuomas Sandholm. Towards a characterization of polynomial preference elicitation with value queries in combinatorial auctions. In *Conference on Learning Theory (COLT)*, pages 1–16, Banff, Alberta, Canada, 2004.

Mark Satterthwaite. Strategy-proofness and Arrow's conditions: existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

Rahul Savani and Bernhard von Stengel. Exponentially many steps for finding a Nash equilibrium in a bimatrix game. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 258–267, 2004.

Grant Schoenebeck and Salil Vadhan. The computational complexity of Nash equilibria in concisely represented games. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 270–279, Ann Arbor, MI, 2006.

Jiefu Shi and Michael Littman. Abstraction methods for game theoretic poker. In *Computers and Games*, pages 333–345. Springer-Verlag, 2001.

John Tagliabue. Global AIDS Funds Is Given Attention, but Not Money. *The New York Times*, 2003. Reprinted on http://www.healthgap.org/press_releases/a03/060103_NYT_HGAP_G8_fund.html.

Moshe Tennenholtz. Some tractable combinatorial auctions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, Austin, TX, August 2000.

Leslie Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

Stan van Hoesel and Rudolf Müller. Optimization in electronic marketplaces: Examples from combinatorial auctions. *Netnomics*, 3(1):23–33, June 2001.

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

Rakesh Vohra. Research problems in combinatorial auctions. Mimeo, version Oct. 29, 2001.

John von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1947.

John von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1927.

Yevgeniy Vorobeychik, Christopher Kiekintveld, and Michael Wellman. Empirical mechanism design: Methods, with application to a supply chain scenario. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, Ann Arbor, MI, 2006.

Peter Wurman and Michael Wellman. AkBA: A progressive, anonymous-price combinatorial auction. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 21–29, Minneapolis, MN, October 2000.

Makoto Yokoo, Yuko Sakurai, and Shigeo Matsubara. Robust combinatorial auction protocol against false-name bids. *Artificial Intelligence*, 130(2), 2004.

Makoto Yokoo, Vincent Conitzer, Tuomas Sandholm, Naoki Ohta, and Atsushi Iwasaki. Coalitional games in open anonymous environments. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 509–514, Pittsburgh, PA, 2005.

Makoto Yokoo. The characterization of strategy/false-name proof combinatorial auction protocols: Price-oriented, rationing-free protocol. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 733–742, Acapulco, Mexico, August 2003.

Peyton Young. Optimal voting rules. *Journal of Economic Perspectives*, 9(1):51–64, 1995.

Martin Zinkevich, Avrim Blum, and Tuomas Sandholm. On polynomial-time preference elicitation with value queries. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 176–185, San Diego, CA, 2003.

Edo Zurel and Noam Nisan. An efficient approximate allocation algorithm for combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, pages 125–136, Tampa, FL, 2001.