Reducing Truth-telling Online Mechanisms to Online Optimization

Baruch Awerbuch [1] Yossi Azar [2] Adam Meyerson [3]

January 2003

CMU-CS-02-209

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1] Johns Hopkins University, Baltimore, MD 21218. E-mail: baruch@cs.jhu.edu

[2] Department of Computer Science, Tel Aviv University, Tel-Aviv, 69978, Israel. E-Mail: azar@tau.ac.il. Research supported in part by the Israel Science Foundation

[3] Department of Computer Science, Carnegie-Mellon University. Email: adam@cs.cmu.edu. Research partially supported by ARO grant DAAG55-98-1-0170 and by the CMU Aladdin project.

Abstract

We describe a general technique for converting an online algorithm $\mathcal{B}$ to a truthtelling mechanism. We require that the original online competitive algorithm has certain "niceness" properties in that actions on future requests are independent of the actual value of requests which were accepted (though these actions will of course depend upon the set of accepted requests). Under these conditions, we are able to give an online truth telling mechanism (where the values of requests are given by bids which may not accurately represent the valuation of the requesters) such that our total profit is within $O(\rho+\log\mu)$ of the optimum offline profit obtained by an omniscient algorithm (one which knows the true valuations of the users). Here ρ is the competitive ratio of $\mathcal{B}$ for the optimization version of the problem, and μ is the ratio of the maximum to minimum valuation for a request. In general there is an $\Omega(\log\mu)$ lower bound on the ratio of worst-case profit for a truth telling mechanism when compared to the profit obtained by an omniscient algorithm, so this result is in some sense best possible. In addition, we prove that our construction is resilient against many forms of "cheating" attempts, such as forming coalitions.
We demonstrate applications of this result to several problems. We develop online truthtelling mechanisms for online routing and admission control of path or multicast requests, assuming large network capacities. Assuming the existance of an algorithm $\mathcal{B}$ for the optimization version of the problem, our techniques provide truthtelling mechanisms for general combinatorial auctions. However, designing optimization algorithms may be difficult in general because of online or approximation lower bounds. For the cases described above, we are able to design optimization algorithms $\mathcal{B}$ by amortizing the lost benefit from online computation (and from approximation hardness in the case of multicast) against the benefit obtained from accepted requests.
We comment that our upper bounds on profit competitiveness imply, as an obvious corollary, similar bound on *global efficiency*, namely overall well-being of all the users. This contrasts with most other work on truthtelling mechanisms for general online resource allocation, where only efficiency is maximized, and competitiveness can be arbitrarily poor.

Keywords: mechanism design, online algorithms, approximations, multicast routing

1 Introduction

We construct a general technique for converting an algorithm for online optimization to a truthtelling mechanism. We will draw examples from optimization problems such as online routing [3, 4] and construction of multicast trees [6, 5, 12]. For the most part, the optimization algorithms described in those papers may be used as black boxes, with our protocol layered atop the algorithm to guarantee truthful behavior by rational agents. Online optimization is a broad topic in computer science literature, and there are many other examples to which our techniques could be applied.

Truthtelling mechanism design is a longstanding problem in economics and gametheory, with one of the earlier examples being the Vickrey auction [18] and VCG mechanisms [18, 9, 14]. In computer science theory, recent work has given results describing truthtelling mechanisms for shortest path [1], multicast [11], load balancing [17], allocation of goods [15, 7], and allocation of digital goods [8, 13]. Many of these problems can be viewed in an online setting (where requests arrive one at a time in an adversarial fashion) and only some of the earlier results consider the online scenario.

There are two separate issues in dealing with combinatorial auctions:

- *existence:* do profit-competitive truthtelling online mechanism designs even *exist*?
- *computatability:* if so, can they be computed in polynomial time?

Our main contribution is the introduction of a general technique to transform various online optimization problems into truthtelling mechanisms. This transformation introduces only an *additive* loss in the competitive ratio (a multiplicative loss is relatively straightforward), enabling us to produce best-possible competitive bounds for a variety of problems of which the specific instances presented are only a sample.

The most general result is a truthtelling mechanism for general online combinatorial auctions, assuming an online optimization function exists. In general, combinatorial auctions require exponential size representation of input and exponential computation effort.

As applications of our general framework we present constructive results for interesting special cases, where internal structure of the problem enables polynomial representation of the input, as well as polynomial approximation in spite of exponential number of options. Specifically, we give first online polynomial-time optimally-competitive truthtelling mechanisms for a number of basic online optimization problems in capacitated networks, including:

- constrained network routing and admission control, where selection must be made among exponentially many paths.
- multicast routing, where steiner trees need to be selected, among exponentially many trees.
- multicast admission control, where multicast requests to be admitted must be selected among exponential number of possibilities.
- combined multicast routing and admission control, combining all of the problems above.

The latter three results may be contrasted against various hardness results [10] for the same problems.

Our techniques assume the existence of an online optimization algorithm. In the case of combinatorial auctions, no such algorithm can have a reasonable competitive ratio in the general setting. This may be seen via a reduction from online routing on a unit-capacity network. However, if we assume large supply of each item in the combinatorial auction, it is straightforward to apply the algorithm of [3] to the optimization version of the problem. Alternately, some auctions with additional constraints (for example each customer requests only a small subset of items) may permit such an optimization algorithm.

In the case of multicast, the hardness results arise from optimizing the function of benefit minus cost. Since we cannot solve steiner tree exactly in polynomial time, we cannot obtain any reasonable approximation to the difference problem [10]. However, we can exploit the fact that in a capacitated network there is no direct notion of externally-imposed cost, and we only need to worry about internally-maintained “opportunity cost”. The error in optimization of opportunity cost because of computational hardness can effectively be amortized against the benefit gained from earlier multicast routings if the capacities are reasonably large.

2 Model and problem statement

Single-option auctions. We consider serving a sequence of requests which arrive one at a time online. Each request may be represented by a pair (r_i, b_i). Here r_i is a statement of the resources requested; this may be in many different forms (for example it might be a source-sink pair in a communication network, a node to be added to a multicast tree, or in general a list of feasible subsets of the products to be auctioned). The variable b_i is a bid, which represents the amount of money the requester is willing to pay the auctioneer if the requested resources are provided. As each request arrives, we have several decisions to make. We must decide whether to accept or reject the request. If we accept, there may be several feasible ways to satisfy r_i, and we must select one such way. Finally, we must determine a price $p_i \leq b_i$ to charge the user for our services.

In general we assume each user has a valuation v_i, which represents the amount he is really willing to pay for request i. The bid b_i will be computed in order to optimize user i's benefit (benefit is zero if the request is rejected and $v_i - p_i$ if the request is accepted and the user is charged price p_i). An algorithm is defined to be *truthtelling* if each user computed $b_i = v_i$ when trying to optimize benefit.

General Multiple-option (combinatorial) auctions. We consider the case of combinatorial auctions, in which we are simultaneously auctioning a number of distinct items. Each customer places bids on several subsets of the items up for auction. Our goal is to provide each customer with one of the subsets he requested (or an empty subset) and charge some price, such that we do not oversell our supply of any of the auctioned items and our income is maximized. This problem does not fit the framework outlined above, in that each customer may have several distinct bids, and we must determine not only whether to accept the customer but which subset of the items to give him. We will again represent incoming requests by pairs (r_i, b_i), but here r_i is a collection of feasible sets of items, and b_i is a function relating the set offered to the bid.

2.1 Measures of performance

Competitiveness. Suppose we accept some set of requests A. Our total income is equal to $I = \sum_{i \in A} p_i$. The optimum offline omniscient income is the maximum, over sets A^*, of $I^* = \sum_{i \in A^*} v_i$. We define our competitive ratio to be the maximum over request sequences R of the ratio $\frac{I^*}{E[I]}$. Our algorithms will be randomized and the expected value of our result is over the set of random choices made.

We assume that we are given an online algorithm for the optimization version of the problem. This algorithm guarantees some competitive ratio ρ, which means the algorithm accepts some set of requests A such that

$$\rho E[\sum_{i \in A} b_i] \geq \sum_{i \in A^*} b_i.$$

This algorithm is not truthtelling, in that users may well benefit from setting $b_i < v_i$.

Efficiency or global well-being. We comment that our upper bounds on profit competitiveness imply, as an obvious corollary, similar bound on *global efficiency*, namely overall well-being of all the users. This contrasts with most other work on truthtelling mechanisms for general online resource allocation, where only efficiency is maximized, and competitiveness is, in most cases, arbitrarily poor.

Computational hardness. Another issue is ability to compute solutions in polynomial time. General combinatorial auctions require exponential representation of input and exponential computation effort. There are two approaches. One “non-constructive” approach is to assume existence of certain external optimization procedure, or "black box", performing necessary computational work; this approach is commonly used in the literature, since many problems in this domain are impossible even to approximate. A "constructive" approach requires specifying a polynomial time algorithm for the above "black box" computation, and we will give such constructive results for problems in routing and admission control.

3 Statement of results

3.1 Basic results

We give a truth telling algorithm using *arbitrary* nice online optimization algorithm as a black box, which will be $O(\rho + \log \mu)$ competitive against the offline omniscient algorithm, where $\mu = \max_i v_i$ (we assume the minimum nonzero valuation is one). We observe that there is an $\Omega(\log \mu)$ lower bound [13] in most cases. In addition, it is straightforward to obtain $O(\rho \log \mu)$ competitivity as follows. We choose a random number m between 1 and μ via an exponential distribution, and discard all requests with $b_i < m$. The remaining requests are sent to the online algorithm with bid equal to m.

In addition to improving this ratio to $O(\rho + \log \mu)$, our algorithm is resilient against various attacks including retries and coalitions. We show in section 4.3 that even if users are allowed to "cheat" in these ways, our expected income remains competitive with the offline omniscient algorithm.

In section 4.4 we explain how to modify the algorithm if we do not have knowledge of the value μ in advance.

We first present our algorithm for the single-choice auction and then generalize to arbitrary combinatorial auctions.

3.2 Applications of basic results

Network admission control for unicast. We consider the following problem. We are given a capacitated network. Communication requests arrive one at a time online. Each request specifies a path from the source to the destination node, and a bid for the service of satisfying the connection. We will model this network in terms of permanent reservations of virtual circuits; the algorithm can be easily extended to controlling reservations with known durations. Our goal is to devise a strategy for admission control (as well as pricing the admitted requests) in order to maximize our income without exceeding the network capacities. We will assume that the network capacities are reasonably large (at least $O(\log n\mu)$ where n is the number of nodes in the graph and μ is the maximum request valuation).

The optimization version of this problem was addressed in [3]. Assuming that each request is accompanied by a true valuation (rather than a possibly untruthful bid), an $O(\log n\mu)$-competitive online algorithm was given, and matching lower bounds were provided. The large capacity restriction on the edges is necessary for this guarantee; the case of unit-capacity edges is provably hard. We would like to extend this result to an auction setting.

The algorithm of [3] works as follows. Each edge in the network is assigned an "opportunity" cost which is exponential in the current load on the edge. For each request, we compute a cost which is the sum of the cost of the edges along the shortest satisfying path. If this cost is less than the value of the request, then we accept and update the loads. Otherwise we reject the request (loads remain unchanged). We observe that the actions of this deterministic algorithm are entirely determined by the set of accepted requests (the actual values of requests do not effect the action of the algorithm other than by determining whether to accept or reject). This algorithm is *nice*, and therefore ideal for the technique described in Fig. 1.

It immediately follows that we can construct a truthtelling mechanism with competitivity $O(\rho + \log \mu) = O(\log n\mu)$, which is asymptotically identical to the competitivity of the best possible online algorithm for the optimization version of the problem. We have effectively lost nothing by transforming to the auction problem.

In addition, we observe that the costs generated by the algorithm of [3] are nondecreasing (as the loads only rise with time) and additive. It follows that our mechanism is resilient against retries, coalitions, and resellers.

Combined unicast routing and admission control. In additional to controlling the rates, we can also select the routes between senders and receivers. Notice that there are exponentially many options, and only one needs to be selected. This is an example of an combinatorial auction that can be handled in polynomial time.

Again, the optimization version of this problem was addressed in [3], with an $O(\log n\mu)$-competitive online algorithm was given, and matching lower bounds were provided. The algorithm in in [3] selects shortest path in the above opportunity cost, and applies admission control procedure above.

Once again, a truthtelling mechanism with optimal competitivity $O(\rho + \log \mu) = O(\log n\mu)$ follows, which is resilient against retries, coalitions, and resellers.

Multicast admission control is a generalization of unicast admission control.

Instead of paths from senders to receivers, we are given a forest of trees, rooted at a number of senders. We are given a set of bids b_i for each of the nodes, and we can choose to connect some or all of the requested nodes into a tree. Our algorithm's income will be the total prices assigned to the nodes which are actually part of the tree we construct. We are limited by network bandwidths in the underlying graph; each edge has a capacity and we may place at most a bounded number of trees along the edge.

Multicast admission control with routing is a generalization of the above, when we also need to construct a "steiner tree". In this case, the computational hardness results arise from optimizing the function of benefit minus "opportunity" cost. Since we cannot solve steiner tree exactly in polynomial time, we cannot obtain any reasonable approximation to the difference problem [10]. However, in a capacitated network there is no direct notion of cost. We will compute an online "opportunity cost" which will be only approximate because of steiner tree hardness, but the errors in this computation can effectively be amortized against the benefit gained from earlier multicast routings if the capacities are reasonably large.

For multicast routing, the hardness results arise from optimizing the function of benefit minus cost. Since we cannot solve steiner tree exactly in polynomial time, we cannot obtain any reasonable approximation to the difference problem [10]. Here we exploit the fact that in a capacitated network there is no direct notion of externally-imposed cost, except internally-maintained "opportunity cost". The latter can be only approximately optimized because of steiner tree hardness, but the errors in this computation can effectively be amortized against the benefit gained from earlier multicast routings if the capacities are reasonably large.

This situation differs from the scenario presented by Feigenbaum et al [10], in that they are trying to construct a single-commodity multicast tree with costs on the edges in an offline scenario, whereas we are solving a multi-commodity problem, namely packing large numbers of multicast trees into a graph with capacity constraints (but no costs) on the edges. In contrast to the hardness results presented for the single tree with costs ([10] shows that the problem is effectively inapproximable), we are able to provide polylogarithmic positive results for the capacitated problem.

4 Single-choice auction

The single-choice auction algorithm $\mathcal{B}'$ uses any deterministic or "nice" (see Def. 1) randomized online algorithm $\mathcal{B}$ as a black box .

Compute m between 1 and μ. With probability $\frac{1}{2}$ we have $m = 1$, otherwise $m = 2^i$ with probability $\frac{1}{2\log\mu}$ for each $i \leq \log \mu$.

Notice that the online algorithm $\mathcal{B}$ will update its state (assuming we accepted request i) even if we in fact computed $b_i < mc_i$ and rejected request i.

Theorem 4.1 *Let $\mathcal{B}$ be a any deterministic or "nice" randomized algorithm (Def. 1) which is ρ-competitive, and let μ be the range of bids. Then, algorithm $\mathcal{B}'$ in Fig. 1 is truthtelling mechanism which is $O(\rho + \log \mu)$ competitive against the offline omniscient algorithm.*

Basic Algorithm $c_i \leftarrow \min v$ such that $\mathcal{B}$ accepts (r_i, c_i) . Feed (r_i, b_i) to $\mathcal{B}$ & update $\mathcal{B}$'s state If $b_i \geq mc_i$ then accept and charge $p_i = mc_i$. /* Otherwise we reject request i * /

Figure 1: The algorithm for a single-choice auction.

4.1 Preliminaries

Definition 1 *We define an online algorithm as* nice *if the following conditions hold.*

- *If a request is a accepted with a bid b, it must also be accepted with larger bid $b' > b$.*
- *the decision of whether to accept request i (whether $i \in A$) may depend upon the current members of set A, but does not depend upon the bids b_j of the members $j \in A$. It may however, depend upon the bids b_j that were rejected, i.e. $j \notin A$.*

Theorem 4.2 *If there exists a deterministic online ρ-competitive algorithm for the optimization problem, then there exists a* nice *algorithm with the same competitive ratio.*

Proof: Suppose we are given a deterministic online algorithm $\mathcal{B}$. We construct algorithm $\mathcal{B}'$ as follows. Some stream of requests R arrives. As each request i arrives, we compute c_i to be the minimum value such that, if algorithm $\mathcal{B}$ were presented with requests $(r_1, \min(b_1, c_1)), (r_2, \min(b_2, c_2)), ..., (r_i, c_i)$, then it would accept the request (r_i, c_i). Algorithm $\mathcal{B}'$ accepts request i if its bid satisfies $b_i \geq c_i$.

It is immediate that algorithm $\mathcal{B}'$ satisfies the niceness conditions. Increasing the bid of an accepted request will not cause it to be rejected (since the new bid is still more than c_i). The decision of whether to accept request i depends upon the previous requests r_j and the values of the minimum of b_j and c_j for them. Since only the rejected requests have $b_j < c_j$, it follows that only the bids of rejected requests can effect our decision.

It remains to analyze the competitive ratio. Suppose there was some request stream R which could be fed to algorithm $\mathcal{B}'$ for which the competitive ratio is worse than ρ. We construct a new request stream R' by replacing the bid of request i with $\min(b_i, c_i)$. Now suppose $\mathcal{B}'$ rejects some request i from stream R. It follows that $\mathcal{B}$ would reject request i from stream R'. Similarly, if $\mathcal{B}'$ accepts request i from stream R, then $\mathcal{B}$ would accept the request. We know that if A is the set of requests accepted by $\mathcal{B}'$ from stream R, then $\rho \sum_{i \in A} b_i < \sum_{i \in A^*} b_i$. However, consider $\mathcal{B}$ acting on stream R'. We have profit $\sum_{i \in A} \min b_i, c_i$, whereas there exists an alternate solution which gains profit $\sum_{i \in A^*} \min b_i, c_i$. Since algorithm $\mathcal{B}$ is ρ-competitive, we know that $\rho \sum_{i \in A} \min b_i, c_i \geq \sum_{i \in A^*} \min b_i, c_i$. We now consider replacing $\min b_i, c_i$ with the value b_i. The lefthand side of the inequality increases, as $b_i \geq c_i$ for each request $i \in A$. The righthand side of the inequality increases by a smaller amount, since $b_i \leq c_i$ for each request in A^* except those also accepted by A. So we have $\rho \sum_{i \in A} b_i \geq \sum_{i \in A^*} b_i$ as desired. ■

4.2 Analysis for a single choice case

We will consider several categories of requests. There are some requests A which our algorithm accepted. There is some set Q of requests which have $c_i \leq b_i$; these are the requests which $\mathcal{B}$ accepts, but we will accept only some of them because of our random choice of m. Finally, there is a set P of requests which the optimum omniscient offline accepts, but which have $b_i < c_i$ and which we reject. We need to show that our total income $\sum_{i \in A} p_i$ is comparable to the optimum income, which is bounded above by $\sum_{i \in P} v_i + \sum_{i \in Q} v_i$.

Lemma 4.1

$$\sum_{i \in P} v_i \leq \rho \sum_{i \in Q} c_i$$

Proof: Consider the given online algorithm for the optimization problem. If we feed this algorithm the pairs (r_i, b_i), it will accept exactly $i \in Q$. We now consider feeding the algorithm the pairs $(r_i, \min(b_i, c_i))$. Since c_i is defined as the minimum bid such that (r_i, c_i) will be accepted, and the future actions of the algorithm do not depend upon the actual bids submitted by accepted requests, the algorithms behavior on this new request stream will be identical to its behavior on the original request stream. Since the algorithm is truth telling, we have $b_i = v_i$, and one feasible solution to the problem involves accepting the set P and obtaining income of $\sum_{i \in P} v_i$. We were given a ρ-competitive algorithm which obtains income of $\sum_{i \in AQ} \min(b_i, c_i) = \sum_{i \in Q} c_i$, and the presumed competitivity yields the desired result. ■

Lemma 4.2

$$E[\sum_{i \in A} p_i] \geq \frac{1}{2} \sum_{i \in Q} c_i$$

Proof: Consider any request $i \in Q$. Notice that this set does not depend upon our choice of m. With probability $\frac{1}{2}$ we will have $m = 1$ and thus $p_i = c_i$ and $A = Q$. So the expected value of $\sum_{i \in A} p_i$ will be at least half the value of $\sum_{i \in Q} c_i$. ■

Lemma 4.3

$$\sum_{i \in Q} v_i \leq 4(\log \mu) E[\sum_{i \in A} p_i]$$

Proof: We note that Q is the set of requests accepted by $\mathcal{B}$, and does not depend upon our choice of the random value m. For any $i \in Q$, we will compute its expected contribution to the righthand sum. With probability $\frac{1}{2 \log \mu}$ we selected m such that $mc_i \leq v_i \leq 2mc_i$, in which case we will have $i \in A$ and $p_i \geq \frac{1}{2} v_i$. The expected contribution of request i to $\sum_{i \in A} p_i$ will be at least $\frac{v_i}{4 \log \mu}$, and summing this over requests i gives the desired bound. ■

Theorem 4.3

$$\sum_{i \in A^*} v_i \leq (2\rho + 4 \log \mu) E[\sum_{i \in A} p_i]$$

Proof: We know the set A^* is a subset of $Q \bigcup P$. Thus we have $\sum_{i \in A^*} v_i \leq \sum_{i \in Q} v_i + \sum_{i \in P} v_i$. We now apply the lemmas to obtain $\sum_{i \in A^*} v_i \leq (2\rho + 4 \log \mu) E[\sum_{i \in A} p_i]$. ■

We also need to show that this algorithm is truthtelling. This is immediate from previous work.

Theorem 4.4 *The algorithm is a truthtelling mechanism.*

Proof: We observe that the price p_i is independant of the bid b_i; the price is set based upon previous requests and a random number m. If the bid does not match the true valuation, the price remains unchanged, so the bidder does not benefit. ■

4.3 Retry, Coalitions, and Resellers

Truthtelling mechanisms make certain assumptions about the requesters. It is assumed that each requester is an independent agent who offers one and only one bid to the auction. This model is not realistic. First, we have the possibility of retry. An individual whose request is rejected might continue resubmitting requests until one is accepted. If the computed price for this request fluxuates, it might be beneficial to the requester to lie about his bid, then retry on a reject (possibly with a different bid). Second, we have the possibility of coalitions. A group of requesters could agree to lie about their valuations or to reorder their requests. Third, we have the possibility of resellers, where a group of requesters merge their requests into a single request (for the sum of the required resources); they will benefit if the price computed by the algorithm for the union is less than the sum of the individual prices.

We will show that our approach remains competitive despite retries, coalitions, and resellers, provided the given online optimization algorithm has certain additional properties.

Theorem 4.5 *If the costs computed by $\mathcal{B}$ for a particular request are nondecreasing, then our approach is resilient against retries.*

Proof: If we repeat request r_i at some later time $j > i$, then the computed cost will be $c_j > c_i$. The computed price will therefore be $mc_j > mc_i$, which is only larger. ■

Theorem 4.6 *If the costs computed by $\mathcal{B}$ for a particular request are nondecreasing, then our approach is resilient against coalitions.*

Proof: Suppose some set of requests R form a coalition. Some of these requests $i \in R$ might lie about their valuations, bidding $b_i \neq v_i$. Bid i has three possibilities, it may be pre-rejected (rejected without effecting future actions of the algorithm), pre-accepted but rejected later (causing $\mathcal{B}$ to update its costs), or actually accepted. The actual bid b_i determines which set the request will join, but has no other effect. Suppose request i lies and causes itself to be rejected (i.e. $b_i < c_i \leq v_i$). In this case, since the request i will not effect the actions of the algorithm it may as well be removed from the request stream. The coalition would only benefit by re-adding request i at a time subsequent to all other coalition members. On the other hand, suppose request i lies and causes itself to be pre-accepted but rejected, instead of being accepted. This means we have $c_i \leq b_i < mc_i \leq v_i$. We observe that the actions of the algorithm on other subsequent requests are uneffected by this (since the algorithm only depends upon the pre-accepts and the choice of m). If we had set $b_i = v_i$ then all future actions of the algorithm are unchanged, but we will obtain additional benefit of $v_i - mc_i > 0$, so the coalition has no incentive to lie in this way. The only other possible "lies" involve setting $b_i > v_i$. It follows that the coalition has a dominant strategy which might reorder the requests, but which guarantees $b_i \geq v_i$ for all requests i. But our competitivity guarantee of ρ for the online black box $\mathcal{B}$ was against adversarial orderings of the requests, so we retain our competitivity in the face of coalitions. ■

Theorem 4.7 *If $\mathcal{B}$ assigns cost to a union of requests which is at least equal to the cost of those requests appearing in sequence, then our approach is resilient against resellers.*

Proof: Suppose requests from i to j apply to a reseller. The algorithm will see a single request $(\sum_{x=i}^{j} r_i, \sum_{x=i}^{j} b_i)$. The cost computed for this union of requests will be $C \geq \sum_{x=i}^{j} c_i$, leading to a price of mC. But if the requests had not used the reseller, they would see a sum of prices which is only smaller, implying that the benefit has been reduced. ■

4.4 Guessing the Max Bid

The technique described in Fig. 1 depends upon knowledge of the maximum possible bid. This knowledge is needed in order to determine the distribution of random variable m. If the algorithm uses a value for this maximum bid which is too low, then it will not be competitive. If the value is too high, then the competitive ratio deteriorates (since it depends logarithmically upon the value μ). In some applications a reasonable accurate guess of μ will be available, but in general this may not be the case. We will describe an alternate method for generating m which does not depend on prior knowledge of the value μ. This method is based on the idea of classify and select on unbounded range, first presented by Lipton and Tomkins [16].

Theorem 4.8 *Let $\mathcal{B}$ be a "nice" randomized algorithm which is ρ-competitive. We can design a truthtelling mechanism which is $O(\rho + \log\mu(\log\log\mu)^2)$ competitive against the offline omniscient algorithm, where μ is the range of bids, without prior knowledge of the value of μ.*

Proof: The algorithm is unchanged, except that we must generate m in a different way. We will set $m = 1$ with probability $\frac{1}{2}$. Otherwise, we set $m = 2^i$ with probability density function $\frac{1}{2(e+i)(\ln(e+i))^2}$ for nonnegative i. Integrating the probability function, we have total probability equal to one. We now consider the probability of selecting m to be "just right" to charge some request in Q its bid. Before, this probability was $\frac{1}{2\log\mu}$. Now, the probability looks like $O(\frac{1}{\log\mu(\log\log\mu)^2})$ and this yields the result claimed. ■

5 General Combinatorial Auction

The multiplicative random factor m in our algorithm (Fig. 1) might tempt the customers to lie in case of auctions with multiple choices. For example, suppose the customer would like either item 1 or item 2, and their values to him are 100 and 10 respectively. Suppose we have costs $c_1 = 90$ for item one and $c_2 = 1$ for item two. Clearly the customer prefers to purchase item one if the prices are equal to the costs. However, once we add the random multiplier m, it will frequently be the case that $100 - 90m < 10 - m$ and the customer prefers item two.

We can avoid this problem by making the factor of m additive instead of multiplicative. We assume we are given a black box online algorithm $\mathcal{B}$ which will be competitive assuming all bids are truthful and each customer bids on only a single set of items. Various results show that producing such a black box is difficult. If we assume that only a single copy of each item is available, then we are required to solve an approximation of the unit-capacity online flow problem, which has hardness results from (among others) [3]. However, if we assume large supply of each item we can immediately obtain such a black box using an exponential-cost routing algorithm.

In addition, we assume that given a request pair (r_i, b_i), we are able to select a set $s \in r_i$ which maximizes $b_i(s) - c_i(s)$. If the request r_i is given as a polynomial length "list" of feasible sets, then this computation is straightforward. There are interesting cases where the list of feasible sets is exponential in length and we are unable to compute this maximization exactly. The next section will deal with extensions of this type.

Combinatorial Auction Algorithm
With probability $\frac{1}{2}$ set $m = 0$, otherwise $m = 2^i$ with probability $\frac{1}{2\log\mu}$ for each $i \leq \log \mu$.
For each set s, compute $c_i(s)$ as the minimum value such that $\mathcal{B}$ accepts $(r_i, c_i(s))$.
Select S which maximizes $b_i(s) - c_i(s)$. Feed $(s, b_i(s))$ to $\mathcal{B}$, updating its state.
If $b_i(s) \geq m + c_i(s)$ then accept
  give s to the customer and charge $p_i = m + c_i(s)$ /*Otherwise we reject request i.*/

Figure 2: The algorithm for mutiple-choice (combinatorial) auction.

We will first prove this algorithm is a truthtelling mechanism.

Theorem 5.1 *The algorithm is truthtelling.*

Proof: Suppose a customer requests one of several sets s. For each set, we implicitly assign a price $p_i(s) = m + c_i(s)$. The customer's goal is to obtain the set which will maximize the difference of utility and price, $v_i(s) - p_i(s)$. This is exactly the same set which maximizes $v_i(s) - c_i(s)$. If the customer is truthful, he will receive the most desirable set, and since the price paid is independent of the customer's bids, the algorithm is truthtelling. ■

We again analyze the algorithm by considering several sets of requests. We have set A of requests which we accept, set Q of requests which algorithm $\mathcal{B}$ accepts (which have $b_i(s) \geq c_i(s)$), and set P of requests which are accepted by the offline omniscient but which we rejected altogether. In addition, we will now divide Q into Q_1, requests which have $b_i(s) \geq \frac{1}{2} b_i(s^*)$, and Q_2, requests which have $b_i(s) < \frac{1}{2} b_i(s^*)$. Here s^* represents the set which the offline optimum ominiscient uses to accept request i (if the optimum rejects request i, then s^* is the set of maximum $b_i(s^*)$).

Theorem 5.2

$$\sum_{i \in A^*} v_i(s^*) \leq (6\rho + 8 \log \mu) E[\sum_{i \in A} p_i]$$

Proof: We split A^* into several sets: $A^* \in Q_1 \bigcup Q_2 \bigcup P$. As before, we have $\sum_{i \in P} v_i(s^*) \leq 2\rho \sum_{i \in A} p_i$ from lemma 4.1. We now consider the members of Q_1. For any request $i \in Q_1$, there is a probability of $\frac{1}{2\log\mu}$ that we set m such that we obtain a price with $b_i(s) \geq p_i \geq \frac{1}{2} b_i(s) \geq \frac{1}{4} b_i(s^*)$, so we have $\sum_{i \in Q_1} v_i(s^*) \leq 8 \log \mu E[\sum_{i \in A} p_i]$.

This leaves the set Q_2. For each of these requests, we could have used used the optimum set to satisfy them but chose not to. It follows that $\frac{1}{2}v_i(s^*) \geq v_i(s) - c_i(s) \geq v_i(s^*) - c_i(s^*)$, from which we conclude that $v_i(s^*) \leq 2c_i(s^*)$. We can now argue that if $\mathcal{B}$ were given the requests in Q_2 along with their optimum subsets and half the optimum valuations, it would reject. We conclude that $\sum_{i \in Q_2} v_i(s^*) \leq 4\rho E[\sum_{i \in A} p_i]$ using arguments similar to those in lemma 4.1 for pre-rejected requests. Combining these equations yields the lemma claimed. ■

We observe that the above algorithm is not resilient against resellers, since the cost assigned to a union of requests adds the random m only once instead of multiple times, allowing the reseller to obtain a benefit.

6 Polynomial-time Multicast Routing/Admission Mechanisms

The problem can be viewed as an extension of the case of combinatorial auctions presented in the previous section. We are given a list of feasible subsets (multicast trees) along with valuations for each of them. However, the list of feasible subsets is now exponentially long, and we cannot determine the set s which optimizes $b_i(s) - c_i(s)$ (optimizing this function is precisely what [10] shows to be hard). In fact, we cannot even approximate the difference! Instead, we will assume we are given a black box algorithm $\mathcal{A}$ which guarantees to find s with $\beta(b_i(s) - c_i(s)) \geq b_i(s^*) - c_i(s^*)$ (for some $\beta \geq 1$) under the assumption that the optimizing set s^* has $b_i(s^*) \geq \alpha c_i(s^*)$ (for some $\alpha \geq 2$). For the case of multicast trees, any of the various approximation algorithms for the maximal dense subtree ($k - MST$) will serve this purpose; for example we could use the algorithms of [2]. As usual, we additionally have the algorithm $\mathcal{B}$ which performs online optimization (without truthtelling) under the assumption that we are given one tree-bid pair at a time and must choose to accept or reject (an exponential cost algorithm on the edges such as [3] will serve for multicast).

We have the following algorithm:

Compute m between 0 and μ. With probability $\frac{1}{2}$ we have $m = 0$, otherwise $m = 2^i$ with probability $\frac{1}{2\log\mu}$ for each $i \leq \log\mu$.

Multicast Algorithm
Use $\mathcal{A}$ to compute s which optimizes $b_i(s) - c_i(s)$
Feed $(s, b_i(s))$ to $\mathcal{B}$ & update $\mathcal{B}$'s state
If $b_i(s) \geq m + c_i(s)$ then accept and charge $p_i = m + c_i(s)$. /* Otherwise we reject request i * /

Figure 3: The algorithm for multicast.

The analysis again proceeds by dividing the requests into four sets. We have set A of requests which we accept, set Q_1 of requests for which $b_i(s^*) \geq \alpha c_i(s^*)$ for the optimum set s^*, set Q_2 for which $b_i(s^*) < \alpha c_i(s^*)$ but the online algorithm $\mathcal{B}$ still finds a way to accept, and set P which the offline omniscient algorithm accepts but $\mathcal{B}$ rejects.

Theorem 6.1 $\sum_{i \in A^*} v_i(s^*) \leq (4\rho\alpha + 8(\log\mu)\beta E[\sum_{i \in A} p_i]$

Proof: We split A^* into several sets: $A^* \in Q_1 \bigcup Q_2 \bigcup P$. We consider a request $i \in P$. If there was a set s with $b_i(s) \geq \alpha c_i(s)$, then we would have found a set with positive benefit minus cost and $\mathcal{B}$ would have accepted. It follows that $v_i(s) < \alpha c_i(s)$. We conclude that the algorithm $\mathcal{B}$ would reject the request $(s, v_i(s)/\alpha)$ for any set $s \in r_i$, and competitivity of $\mathcal{B}$ yields the inequality $\sum_{i \in P} v_i(s^*) \leq 2\rho\alpha E[\sum_{i \in A} p_i]$ as in lemma 4.1. We now consider the members of Q_1. For these requests we have a probability $\frac{1}{2\log\mu}$ of setting m just right such that we obtain a price of $p_i \geq \frac{1}{2}b_i(s)$. Membership in Q_1 implies that $b_i(s^*) \geq \alpha c_i(s^*)$ which implies $\beta(b_i(s) - c_i(s)) \geq b_i(s^*) - c_i(s^*)$. From these inequalities along with $\alpha \geq 2$, we conclude that $\beta b_i(s) \geq \frac{1}{2}b_i(s^*)$, allowing us to show that

$$\sum_{i \in Q_1} v_i \leq 8(\log\mu)\beta E[\sum_{i \in A} p_i].$$

Finally, we consider set Q_2. Membership in Q_2 implies that $v_i(s^*) < \alpha c_i(s^*)$. It follows that the algorithm $\mathcal{B}$ would reject $(s^*, v_i(s^*)/\alpha)$, so again obtain the inequality $\sum_{i \in Q_2} v_i \leq 2\rho\alpha E[\sum_{i \in A} p_i]$. Combining these inequalities yields the result claimed. ■

This yields a competitive ratio of $O(\rho\alpha + \beta \log \mu)$. However, we observe that the algorithm described is not truthtelling. If the customer lies about his valuations b_i, it might induce our approximation algorithm $\mathcal{A}$ to choose a different set s to satisfy the customer. Conceivably this set s could have a larger value of $v_i(s) - c_i(s)$, leading the customer to prefer it. However, by modifying the bids to cause this to occur, the customer is only helping the algorithm. We state this formally as follows.

Theorem 6.2 *If the customers lie in order to improve their benefit, the algorithm is still competitive.*

Proof: Suppose customer i lies. This leads to customer i being assigned some set s' instead of the set s which would have been assigned had he been truthful. So the customer's value minus price is now $v_i(s') - c_i(s') - m$ instead of $v_i(s) - c_i(s) - m$. We assume that customer i is acting in his own self-interest, so we conclude that $v_i(s') - c_i(s') \geq v_i(s) - c_i(s)$. But this means the customer is effectively producing a "better" approximation algorithm $\mathcal{A}'$ for us, and the proof of the preceding theorem will again hold. ■

7 Appendix: Published Prices and Open Problems

We consider the problem of publishing prices. Here, we would like to be able to post a price which does not change based upon rejected requests. In our modular approach described earlier, the costs will change on the basis of pre-rejects. We can modify the algorithm to the following:

> **Published Price Algorithm**
> $c_i \leftarrow \min v$ such that $\mathcal{B}$ accepts (r_i, c_i) .
> Post prices mc_i for r_i; a new m is computed for each customer.
> If $b_i \geq mc_i$, then feed (r_i, b_i) to $\mathcal{B}$ & update $\mathcal{B}$'s state

Figure 4: The algorithm for price-posting.

In this algorithm, the prices are independent of the rejects. However, the prices are continually being assigned new random values for each request. This is undesirable both because we do not have true "published prices" and because the algorithm is no longer resilient against retries.

Since our offered price is always at least c_i, we guarantee we will not violate capacity constraints and so forth if the generic online could not do so.

The proof of competitivity for this new algorithm requires only slight modifications. We can still classify requests as "pre-rejected" if they have $b_i < c_i$ (in other words, they will be rejected automatically regardless of the random number). Each request belongs to some category, P, Q, or A. Which category it belongs to depends upon random variables, since the value of the cost at the current time is no longer deterministic (it depends on which earlier requests were accepted).

However, we can still show that have $\sum_{i \in P} v_i \leq \rho \sum_{i \in Q} c_i$ as a deterministic guarantee. We now consider some request i. There is some probability it was pre-rejected, but let's assume that is not the case. Once we take $b_i \geq c_i$ as given, whether this request is actually accepted is independent of previous events.

Also, we can still guarantee that $4(\log \mu) E[p_i | i \in Q] \geq v_i$. This enables us to bound the expected cost much as before.

The above arguments are formalized below.

Lemma 7.1

$$\sum_{i \in P} v_i < \rho \sum_{i \in A} c_i$$

Proof: This will be a deterministic guarantee, and we will show it is true for any request stream and set of random choices. Suppose for some request stream and random choices the theorem fails. We will construct a modified request stream as follows. First, for each request i with $c_i < v_i < c_i r_i$, we remove request i and random number r_i from the streams of requests and random numbers. We now have a new request stream, and the behavior of the algorithm on this stream will be identical (since the eliminated requests were to be rejected anyway). We now consider what would happen if we feed this request stream to the generic online algorithm. We have eliminated the "random rejects" so the generic online will accept exactly the same requests on this new request stream as our randomized algorithm. In fact, even if we set $v_i = min(v_i, c_i)$ for each request, the generic online will still accept exactly the same set of requests A which our online algorithm accepts. This means the generic online obtains benefit equal to $\sum_{i\in A} c_i$, whereas it could have obtained at least $\sum_{i\in P} v_i$ (observing that these requests were not modified in any way and are still a part of the request stream with their original valuations). Since the generic online has a competitive ratio of ρ, the theorem must hold. ∎

Lemma 7.2

$$E[\sum_{i\in Q} v_i] < 2(\log\mu)E[\sum_{i\in A} c_i r_i]$$

Proof: Define potential function

$$\phi = 2(\log\mu)\sum_{i\in A} c_i r_i - \sum_{i\in Q} v_i.$$

Consider what happens to ϕ when request i arrives. There is some probability (based upon earlier requests and random choices) that $v_i < c_i$. If this happens, then request i will be rejected, but will not be a member of the set Q, so the value of ϕ will be unchanged. Otherwise, with some probability we have $v_i > c_i$. In this case, we will have $c_i r_i < v_i < 2c_i r_i$ with probability $1/(log\mu)$. If this happens we will have $i \in A$, and the potential function will increase by $v_i(log\mu)$. Regardless, we will have $i \in Q$ so the potential function will decrease by v_i. In an expected sense, the potential function does not decrease.

This gives the lemma claimed. ∎

Combining the two theorems gives:

$$E[\sum_{i\in Q} v_i + \sum_{i\in P} v_i] < 2[(\log\mu) + \rho]E[\sum_{i\in A} c_i r_i]$$

The main open problem is combining the various extensions to the algorithm. Ideally, we would like to be able to publish prices which the customers can take or refuse, with the assumption that prices do not change except after a sale is made. Our extension regarding published prices does not quite handle this because a distinct random number must be applied to each customer (they do not simply accept or refuse the published price).

References

[1] A. Archer and E. Tardos. Frugal path mechanisms. *Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms*, 2002.

[2] A. Awerbuch, Y. Azar, A. Blum, and S. Vempala. New approximation guarantees for minimum weight k-trees and prize-collecting salesman. *SIAM Journal of Computing*, 1999.

[3] A. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. *Proceedings of the 25th ACM Symposium on Theory of Computing*, 1993.

[4] A. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts. Competitive routing of virtual circuits with unknown duration. *Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[5] B. Awerbuch. Online algorithms for selective multicast: A survey. *Proceedings of the Dagstuhl Workshop on Online Algorithms*, 1996.

[6] B. Awerbuch and T. Singh. Online algorithms for selective multicast and maximal dense trees. *Proceedings of the ACM Symposium on Theory of Computing*, 1996.

[7] A. Bagchi, A. Chaudhary, R. Garg, M. Goodrich, and V. Kumar. Seller-focused algorithms for online auctioning. In *7th International Workshop on Algorithms and Data Structures (WADS 2001)*, pages 135–147, 2001.

[8] Z. Bar-Yossef, K. Hildrum, and F. Wu. Incentive-compatible online auctions for digital goods. In *13th ACM-SIAM Symp. on Discrete Algorithms*, pages 964–970, 2002.

[9] E. Clarke. Multipart pricing of public goods. *Public Choice*, 1971.

[10] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transactions. *Proceedings of the ACM Symposium on Theory of Computing*, 2000.

[11] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharting the cost of multicast transmissions. *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2001.

[12] A. Goel, M. Henzinger, and S. Plotkin. Online throughput-competitive algorithm for multicast routing and admission control. *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[13] A. Goldberg, J. Hartline, and A. Wright. Competitive auctions and digital goods. *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms*, 2001.

[14] T. Groves. Incentives in teams. *Econemetrica*, 1973.

[15] R. Lavi and N. Nisan. Competitive analysis of incentive compatible on-line auctions. In *ACM Conference on Electronic Commerce*, pages 233–241, 2000.

[16] R. J. Lipton and A. Tomkins. Online interval scheduling. *Proceedings of the 5th ACM-SIAM Symposium on Discrete Algorithms*, 2001.

[17] N. Nisan and A. Ronen. Algorithmic mechanism design. In *31st ACM Symp. on Theory of Computing*, pages 129–140, 1999.

[18] W. Vickery. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 1961.