Profit Maximizing Mechanisms for the Extended Multicasting Game[1]

Shuchi Chawla[2] David Kitchin[3] Uday Rajan[4]
R. Ravi[4] Amitabh Sinha[3]

July, 2002

CMU-CS-02-164

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We consider the design of multicast networks when both edges and nodes are selfish agents. Our objective is a budget-balanced, strategy-proof mechanism which selects the set of clients to receive service and constructs a network to provide the service. It extracts payments from the clients and pays edges to participate in the network, and aims to maximize profit from the transaction. We introduce the notion of *profit guaranteeing* mechanisms, and show the existence of one such mechanism. The mechanism provides guarantees of the form that in a sufficiently profitable market, it obtains some fraction of the obtainable profit, and if the market is sufficiently unprofitable, then the mechanism demonstrates that no profitable solution exists. The mechanism runs in polynomial time. To our knowledge, this is the first study of mechanisms for designing multicast networks in which edge values are unknown.

[1]This research was supported by the ALADDIN Center for Applied Algorithms and funded by the National Science Foundation ITR program under Grant No. CCR-0122581.
[2]Department of Computer Science, Carnegie Mellon University. `shuchi@cs.cmu.edu`
[3]Research supported by the ALADDIN Center and its REU Program and funded by the National Science Foundation under Grant No. CCR-0122581. {`dkitchin,asinha`}`@andrew.cmu.edu`
[4]Graduate School of Industrial Administration, Carnegie Mellon University. {`urajan,ravi`}`@andrew.cmu.edu`

Keywords: Game Theory, Mechanism Design, Approximation Algorithms, Multicasting Game

1 Introduction

The design of multicast networks in a game-theoretic setting has received a lot of attention among researchers recently. Given a network with a distinguished node (root) and clients at other nodes, the problem is to select a set of users for service and construct a multicast tree connecting these users to the root. Clients are willing to pay some maximum amount of money for receiving the service and agents owning edges charge a minimum fee for participating (known as their *values*). The task is to charge fees from clients and pay agents owning the edges so as to maximize profit from the transaction. The problem was introduced by Herzog, et al [9].

From an optimization perspective, given a graph with values associated with nodes (clients) and edges, this problem reduces to finding a subset of the graph that maximizes the sum of values of selected nodes minus the sum of values of selected edges (this is the *efficiency*/surplus of the solution). This problem is known to be NP-hard [5].

Several variants of the problem have been studied in the approximation algorithms literature. Most notable is the *Prize Collecting Steiner Tree* (PCST) problem, which is to find a subset that minimizes the cost of selected edges plus the values of the nodes that are *not* selected. For the optimal solution, this is the same as maximizing efficiency. However, in terms of approximation, the two problems are different. In particular, the problem of maximizing efficiency is inapproximable in polynomial time (as we show in a later section), whereas Goemans, et al [8] achieve a 2-approximation for PCST.

From a game theory perspective, the problem becomes harder as we now have to pay (selfish) edges and charge (selfish) nodes in such a way that they reveal their values truthfully. So far game theorists have studied a simpler subproblem in which edge costs are known [3, 5, 10, 13]. The objectives for this subproblem have been two-fold: to recover the cost of constructing the network from the nodes (Budget Balance), and to output the most efficient solution – where the difference between the total value of selected nodes and total cost of selected edges is maximized. It is well known that the two objectives cannot be met simultaneously [13, 4].

For this subproblem, two kinds of mechanisms have been proposed. The first kind output the most efficient solution (albeit in exponential time or for simple networks such as trees[5, 13]), but are unable to recover the cost of edges from nodes. The second kind [10] are budget balanced and occasionally make a profit. However, they give no guarantee on the efficiency loss of the solution.

In this paper, we study the problem from a different perspective. We aim to *maximize the profit* from the solution, and we do so when edge costs are *unknown* (edges are owned by selfish agents). We call this the *Extended Multicasting Game*.

The extended multicasting game has a flavor very similar to dual auctions. It is well known that in dual auctions, no reasonable approximation to efficiency can be achieved without making distributional assumptions about the input. McAfee [12] and later Tatur [15] have shown that using the assumption that all bidders derive their values from a common distribution, it is possible to achieve almost a $1 - 1/n$ fraction of the maximum efficiency by using a simple direct auction that rejects the least efficient trade.

Fiat et al [6] were the first ones to consider the question of profit maximization in the context of multicasting games. However, like others they dealt with the case when edge costs are known. Their mechanism either needs to be provided with the multicast tree, or it needs exponential time to construct the tree. They propose profit maximizing auctions and suggest that these could be used at every node to maximize profit in the multicasting game. There has been no further work in this area to our knowledge.

1.1 Our results

Our objective is a truthful polynomial time mechanism which aims to maximize the profit while constructing a multicast network. To get around the inapproximability of the problem, our mechanism outputs an approximately profitable solution if the optimal solution is sufficiently profitable. If the optimal (non-trivial) solution is sufficiently unprofitable, we demonstrate that no profitable solution exists. For other cases, we output a non-negative profit solution but give no guarantee for the amount of profit obtained. Since profit is always bounded above by the efficiency of the solution, our mechanism also gives guarantees for the efficiency of the solution when the optimal solution is sufficiently profitable. It also satisfies other desirable properties such as CS, IR and NPT (described in the next section). We call such a mechanism a *profit guaranteeing* mechanism. We make no assumptions on the distribution from which the utilities of the agents are drawn; however, we use a notion of *competitiveness* of the input, and justify the need for it. We also show that any mechanism that seeks to maximize profit must be of the same general form as our mechanism.

The rest of this paper is structured as follows. In the following section, we define the problem and discuss related work, some of which we use in our results. In Section 3, we describe some hardness results. Motivated by these, we define criteria for evaluating profit maximizing mechanisms in Section 4. We also discuss some general properties that any profit maximizing mechanism must have. Then in Section 4.2, we give a profit maximizing mechanism for the case of unknown edge values and known node values. In Section 4.3, we extend this mechanism to handle the case of unknown node values when there is sufficient competition among clients at nodes. Finally in Section 4.4, we give a budget balanced mechanism for the situation where there is not enough competition among clients. We conclude in Section 5.

2 Problem Definition and Related Work

We are given a graph $G = (V, E)$, with a root node $r \in V$ (the multicast source). Each node contains a set of *clients*, each with a private value (utility) u_i for receiving the multicast. The utility of a node is the sum of utilities of clients at the node. Let U denote the total utility of all clients in the graph. Each edge is also owned by an agent who has a private value c_e for its participation cost.

The multicaster is required to implement a mechanism that takes the following inputs: all clients bid the maximum amount that they are willing to pay for receiving the service and all edges bid their minimum asks for participating in the solution. The multicaster then computes a solution that consists of a set of clients that receive the service, a set of edges that connect these clients to the root node and a vector of fees taken from the clients (p_i) and payments made to the edges (p_e). Any residual money is the multicaster's profit, which we wish to maximize.

Formally, let T denote a multicast tree; T_V and T_E respectively denote the set of selected nodes and edges. Also, define $p(T_V) = \sum_{i \in T_V} p_i$ and $p(T_E) = \sum_{e \in T_E} p_e$. In terms of the original utilities of the agents, we also define $u(T_V)$ (abbreviated $u(T)$) and $c(T_E)$ (abbreviated $c(T)$). The profit of the solution is $\pi(T) = p(T_V) - p(T_E)$. We assume that all agents are rational (selfish) and wish to maximize their own profit. For a client i and a bid vector b, let $p_i(b)$ denote the payment asked from the client. We use indicator variables $1_{i \in T_V}$ ($1_{e \in T_E}$) which are 1 if node i (edge e) is selected for service, and 0 otherwise. The client bids $\text{argmax}_{b_i}\{u_i 1_{i \in T_V} - p_i(b)\}$. Similarly, edge e bids $\text{argmax}_{b_e}\{p_e(b) - c_e 1_{e \in T_E}\}$.

A mechanism is strategy-proof (SP) if for all clients i, $b_i = u_i$ is a dominant strategy irrespective of the bids of other agents and for all edges, $b_e = c_e$ is a dominant strategy. Formally, for all i and all vectors b, $u_i \in \text{argmax}_{b_i}\{u_i 1_{i \in T_V} - p_i(b)\}$, and for all edges $c_e \in \text{argmax}_{b_e}\{p_e(b) - c_e 1_{e \in T_E}\}$. We only consider strategy-proof mechanisms.

The following are some natural constraints which any mechanism must satisfy:

1. No Positive Transfers (NPT): We must not pay clients to participate; i.e., $p_i \geq 0$ for all clients i. Similarly, $p_e \geq 0$ for all edges e.

2. Individual Rationality (IR), also called Voluntary Participation: Every participating agent must have non-negative net utility. That is, $p_i \leq u_i \forall i \in V$, and $p_e \geq c_e \forall e \in E$.

3. Consumer Sovereignty (CS): For every client i, if u_i is increased high enough and all else is held constant, then client i is guaranteed service. There is no notion of CS for edges.

4. Polynomial Time Computability (PC): All computation is done in polynomial time.

There are two other objectives which are crucial, but can be compromised to some extent.

1. Efficiency: The efficiency of a solution T is $\phi(T) = u(T) - c(T)$. It follows from IR and NPT that the profit of a solution is always at most the efficiency of the solution, that is, $\pi(T) \leq \phi(T)$. Hence if we generate a solution with non-negative profit, then the efficiency of the solution is also non-negative. A mechanism is called efficient if it outputs $\text{argmax}_T \phi(T)$. Let T^* denote the most efficient solution.

2. Budget Balance (BB): A strong version of BB requires that $\pi(T) = 0$. However, in the context of profit maximization, we relax this to $\pi(T) \geq 0$.

2.1 Related Work

The field of *algorithmic mechanism design* was brought into focus by Nisan and Ronen [14]. They consider strategy-proof mechanisms for the shortest path problem (among others). They use the VCG mechanism, and show that in the worst case, the mechanism overpays the selected edges by an $O(n)$ factor of their cost. Archer and Tardos [1] extend that result and show that any strategy-proof mechanism in the class of *min-function* mechanisms must also overpay by the same factor.

The MST Game. Fortunately, this overpayment is not a problem for the minimum spanning tree problem. MST is a subproblem of our problem, that assumes that *every node* must be connected to the root. Bikhchandani, et al [2] give an elegant strategy-proof mechanism (which we call the *VST mechanism*) based on the Vickrey auction [16]. Essentially, if an edge is selected in the MST, its payment is the cost of the second-cheapest edge across the cut induced by the selected edge. We call this the *Vickrey price* of the edge that is selected in the MST. Edges which are not selected are paid nothing. We make use of this result to induce truthfulness of edges, using the following lemma.

Lemma 1 *The VST mechanism is strategy-proof even if only a subtree $T' \subseteq MST$ is selected, so long as the decision process for determining T' uses the tree's Vickrey prices as its edge costs.*

Proof: An edge which is not in the MST cannot get selected by bidding higher; if it bids lower, it can get selected only at a Vickrey price lower than its own utility, resulting in a loss. An edge which is in the MST cannot gain by bidding lower, since it is paid its Vickrey price which is independent of it. By bidding higher, it may get dropped.

Further, Vickrey payments are determined by edges which are not in the MST; hence, by changing its bid, an edge in the MST cannot affect the Vickrey payment to any other edge in the MST. □

Another special case of the extended multicast problem is when edge costs are known and all agents are at the nodes. Moulin and Shenker [13] study this problem in great detail, reviewing various Efficient and BB mechanisms including the Marginal Cost (each node is charged the marginal cost of serving it), and Shapley Value (the cost of each edge is distributed equally to each node downstream of it) mechanisms. Feigenbaum, et al [5] consider the computational cost of these mechanisms, and show them to be feasible only on tree networks. Jain and Vazirani [10] give an approximate budget-balanced group strategyproof mechanism which is polynomial time for any network and also gives a cost-sharing function for the nodes.

Cancellable Auctions. The recent research on cancellable auctions by Fiat, et al [6] is also of note. An auction is *cancellable* if the auctioneer has the option of canceling the auction if some pre-specified criterion (such as minimum revenue) is not met, and this does not affect the strategy of the participants. The notion of cancellability is helpful in multicast network problems because it allows for pruning of the multicast tree based on revenue achieved from nodes while still retaining truthfulness. Fiat, et al give a randomized auction that is cancellable and when run with at least two bidders, is guaranteed to generate at least $\frac{1}{4}$th of the revenue achievable by any truthful auction. We use the auction devised by them in one of our mechanisms in Section 4.3.

The Prize collecting Steiner tree problem. The Prize collecting Steiner tree problem (PCST) is closely related to the problem of finding a maximum efficiency multicast tree. Given (G, c, u, r), the objective of PCST is to find a tree T which minimizes $PC(T) = c(T) + u(\overline{T})$, where $u(\overline{T})$ is the sum of utilities of nodes not in the tree. For any tree T, we have $PC(T) = U - \phi(T)$, so that the two problems are equivalent from an optimization point of view. We use ρ to denote the approximation ratio of some algorithm which approximates PCST. Goemans and Williamson [8] give a $\rho = (2 - \frac{1}{|V|-1})$-approximation, the best known so far. We refer to their algorithm as GW, and use it in our work; a description of the algorithm as well as proofs of the following useful lemmas appear in the appendix.

Lemma 2 *Let T be the solution produced when GW runs on (G, c, u, r). Let G' be the subgraph of G induced by T_V. Then GW continues to output T when run on (G', c, u, r).*

Lemma 3 *Let G' be obtained from G by increasing the cost of a single edge e from c_e to c'_e. Then the set of nodes selected in the GW solution to G' is a subset of the set of nodes selected in G. Moreover, if e was not part of the solution in G, then it is not part of the solution in G' either.*

3 Hardness

It is well known that it is impossible to simultaneously achieve Efficiency and BB for a large class of mechanism design problems [4, 13, 14]. The multicasting problem is more notorious, as it is NP-hard to determine the most efficient solution, even when u and c are known. In fact it is also NP-hard to compute whether or not a non-trivial positive efficiency solution exists. This was first proved in [5]. We provide an alternate proof here, which leads to a stronger hardness of approximation result.

Theorem 4 *Given a graph $G = (V, E)$ with a root node $r \in V$, non-negative node utilities u_i, and non-negative edge costs c_e, it is NP-complete to determine whether there exists a tree T rooted at r such that $\phi(T) > 0$. Hence it is also impossible to approximate $\phi(T)$ within any factor at all.*

Proof: We use a reduction from the decision version of PCST to prove this. The decision version, also NP-hard, is to determine if there exists a tree T rooted at r with $c(T) + u(\overline{T}) \leq z$ (given $z < U$).

We construct a new graph G' by taking a copy of G and adding a new node r'. There is only one new edge, (r, r') of cost $U - z$. Define the natural bijection between trees in G rooted at r and trees in G' rooted at r' by $T(\in G) \mapsto T'(\in G')$. It is easy to see the following relation: $c(T) + u(\overline{T}) < z \iff u(T') - c(T') > 0$. Hence a polynomial time algorithm which decides the existence of a solution of positive efficiency implies a polynomial time algorithm for the decision version of PCST. Since an approximation algorithm for efficiency would still be able to decide whether or not there exists a tree with positive efficiency, it is also impossible to approximate efficiency within any factor. □

The following impossibility result due to Feigenbaum, et al [4] states that we cannot achieve a reasonable trade-off between Efficiency and BB even if exponential computation is allowed. This result holds even when edge costs are known and G is a tree. For completeness, we include a proof sketch.

Theorem 5 *A strategy-proof mechanism for multicast cost sharing satisfying NPT, IR and CS cannot achieve both γ-approximate efficiency and κ-approximate budget balance for any constants γ and κ.*

Proof Sketch: Consider a network in which there are p clients at a node, each with utility u and there is a single link from the root to the node of cost $(p-1)u + \delta$ with $\delta > 0$. Any γ-approximate efficient solution must include all the p clients even if one of them lowers its bid to $\delta + \epsilon < u$. Thus for SP, the mechanism can charge at most δ from each user. The mechanism can make at most $p\delta$ from clients and pays at least $(p-1)u + \delta$ to the edge. Thus for appropriate values of u and δ, it fails to be κ-approximate budget balanced. □

Comment 1 *Notice that if we take $p = 2$ in the above example, then even though the optimal solution is highly efficient ($\phi(T) \approx \frac{1}{2}U$), any budget balanced mechanism satisfying SP and other properties must output the empty solution and will fail to give any approximation to efficiency.*

Theorem 4 suggests that we should try to achieve an approximation for profit at least in the case when we are guaranteed that a sufficiently profitable solution exists. This motivates our definition of profit guaranteeing mechanisms in the next section. In order to define the antecedent that a "sufficiently profitable solution exists", we need a notion of the maximum revenue that can be raised from the input. In particular, for every node i, we let r_i denote the maximum revenue that any strategy-proof mechanism can raise from the agents at that node[1]. Note that this may be much less than the total utility at the node, but is a more reasonable target to compare with. R denotes the sum of r_i over all nodes, and $r(T) = \sum_{i \in T} r_i$.

[1] For a discussion of the maximum revenue that can be raised, refer to [6]

4 Profit Guaranteeing Mechanisms

Given such strong hardness results, it is not immediately clear how the performance of a mechanism for designing multicast networks should be measured. We give the following definition for a profit guaranteeing mechanism that we use as a yardstick in evaluating our mechanisms.

Definition 1 *An (α, β)-profit guaranteeing mechanism, where $\alpha \in [0,1]$ and $\beta \geq 1$, satisfies the following criteria:*

1. *SP, IR, NPT, CS, and PC.*
2. *If the optimal efficiency $\phi(T^*) = (1-\delta)R$, where $\delta < \alpha$, it finds a tree with profit at least $k(\delta)R$ where $k(\delta) \geq 0$ is decreasing in δ.*
   ($k(\delta)$ increases as $\phi(T^*)$ increases).
3. *If for every tree T, $c(T) \geq \beta r(T)$, it demonstrates that no non-trivial positive surplus tree exists.*
4. *If neither of the conditions in 2 and 3 above are met, the mechanism returns a solution with non-negative profit (possibly the trivial solution, with no node served and no edges selected).*

From Theorem 4, it is clear that it is impossible to obtain a mechanism that is (α, β)-profit guaranteeing with either $\alpha = 1$ or $\beta = 1$. From comment 1 we also get that $\alpha \leq \frac{1}{2}$. The best that can be done is to obtain as large α and as small β as possible.

We begin with a characterization of a mechanism that aims to maximize profit. Observe that without making any distributional assumptions, for a mechanism to be strategy-proof, the payments made to edges and received from clients should be bid-independent [1, 6]. That is, for an edge e, p_e should be a function of $b_{e' \neq e}$ and $b_{i \in V}$. Similarly, p_i should be a function of $b_{e \in E}$ and $b_{i' \neq i}$.

Now consider the following example: there is a single client i connected by a single edge e to the root. The client's utility is u and the edge's cost is c. If $c \geq u$, every mechanism outputs the empty solution. Consider the case when $c < u$. In order to induce strategy-proofness, we must charge the client $p_i = p_i(c)$ and pay the edge $p_e = p_e(u)$, where p_i and p_e are non decreasing functions with $p_i(c) \geq c$ and $p_e(u) \leq u$. Also, the mechanism should output an empty solution if $p_i(c) < p_e(u)$. Since p_i and p_e are non decreasing, given a value of c, if we increase the value of u, the profit made by the mechanism decreases or stays the same. On the other hand, the efficiency of the solution increases. Thus such a mechanism cannot achieve profit that is a constant fraction of the efficiency of the optimal solution. This justifies our argument that we cannot compare our mechanism's performance with the maximum efficiency; instead, we should compare it with some other measure of the maximum achievable profit.

The next lemma shows that in order to obtain a constant fraction of the achievable profit, a mechanism should use other edges' (clients') values to determine the payment (fee) to an edge (client). Moreover, for achievable profit to be high, there should be some amount of competition among the clients and edges.

Lemma 6 *Let M be a mechanism that is (α, β)-profit guaranteeing for some constants α and β. Let p_V be a function specifying the total fees charged by M from all clients and p_E be the function used by it to determine the total payment for all edges. Then, p_V should be a function of the utilities of clients (among other inputs) and p_E should be a function of costs of edges (among other inputs).*

Proof: Assume to the contrary that the claim does not hold and (without loss of generality) p_V does not depend on the utilities of clients.

Consider the behavior of M on the following example: There are two clients at the same node and two edges connecting this node to the root. Let the respective utilities and costs be $u_1 = u_2 = u$, and $c_1 = c_2 = c$. Now, let $r(u)$ be the maximum revenue that can be generated at the node by a strategy-proof mechanism. Notice that $r(u)$ is strictly increasing in u. Observe that a truthful auction can make a profit of $r(u) - c$ by running a Vickrey auction on edges and generating $r(u)$ from the node. Let u be high enough such that $r(u) - c > (1-\alpha)r(u)$.

M makes a profit of $\pi(M) = p_V(c) - p_E(c, u)$. Since M is (α, β)-profit guaranteeing, this should be at least $k(\alpha)(r(u) - c)$, where $k(\cdot)$ is the guarantee given by the mechanism.

Let us increase u and decrease c simultaneously. If p_E is a function of u, it has to be non decreasing in u, because otherwise as u increases, there exist sufficiently low values of c for which $p_E < c$ and edges will refuse to participate, while $r(u) - c > (1-\alpha)r(u)$, violating the (α, β)-profit guarantee of M.

Now keeping c constant, we increase u. Notice that $\pi(M)$ decreases or stays constant, while $r(u) - c$ increases. Thus at some point, $\pi(M) < k(\alpha)(r(u) - c)$. This contradicts the fact that M is (α, β)-profit guaranteeing.

□

The above lemma suggests that a profit guaranteeing mechanism should have the following form: It should run profit maximizing auctions among clients and among edges separately, and then combine these using some algorithm to determine the final solution. This is a basic outline of our profit guaranteeing mechanism.

Some details still need to be taken care of. We should determine the set of clients/edges in the final solution *after* we have run the auctions so that we know the amount of payment to be made to edges and obtained from nodes. However, for this mechanism to remain strategy-proof, these auctions should be cancellable (as defined in Section 2.1). Further, in order to achieve good revenue from the auctions, we need our input to be *competitive*.

4.1 Competitive Inputs

The preceding discussion motivates the need for competition among the agents. As observed in previous studies [1, 6, 7], one cannot hope to achieve a good approximation to profit in a highly uncompetitive market. In the case of auctions, if a market has a single high value agent, no strategy-proof mechanism can extract a reasonable fraction of his utility. On the other hand, the best fixed price mechanism that has knowledge of bidders' values, would simply ask the highest bidder for all his value.

Similarly in the case of multicast mechanisms, if there is a cut containing only one edge, there is no way of inducing truthtelling for that edge without paying huge sums of money to it. This is similar to a monopoly situation. Hence we restrict our attention to *competitive* markets, where there are sufficiently many players so that they can play against each other and the mechanism benefits from the advantages of a free-market system. We use the following two definitions of competition among edges and nodes.

Definition 2 *Consider a graph $G = (V, E)$ with edge costs c. For a cut $[S_1 : S_2]$, let c_{S_1,S_2} denote the cheapest edge across the cut. G is ϵ-competitive if there is a constant ϵ such that for every set $V' \subseteq V$, across every cut $[S : V' \setminus S]$, there are at least two edges of cost no more than $(1+\epsilon)c_{S,V'\setminus S}$.*

Definition 3 *Consider a graph $G = (V, E)$. G is node competitive if there are at least two clients at every node.*

A graph is called *ϵ-competitive* if it is both ϵ-edge competitive and node competitive.

4.2 Known Node Utilities

In this section we design and analyze a profit guaranteeing mechanism for designing multicast networks when node utilities are known but the edges are selfish agents. We assume that the graph is ϵ-edge competitive. Notice that when node utilities are known, we do not have to bother about truthfulness at nodes, and so we can charge them their true values. Thus we have $r = u$ at every node. In this section, we use r and u interchangeably.

Before describing the mechanism, we first consider the simple case where the set of nodes being served has been determined and we are to only select edges and assign payments to them. Clearly, when the set of nodes to be served is known, the most efficient solution is to pick the minimum Steiner tree on these nodes. Since computing the minimum Steiner tree is NP-hard, we adopt the simpler solution of picking the MST which is a 2-approximation to the cost of the Steiner tree. We assign Vickrey payments to edges as described in Section 2.1. Also, ϵ-competitiveness of the input guarantees that these payments will never be more than $(1+\epsilon)$ times the cost of the MST, which is $2(1+\epsilon)$ times the cost of the most efficient solution.

We will use the above as a subroutine in the main algorithm. Our mechanism is as follows.

Mechanism M_1

1. Ask edges to reveal their bids.
2. Use GW to approximately solve PCST using the revealed edge and node utilities. Let V' be the set of nodes selected for service, and define $G' = (V', E')$ to be the subgraph of G induced by V'.
3. Construct a minimum spanning tree T_1 on G'. The edges in T_1 are paid their Vickrey prices. Prune the solution from bottom up to improve its efficiency based on Vickrey prices on edges[2].:
   (a) For each node i, let $e(i)$ denote its parent edge in the MST and $ch(i)$ its children nodes.
   (b) Compute the *surplus* of each node as follows. The surplus of a leaf node is $\pi(i) = p_i - p_{e(i)}$. For an internal node, the surplus is $p_i - p_{e(i)} + \sum_{j \in ch(i): \pi(j) > 0} \pi(j)$.
   (c) Identify all nodes with negative surplus. Delete the subtrees rooted at these nodes. Call this pruned solution T.
4. If the GW solution T_1 is non trivial, return the solution T.
5. If T_1 is trivial, rescale all the node utilities to $u'_i = 2u_i$ and rerun GW. If the algorithm again returns the trivial solution, output "Fail, no positive solution exists".

Lemma 7 *The pruning subroutine does not decrease profit; that is, $\pi(T) \geq \pi(T_1)$.*

Proof: The pruning step only deletes subtrees with negative surplus. □

Lemma 8 *Mechanism M_1 is strategy-proof.*

Proof: An edge which is in T has no incentive to change its bid, because its payment is independent of its bid. Moreover, raising its bid may cause it to be dropped from the solution (Lemma 3).

Now consider an edge that is not in the final solution. If the edge decreases its bid, it might get selected in the GW solution. If the Vickrey payment is less than the original price, then it makes a loss. Now consider the case that the Vickrey payment is more than its price. Since the edge was not in the GW solution with its original cost, this suggests that the nodes that it is connecting are not able to pay for its cost. In that case, the pruning step with Vickrey payments will remove this edge. On the other hand, if the edge increases its bid, it will still not be selected (Lemma 3), and will continue to receive zero payment. Finally, Lemma 1 shows that the pruning step does not induce dishonesty in edges, since we are only pruning edges of the MST T_1. □

Lemma 9 *If $\phi(T^*) = (1 - \frac{1}{\gamma})U$, where $\gamma > 2(1+\epsilon)\rho$, then the mechanism finds a budget balanced multicast tree with efficiency at least $(1 - \frac{\rho}{\gamma})U$ and profit at least $(1 - \frac{2(1+\epsilon)\rho}{\gamma})U$.*

Proof: Suppose T^* is a tree such that $\phi(T^*) = (1 - \frac{1}{\gamma})U$. Then the value of the PCST objective for this tree is $PC(T^*) = \frac{1}{\gamma}U$. Hence GW finds a tree T_1 such that $PC(T_1) \leq \frac{\rho}{\gamma}U$. So we have $\phi(T_1) \geq (1 - \frac{\rho}{\gamma})U$. Using Lemma 7, the efficiency of the solution is $u(T) - c(T) = \phi(T) \geq \phi(T_1) \geq (1 - \frac{\rho}{\gamma})U$.

Next, using the fact that a spanning tree is a 2-approximation for a Steiner tree and the Vickrey price of each edge is no more than $(1+\epsilon)$ times the edge cost, we find that we pay $p(T_E) \leq 2(1+\epsilon)c(T)$. Moreover, we are extracting $u(T)$ payments from the edges, so $p(T_V) = u(T)$. Hence we have $\pi(T) \geq (1 - \frac{2(1+\epsilon)\rho}{\gamma})U$. This is non-negative, since $\gamma > 2(1+\epsilon)\rho$. □

Lemma 10 *If M_1 outputs "Fail", then there us no non-trivial tree T with $c(T) < u(T)$.*

Proof: Say T is a tree with $c(T) < u(T)$. Then, $2c(T) < u'(T)$. Consider the behavior of GW on the rescaled utilities on the graph $G[T_V]$. Now, the solution containing the entire tree on this new problem has (PC) cost $c(T)$. Since GW is a 2-approximation, it will produce a solution of (PC) cost at most $2c(T)$ (using Lemma 2). Thus it cannot output the trivial solution that has (PC) cost at least $2c(T)$. □

[2]This subroutine is similar to the pruning subroutines of Feigenbaum, et al [3] and Johnson, et al [11].

Lemma 11 *If for every tree T in the graph we have $c(T) > 4u(T)$, then M_1 outputs "Fail".*

Proof: Assume that GW on original utilities returns a non-trivial tree T. Then, if GW is run on the graph restricted to T_V, it still returns the entire tree (Lemma 2). However, this contradicts the 2-approximation of GW, since $c(T) > 2u(T)$. Now on the rescaled utilities, we still have $c(T) > 4u(T) > 2u'(T)$. Thus by the same argument as before, GW returns the trivial solution. □

Lemma 12 *If M_1 returns a non-trivial solution, the solution is budget balanced and has non-negative efficiency.*

Proof: Step 3 of Mechanism M_1 guarantees that the tree returned is budget balanced, that is, $\pi(T) \geq 0$. Since $\phi(T) \geq \pi(T)$, we find that the tree returned has non-negative efficiency $\phi(T)$. □

It is easy to check from the definition of M_1 that NPT, IR, CS and PC are satisfied. We therefore have the following theorem.

Theorem 13 *Mechanism M_1 is a $(\frac{1}{2(1+\epsilon)\rho}, 4)$-profit guaranteeing mechanism.*

4.3 Unknown Node Utilities with Competition at Nodes

To adapt M_1 to the case where there are selfish users at nodes with unknown utilities, we need the input graph to be ϵ-competitive. Moreover, we no longer have $r_i = u_i$.

Mechanism M_2 for this case adds a new step, (say 1(a)), where we run a cancellable auction at each node. If r_i is the maximum achievable revenue at each node, we know that the SCS auction of Fiat, et al [6] recovers at least $r'_i = r_i/4$ at each node. We now treat r'_i as known node utilities, and continue with Step 2 onwards of M_1. If the mechanism reaches Step 5, it uses r_i instead of r'_i.

There are two differences between this mechanism and M_1. Firstly, in this case we compare the profit achieved with the maximum achievable profit, which in turn is defined in terms of maximum achievable revenue at each node. Secondly, M_2 satisfies only a weaker notion of CS. That is, each *node* is guaranteed that some of its clients will get service if the maximum achievable revenue at that node is sufficiently high. There is no guarantee of CS in terms of single *clients*, due to the results in [6].

Lemma 14 *If $r(T^*) - c(T^*) = (1 - \frac{1}{\gamma})R$, where $\gamma > 8(1+\epsilon)\rho$, M_2 finds a budget balanced multicast tree with efficiency at least $\frac{1}{4}(1 - \frac{\rho}{\gamma})R$ and profit at least $\frac{1}{4}(1 - \frac{2(1+\epsilon)\rho}{\gamma})R$.*

The proof of the above lemma is similar to the proof of Lemma 9, and uses the fact that $r'_i \geq r_i/4$ for all i. Lemmas 7, 8, 10, 11 and 12 continue to hold. We therefore have the following theorem.

Theorem 15 *M_2 is a $(\frac{1}{8(1+\epsilon)\rho}, 4)$-profit guaranteeing mechanism when both edges and nodes are players and the input graph is ϵ-competitive.*

4.4 Unknown Node Utilities Without Competition

Finally, we consider the case where node utilities are not known, and there is no competition at nodes, i.e. some nodes have only one user. For nodes with multiple users, we use the simple trick of creating a new node for each client and attaching this to its original node using a zero length edge. Thus the mechanism assumes that there is a single user at every node.

We cannot use a strategy-proof mechanism and still maximize profit, since in the absence of competition, nodes have no incentive to add to our profit. However, it is possible to give a strategy-proof mechanism which always has non-negative profit (and thus non-negative efficiency), although it is no longer profit-guaranteeing. **Mechanism** M_3 is defined as follows.

1. Obtain bids c_e from the edges, and u_i from the clients.

2. Eliminate all nodes which have utility zero.

3. Build an MST on the remaining nodes, and let the price of each edge be its Vickrey price.

4. Use the Shapley Value mechanism (SV)[3] of [13] to divide cost of edges among nodes. (Prune the tree if required by the SV mechanism).

Lemma 16 *Mechanism M_3 is strategy-proof for nodes and edges.*

Proof: M_3 is strategy-proof for nodes because SV is strategy-proof. For the edges, strategy-proofness follows from Lemma 1. □

Theorem 17 *Mechanism M_3 has non-negative profit and efficiency.*

Proof: This is by definition, since the SV cost allocation function guarantees that we prune the tree until we have non-negative profit. □

Even though the above mechanism is not a profit guaranteeing mechanism for any $\alpha > 0$, it satisfies Conditions 1 and 4 of Definition 1, using the above lemmas. Furthermore, if the mechanism outputs the trivial solution, we can run GW on the entire graph and then again with rescaled node utilities (as in M_1) and if both the runs return the trivial solution, we output "No positive solution". (Notice that we are again using $r_i = u_i$.) It follows from Lemmas 10 and 11 that this satisfies Condition 3 with $\beta = 4$. As for Condition 2, an example in the appendix shows that even if there is a highly profitable input, M_3 could end up with a zero profit solution; though the condition is trivially satisfied with $\alpha = 0$.

Theorem 18 *Mechanism M_3 is a $(0, 4)$-profit guaranteeing mechanism.*

4.5 Running Time and Network Complexity

GW runs in $O(n^2 \log n)$ time, where $n = |V|$. The pruning procedures require $O(n^2)$ time, so the running time of the mechanisms are $O(n^2 \log n)$. Assuming that all mechanisms are run at the root, we need $O(|E| + |V|)$ messages to obtain the bids. Moreover, our mechanisms have been defined such that the only other communication is when the root tells the agents the final outcome.

5 Conclusions

The hardness results make it clear that in the absence of any distributional assumptions, it is impossible to obtain any approximation to profit in arbitrary graphs for the problem of designing multicast networks when both edges and nodes are agents. In this paper, we give mechanisms that approximate profit and efficiency when there exists a highly efficient solution. We define the concept of *profit guaranteeing* mechanisms, which is one reasonable way of measuring the performance of such mechanisms, where even deciding whether there exists a non-trivial efficient solution is NP-hard. Improving the parameters of our profit guaranteeing mechanisms remain open. It would also be interesting to see if such mechanisms exist for other hard problems.

Acknowledgements

We wish to thank Avrim Blum and Christine Parlour for several useful discussions.

References

[1] A. Archer and E. Tardos. Frugal path mechanisms. In *Proc. 13th ACM-SIAM Symposium on Discrete Algorithms*, 991-999, 2002.

[2] S. Bikhchandani, S. de Vries, J. Schummer and R. Vohra. Linear programming and Vickrey auctions. *Manuscript*, 2001.

[3] J. Feigenbaum, A. Krishnamurthy, R. Sami and S. Shenker. Approximation and collusion in multicast cost sharing. In *Proc. 3rd ACM Conference on Electronic Commerce*, 2001.

[3] Any cost division mechanism can be used here; two are described in the appendix.

[4] J. Feigenbaum, A. Krishnamurthy, R. Sami and S. Shenker. Hardness results for multicast cost sharing. *Manuscript*, 2002.

[5] J. Feigenbaum, C. Papadimitriou and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63:21-41, 2001.

[6] A. Fiat, A. Goldberg, J. Hartline and A. Karlin. Competitive generalized auctions. In *Proc. 34th ACM Symposium on Theory of Computing*, 72-81, 2002.

[7] A. Goldberg, J. Hartline and A. Wright. Competitive Auctions and Digital Goods. In *Proc. 12th Symposium on Discrete Algorithms*, 735-744, 2001.

[8] M. Goemans and D. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal of Computing*, 24(2):296-317, 1995.

[9] S. Herzog, S. Shenker and D. Estrin. Sharing the cost of multicast trees: An axiomatic analysis. *Transactions on Networking*, 5(6):847-860, 1997.

[10] K. Jain and V. Vazirani. Applications of approximation algorithms to cooperative games. In *Proc. 33rd ACM Symposium on Theory of Computing*, 364-372, 2001.

[11] D.S. Johnson, M. Minkoff and S. Phillips. The prize collecting Steiner tree problem: Theory and practice. In *Proc. 11th Symposium on Discrete Algorithms*, 760-769, 2000.

[12] R. P. McAfee. A dominant strategy double auction. *Journal of Economic Theory*, 56:434-450, 1992.

[13] H. Moulin and S. Shenker. Strategyproof sharing of submodular costs: budget balance vs efficiency. *Economic Theory*, 18:511-533, 2001.

[14] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proc. 31st ACM Symposium on Theory of Computing*, 129-140, 1999.

[15] T. Tatur. Asymptotically optimal market mechanisms. *Working Paper, Northwestern University*, 2001.

[16] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8-37, 1961.

A Appendix

A.1 The GW Algorithm for PCST and Proofs

Consider the following integer programming formulation of PCST. For a subset of nodes R, let z_R be 1 if that subset is not spanned by the selected tree. Then the following IP (**PCIP**) describes PCST:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e + \sum_{R \subset V : r \notin R} z_R u(R) \\ & x(\delta(S)) + \sum_{R \supseteq S} z_R \geq 1 \quad \forall S \subset V : r \notin S \\ & \sum_{R \subset V : r \notin R} z_R \leq 1 \end{aligned}$$

The dual of the linear relaxation of PCIP is **PCD**:

$$\begin{aligned} \max \quad & \sum_{S : r \notin S} y_S \\ & \sum_{S : e \in \delta(S)} y_S \leq c_e \quad \forall e \in E \\ & \sum_{S \subseteq R} y_S \leq u(R) \quad \forall R \subset V : r \notin R \end{aligned}$$

GW is a primal-dual algorithm, which begins by setting all dual variables y_S to zero. Sets S which are inclusion wise minimal and such that their corresponding primal constraints are violated are called *valid* sets. GW raises the dual values of valid sets simultaneously while maintaining dual feasibility. If an edge constraint in the dual becomes tight, the corresponding edge has its primal value set to 1, and is selected in the solution. Duals are raised only until the total utility inside the sets can pay for them. Sets that exhaust all the utility inside them (and are thus unable to grow their duals) are called *dead*. When no dual variable can be raised any further, the algorithm stops and goes into a second stage. At the end of the first stage, there are some nodes connected to the root by a tree. Other nodes belong to sets that died before reaching the root.

In the second stage (the pruning stage), all nodes that are not connected to the root are discarded. Nodes belonging to sets that did not die before reaching the root are retained and connected to the root. Other nodes that belong to dead sets, but were still connected to the root at the end of the first step are retained if they are on the path from the root to some other alive set. The retained nodes and tree on them form the final solution. Goemans et al [8] prove that the total utility of all nodes in the final solution is at least $\frac{1}{2}$ the cost of the tree connecting them. This gives a 2-approximation to the problem of minimizing $c(T) + u(\overline{T})$.

For a detailed description of the algorithm and its analysis, the reader is referred to [8].

Proof of Lemma 2: The statement simply follows from the observation that the behavior of GW in terms of the dual variables which are affected by T is unaffected by the rest of the graph. Hence if we restrict our attention to G', the behavior of GW is unchanged, and it returns T. □

Proof of Lemma 3: Suppose edge e has its cost increased by a small amount, from c_e to c'_e. If the edge was not tight in G, then clearly it will not be tight in G' either. If it was tight in G, then it may or may not be tight in G'. However, from the description of the algorithm, it follows that the set of nodes selected in G' cannot include a node which wasn't selected in G. These two observations complete the proof. □

A.2 Mechanism M_3 is not profit guaranteeing

The following example shows that even in the presence of a solution with extremely high surplus, Mechanism M_3 may return the trivial solution. The graph consists of $n+2$ nodes (one being the root), arranged in a cycle. One of the neighbors of the root is a special node with utility k. This is connected to the root with an edge of cost 2. All other nodes have utility δ, and all other edges have cost 1. Let δ be small enough so that $n\delta << k$. Hence $U \approx k$. Moreover, k is such that $k + n\delta < n$.

The mechanism will first pick the MST, which is all the edges of cost 1. The MST costs n. Any kind of cost allocation will prune out all the nodes, and return the trivial solution. However, the solution consisting of just the special node and the edge of cost 2 has efficiency $k - 2 \approx U$.

It can also be shown that "filtering" nodes with very low utility does not help, by a simple modification of the above example.

A.3 Pruning Rules and Cost Division Methods

Suppose we are given a rooted tree T, with edge costs c_e and node utilities u_i. We want to select a subtree T' of T and allocate costs $p(i)$ to the nodes, such that $p(i) \leq u_i$ for all nodes i and $\sum_{i \in T'} p(i) = c(T')$, that is, the cost of the subtree is recovered from the nodes. Such a cost allocation function is called budget-balanced. Given any cost allocation function p, the following is a pruning subroutine which computes T':

1. Set $T' = T$.
2. Mark all nodes "alive".
3. For all alive nodes i, compute $p(i)$.
4. While there exists a node with $p(i) > u_i$:
   (a) Mark all nodes with $p(i) > u_i$ to be "dead".
   (b) For every edge, if all nodes downstream of it are dead, delete the edge from T'.
   (c) Recompute $p(i)$ for all alive nodes i to distribute the costs of the edges in T' to the alive nodes.

5. Return T', and the set of alive nodes.

The *Shapley Value* cost allocation function [5, 13] is as follows. For any edge e, let a_e denote the number of alive nodes downstream of it. For any node i, let $E(i)$ be the set of edges in the path from the node to the root. Then we define $p(i) = \sum_{e \in E(i)} c_e/a_e$. In other words, the cost of every edge is distributed equally among the alive nodes downstream of it.

Another budget-balanced cost allocation function is the Jain-Vazirani function (JV), derived from the Primal-Dual algorithm for computing a minimum spanning tree. Essentially, each alive node maintains a counter, and all counters are initially at zero. All counters go up at the same rate, and initially, each counter "loads" the edge immediately upstream of the node. When the sum of counter values loading an edge equals the cost of the edge, we say the edge is "tight", and all counters loading the edge are moved up to load the edge immediately upstream of it, with their values reset to zero. For each node, this goes on until the counter finishes loading the edge adjacent to the root. At this point, the node i is charged $p(i)$ which is defined to be the sum of the values reached by all its counters on the path between the node and the root. For a detailed discussion of the JV function, the reader is referred to their paper [10].