Strongly Polynomial-Time Truthful Mechanisms in One Shot*

Paolo Penna[1], Guido Proietti[2], and Peter Widmayer[3]

[1] Dipartimento di Informatica ed Applicazioni "Renato M. Capocelli", Università di Salerno, Via **S. Allende** 2, 84081 Baronissi (SA), Italy
penna@dia.unisa.it

[2] Dipartimento di Informatica, Università di L'Aquila, Via Vetoio, 67010 L'Aquila, Italy, and Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti", CNR, Viale Manzoni 30, 00185 Roma, Italy
proietti@di.univaq.it

[3] Institut für Theoretische Informatik, ETH Zürich, CAB H 15 Universitätstrasse 6, 8092 Zürich, Switzerland
widmayer@inf.ethz.ch

Abstract. One of the main challenges in *algorithmic mechanism design* is to turn (existing) efficient algorithmic solutions into efficient *truthful mechanisms*. Building a truthful mechanism is indeed a difficult process since the underlying algorithm must obey certain "monotonicity" properties and suitable *payment functions* need to be computed (this task usually represents the bottleneck in the overall time complexity).

We provide a general technique for building truthful mechanisms that provide optimal solutions in *strongly polynomial time*. We show that the *entire* mechanism can be obtained if one is able to express/write a strongly polynomial-time algorithm (for the corresponding optimization problem) as a "suitable combination" of simpler algorithms. This approach applies to a wide class of *mechanism design graph problems*, where each selfish agent corresponds to a weighted edge in a graph (the weight of the edge is the cost of using that edge). Our technique can be applied to several optimization problems which prior results cannot handle (e.g., MIN-MAX optimization problems).

As an application, we design the first (strongly polynomial-time) truthful mechanism for the *minimum diameter spanning tree* problem, by obtaining it directly from an existing algorithm for solving this problem. For this non-utilitarian MIN-MAX problem, no truthful mechanism was known, even considering those running in exponential time (indeed, exact algorithms do not necessarily yield truthful mechanisms). Also, standard techniques for payment computations may result in a running time which is not polynomial in the size of the input graph. The overall

* Work partially supported by the Research Project GRID.IT, funded by the Italian Ministry of Education, University and Research, by the European Project IST-15964 "Algorithmic Principles for Building Efficient Overlay Computers" (AEOLUS), by the European Union under COST 295 (DYNAMO), and by the Swiss BBW. Part of this work has been developed while the first and the second author were visiting ETH.

P. Spirakis et al. (Eds.): WINE 2006, LNCS 4286, pp. 377–388, 2006.

running time of our mechanism, instead, is polynomial in the number n of nodes and m of edges, and it is only a factor $O(n\,\alpha(n,n))$ away from the best known canonical centralized algorithm.

1 Introduction

The emergence of the Internet as the platform for distributed computing has posed interesting questions on how to design efficient solutions which account for the lack of a "central authority" [11,15,16]. This aspect is certainly a key ingredient for the success of the Internet and, probably, of any "popular" system that one can envision (peer-to-peer systems are a notable example of this type of anarchic systems). In their seminal works, Koutsoupias and Papadimitriou [11] and Nisan and Ronen [15], suggest a *game-theoretic* approach in which the various "components" of the system are modeled as *selfish agents*: each agent performs a "strategy" which results in the highest *utility* for him/her-self. For instance, each agent may control a link of a communication network and each link has a cost for transmitting (i.e., for using it). A protocol that wishes to establish a minimum-cost path between two nodes would have to ask the agents for the cost of the corresponding link [15,2]. An agent may thus find it to be in his/her interest to lie about his/her costs (e.g., an agent might untruthfully report a very high cost in order to induce the protocol to use an alternative link, and thus no cost for the agent). Nisan and Ronen [15] propose a *mechanism design* approach that combines an underlying algorithm (e.g., a shortest-path algorithm) with a suitable payment function (e.g., how much we pay an agent for using his/her link). The idea is to come up with a so called *truthful mechanism*, that is, a combination of an algorithm with payments which guarantee that no agent can improve his/her own utility by misreporting his/her piece of private information (e.g., the cost of his/her link). Unfortunately, the design of truthful mechanisms is far from trivial and known results, originally developed in the microeconomics field [21,3,5,13], pose new algorithmic challenges which are the main subject of *algorithmic mechanism design* (see e.g. [4]).

Some interesting classes of problems (including a family of mechanism design graph problems considered here and in a number of works [15,7,6,10]) require the underlying algorithm to be *monotone* (e.g., if the algorithm selects an edge then it cannot drop this edge if its cost gets smaller and everything else remains the same). Though this condition suffices for the existence of a truthful mechanism [13,2], it is not clear how to guarantee this property nor how the corresponding payment functions can be efficiently computed (see e.g. [9,14]).

Mu'Alem and Nisan [12] were the first to propose a general method for constructing monotone algorithms (and thus truthful mechanisms). Basically, their approach consists of a set of "rules" to combine monotone algorithms so that the final combination results in a monotone algorithm as well. As observed by Kao *et al.* [10], the method in [12] does not provide an *efficient* way of computing the payments. Kao *et al.* [10] then extend some of the techniques in [12] and provide an efficient way for computing the corresponding payment functions.

Kao *et al.*'s approach [10] represents a significant progress towards a general technique which accounts for computational issues, though it cannot be applied to some very basic graph problems (e.g., a problem recently tackled in [19] – we discuss this issue more in detail below).

1.1 Our Contribution

In this work, we turn one of the main results in [12] into a general technique for building optimal truthful mechanisms running in *strongly polynomial time* (optimality refers to the quality of the computed solution). We show that the *entire* mechanism can be obtained if one is able to express/write an algorithm (for the corresponding optimization problem) as a "suitable combination" of simpler ones (see Section 2 and Theorem 1 therein). Obviously, the resulting mechanism is optimal and/or runs in strongly polynomial time if the algorithm does. However, neither of these conditions is required by our technique to guarantee truthfulness. This approach applies to a wide class of *mechanism design graph problems*, where each selfish agent corresponds to a weighted edge in a graph (the weight of the edge is the cost of using that edge). Our technique can deal with problems in which the cost function "underlying" the algorithm(s) is any *monotonically non-decreasing* function in the edge weights of the graph (i.e., in the costs of the agents). Since this includes several *non-utilitarian*[1] problems (e.g., MIN-MAX optimization functions), the results in [12] extend "only partially", that is, truthfulness can be guaranteed but the payments computation cannot be done "directly" by computing the "alternative" solution in which an agent is removed from the input (see e.g. [15,9]). We indeed observe that, for the problems considered in this work (see the discussion in Example 1), the payments computation is more complex than the case of *monotonically increasing* optimization functions, which are assumed in both [12] (where the problem is utilitarian) and in [10] (this assumption precedes Theorem 10 in [10] and the applications therein consist exclusively of utilitarian graph problems).

In Section 3, we apply our technique to the *minimum diameter spanning tree* problem and obtain the first (strongly polynomial-time) mechanism for it. For this non-utilitarian MIN-MAX problem, no truthful mechanism was known, even considering those running in exponential time (indeed, exact algorithms do not necessarily yield truthful mechanisms – see [17]). Also, standard techniques for payment computations may result in a running time which is not polynomial in the size of the input graph (see discussion in Example 1). The overall running time of our mechanism is instead $O(mn^2\,\alpha(n,n))$, and thus is only a factor $O(n\,\alpha(n,n))$ away from the best known algorithm for this problem [8], where $\alpha(\cdot,\cdot)$ is the classic inverse of the Ackermann's function. For two-edge connected graphs we also guarantee the *voluntary participation* condition, that is, no truthful agent runs into a loss (see next section for a formal definition). The minimum

[1] An optimization problem is called utilitarian if the goal is to minimize the sum of all agents costs or, equivalently, to maximize the sum of all agents valuations. Utilitarian graph problems have been studied in [15,9,10].

diameter spanning tree has both theoretical and practical relevance (e.g., in a peer-to-peer system we may want to set up a loop-free logical network using the resources – links – of a physical network so that any two peers can communicate efficiently).

The results for the minimum diameter spanning tree are paradigmatic of what happens when considering certain non-utilitarian mechanism design problems (another case is the *minimum radius spanning tree* problem [19] described in Example 1). First, one has to determine whether an existing algorithm can be turned into a truthful mechanism, whether a new one is needed, or if none can serve for this purpose [15,2,19]. In case a suitable algorithm exists, one has to find out how to compute the corresponding payments efficiently, possibly without burdening the complexity of the chosen algorithm [9,19]. Our technique can be used to give a positive answer to both questions, and thus to obtain the efficient mechanism in "one shot" (see Theorem 1).

We discuss other possible extensions and applications of our technique in Section 4 (these include a mechanism for the *p-center* graph problem and an improvement in the running time of the mechanism for the *minimum radius* in [19]).

1.2 Mechanism Design Graph Problems

Consider problems in which we are given a graph $G = (V, E)$ and the set of feasible outcomes consists of a suitable set $\mathcal{O} = \mathcal{O}(G)$ which depends only on the combinatorial structure of the graph (e.g., it consists of certain subgraphs of G). We have one agent per edge and the type $t_e \in \mathbb{R}^+$ of agent e is nothing but the *weight* of edge $e \in E$. Each solution $Y \in \mathcal{O}$ *uses* a subset of the edges of G; in particular, if Y uses edge e, then agent e has a cost (for implementing this outcome) equal to t_e. This scenario is common to several problems considered in the algorithmic mechanism design community: shortest-path [15], minimum spanning tree [15], shortest-paths tree [7], minimum-radius spanning tree [19]. Consider an agent e and let $\mathbf{r}_{-e}$ denote the values reported by the other agents, that is, $\mathbf{r}_{-e} = (r_1, \ldots, r_{e-1}, r_{e+1}, \ldots, r_m)$. When agent e reports x and the other agents report $\mathbf{r}_{-e}$, algorithm A computes a feasible outcome $A(x, \mathbf{r}_{-e})$. (That is, the algorithm returns a solution on input the vector $(x, \mathbf{r}_{-e}) := (r_1, \ldots, r_{e-1}, x, r_{e+1}, \ldots, r_m)$.) We say that declaration x is a *winning declaration* if solution $A(x, \mathbf{r}_{-e})$ uses edge e. A *mechanism* $M = (A, P)$ associates a payment $P_e(x, \mathbf{r}_{-e})$ with every agent e whose declaration x is a winning declaration (given the other agents' declarations $\mathbf{r}_{-e}$). This determines the *utility* of agent e:

$$u_e^M(x, \mathbf{r}_{-e}) := \begin{cases} P_e(x, \mathbf{r}_{-e}) - t_e & \text{if } A(x, \mathbf{r}_{-e}) \text{ uses } e, \\ 0 & \text{otherwise.} \end{cases}$$

Mechanism M is a *truthful* mechanism (with dominant strategies) if every function $u_e^M(x, \mathbf{r}_{-e})$ is maximized for $x = t_e$, for all $\mathbf{r}_{-e}$. We are interested in truthful mechanisms which optimize some objective function $\mu(Y, \mathbf{t})$ depending on the agents types $\mathbf{t} = (t_1, \ldots, t_m)$. Notice that the mechanism will work on the

reported types $\mathbf{r}$. Hence, truthfulness guarantees that, if the algorithm returns an optimal solution for the given input, then the mechanism outputs an optimal solution w.r.t. the true types. We will also consider mechanisms which satisfy the *voluntary participation*, that is, a truthful agent is guaranteed to have a non-negative utility (i.e., $u_e^M(t_e, \mathbf{r}_{-e}) \geq 0$). This property will be achieved whenever there exists an "alternative" solution that does not use edge e, i.e., $\mathcal{O}(G-e) \neq \emptyset$.

2 A Technique for Efficient Truthful Mechanisms

Our approach consists in defining an optimal algorithm A as a "suitable combination" of simpler ones. For minimization problems, we combine algorithms by means of the following 'MIN' operator, which is essentially the same as the 'MAX' operator by Mu'Alem and Nisan [12]:

$\mathbf{MIN}_\mu(A_1,A_2)$ operator
- compute $Y_1 = A_1(\mathbf{r})$ and $Y_2 = A_2(\mathbf{r})$;
- if $\mu(Y_1, \mathbf{r}) \leq \mu(Y_2, \mathbf{r})$ then return Y_1 else return Y_2.

We can recursively apply this operator to several algorithms and obtain a new one:

$$\mathrm{MIN}_\mu(A_1, \ldots, A_k) := \mathrm{MIN}_\mu(\mathrm{MIN}_\mu(A_1, \ldots, A_{k-1}), A_k).$$

Notice that the ordering among the algorithms specifies how the new algorithm breaks ties. Our main concern is to have a general technique for building truthful mechanisms which optimize $\mu(\cdot)$ and that are *computationally efficient*.

To this end, we will assume that each algorithm A_i satisfies a property (called *plateau-like*) which is slightly stronger than the one (called bitonic) used in [12]:

Definition 1 (plateau-like algorithm). *An algorithm A for a mechanism design graph problem is* monotone *if, for all agents e, and for all $\mathbf{r}_{-e}$ there exists a threshold $\theta_e(\mathbf{r}_{-e}) \in (\mathbb{R}^+ \cup \infty)$ such that (i) every $x \leq \theta_e(\mathbf{r}_{-e})$ is a winning declaration and (ii) every $x > \theta_e(\mathbf{r}_{-e})$ is not a winning declaration. A monotone algorithm A is* plateau-like *w.r.t. $\mu(\cdot)$ if, for all e, for all $\mathbf{r}_{-e}$, the function $g_A(x) := \mu(A(x, \mathbf{r}_{-e}), (x, \mathbf{r}_{-e}))$ is non-decreasing in x and constant for $x > \theta_e(\mathbf{r}_{-e})$.*

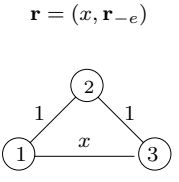
It is well known that an algorithm A can be turned into a truthful mechanism (A, P) if and only if A is monotone [13,2], in which case the payments are uniquely[2] determined by the thresholds:

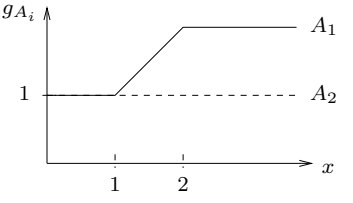
$$P_e(x, \mathbf{r}_{-e}) = \begin{cases} \theta_e(\mathbf{r}_{-e}) & \text{if } A(x, \mathbf{r}_{-e}) \text{ uses } e, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

[2] If $\theta_e(\mathbf{r}_{-e}) = \infty$, then we can set the payment of e to be any constant value and guarantee truthfulness. This case arises if edge e will be always included by A, e.g. for $\mathcal{O}(G-e) = \emptyset$, in which case voluntary participation cannot be guaranteed (unless we assume an upper bound on t_e). Otherwise, i.e. $\theta_e(\mathbf{r}_{-e}) < \infty$, the only payments which guarantee truthfulness are those in (1) [12] which then satisfy the voluntary participation condition.

Mu'Alem and Nisan [12] proved that, if all algorithms A_i are bitonic, then the algorithm $A = \text{MIN}_\mu(A_1, \ldots, A_k)$ is monotone, and thus truthfulness can be guaranteed. Our main contribution here is a method for computing these payments efficiently if we assume that the algorithms are plateau-like. This task is nontrivial since the computation of the thresholds of a 'MIN' combination of algorithms can be rather involved if $\mu(\cdot)$ is not monotone increasing in x as in [12,10]:

Example 1 (Minimum Radius Spanning Tree (MRST)). Consider the problem of computing the *minimum radius spanning tree*, that is, a tree rooted at some node of the graph whose height is minimal. Consider the following simple graph (left):





If A_i outputs a shortest paths tree rooted at node i and $h(\cdot)$ denotes the height of any rooted tree, then both $A := \text{MIN}_h(A_1, A_2)$ and $A' := \text{MIN}_h(A_2, A_1)$ compute a MRST for this graph. However, the thresholds $\theta_e(\mathbf{r}_{-e})$ and $\theta'_e(\mathbf{r}_{-e})$ of the two algorithms are different. This is due to a different tie-breaking rule: For $0 \le x \le 1$, algorithms A_1 and A_2 have the same cost, i.e., $g_{A_1}(x) = g_{A_2}(x)$; hence,

$$A(x, \mathbf{r}_{-e}) = \begin{cases} A_1(x, \mathbf{r}_{-e}) \text{ if } x \le 1 \\ A_2(x, \mathbf{r}_{-e}) \text{ otherwise} \end{cases}$$

while $A'(x, \mathbf{r}_{-e}) = A_2(x, \mathbf{r}_{-e})$. Since A_2 never uses edge e, while A_1 uses this edge for $x \le 2$, it turns out that $\theta_e(\mathbf{r}_{-e}) = 1$ and $\theta'_e(\mathbf{r}_{-e}) = 0$.

Observe that, the threshold of algorithm $\text{MIN}_h(A_1, A_2)$ is *different from the thresholds of the two algorithms*. Its computation depends on the way the functions g_{A_i} cross with each other, which in general can be quite involved (we have to consider how n "stairway" functions intersect pairwise [19] and the order in which we break ties). Finally, a binary search of this threshold may require a time which *depends on the edge weights* (namely, the logarithm of the largest reported type) and thus *not* strongly polynomial time, i.e., not polynomial in the number of nodes and edges. □

We reduce the computation of the payment $P_e(x, \mathbf{r}_{-e})$ to the task of computing, for every algorithm A_i, three thresholds $\theta^i = \theta^i_e(\mathbf{r}_{-e})$, $\hat{\theta}^i = \hat{\theta}^i_e(\mathbf{r}_{-e})$ and $\breve{\theta}^i = \breve{\theta}^i_e(\mathbf{r}_{-e})$. The value θ^i is the threshold in Def. 1 relative to algorithm A_i. The other two thresholds are defined as follows. Since A_i is plateau-like, $g_{A_i}(x) = \mu(A_i(x, \mathbf{r}_{-e}), (x, \mathbf{r}_{-e}))$ is constant for all $x > \theta^i$, where it also reaches its maximum. Let $\overline{g}_i$ be this maximum and let $g_{\min} := \min_i\{\overline{g}_i\}$. We let $\inf\{\emptyset\} = \infty$ and define $\hat{\theta}^i, \breve{\theta}^i \in (\mathbb{R}^+ \cup \infty)$ as follows:

$$\hat{\theta}^i := \inf\{x |\ g_{A_i}(x) \ge g_{\min}\}; \tag{2}$$

$$\breve{\theta}^i := \inf\{x |\ g_{A_i}(x) > g_{\min}\}. \tag{3}$$

Notice that the maximum $\overline{g}_i$ can be easily computed knowing θ_i. This is the main "additional" feature of plateau-like algorithms over bitonic ones. Intuitively speaking, $g_{\min}$ is the minimum cost if we do *not* use edge e. Thus, the solution of algorithm A_i will be selected only if its cost is better/not worse than this value (depending on the used tie-breaking rule). The two thresholds in (2-3) say what is the largest x for which this happens.

Our general approach for constructing computational efficient mechanisms consists in rewriting algorithms as suggested by the following:

Definition 2 (MIN-reducible algorithm). *An algorithm A is* MIN-reducible *if it can be written as the 'MIN' of plateau-like algorithms. That is, there exist k algorithms $A_1, \ldots, A_k$ such that $A = \mathrm{MIN}_\mu(A_1, \ldots, A_k)$ and each algorithm A_i is plateau-like w.r.t. $\mu(\cdot)$. Such an algorithm A is* MIN-reducible in τ time *if, for every input $\mathbf{r}$, it is possible to compute all thresholds $\theta_e^i(\mathbf{r}_{-e})$, $\hat{\theta}_e^i(\mathbf{r}_{-e})$ and $\breve{\theta}_e^i(\mathbf{r}_{-e})$ in at most τ time steps, for all $1 \leq i \leq k$ and for all edges e used by $A(\mathbf{r})$.*

The following result provides a powerful tool for designing efficient truthful mechanisms:

Theorem 1. *If algorithm A is MIN-reducible in $O(\tau)$ time, then there exist payment functions P such that (A, P) is a truthful mechanism and all payments $P_e(x, \mathbf{r}_{-e})$ can be computed in $O(\tau + k(\tau_\mu + N))$ time, where τ_μ is the time to compute $\mu(\cdot)$ and N is the number of used agents/edges.*

Proof Sketch. The first part of the theorem follows from a result by Mu'Alem and Nisan [12]. In order to prove the second part, we simply show that, given the values θ^i, $\hat{\theta}^i$ and $\breve{\theta}^i$, it is possible to compute $\theta_e(\mathbf{r}_{-e})$ in $O(k)$ time after the following preprocessing requiring $O(k \cdot \tau_\mu)$ time. First of all, we compute the index $i_{\min}$ of the first algorithm A_i such that $\overline{g}_i = g_{\min}$. This requires $O(k \cdot \tau_\mu)$ time for computing all $\overline{g}_i$, and from that the computation of $g_{\min}$ and $i_{\min}$ requires $O(k)$ time. (Recall that $\overline{g}_i = g_{A_i}(x)$ for any $x > \theta^i$.) Then we prove the following identity (see the full version [17] for the proof):

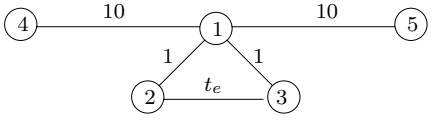
$$\theta_e(\mathbf{r}_{-e}) = \max\{\theta^{i_{\min}}, \max\{\hat{\theta}^i |\ i > i_{\min}\}, \max\{\breve{\theta}^i |\ i < i_{\min}\}\}. \tag{4}$$

Obviously, if we know $\hat{\theta}^i$, $\breve{\theta}^i$ and $i_{\min}$, then the above equality says that a single $\theta_e(\mathbf{r}_{-e})$ can be computed in time linear in k. From (1) we need to compute the payments only for the N edges used in $A(\mathbf{r})$. In this way, by Definition 2, the overall computation of all such $P_e(x, \mathbf{r}_{-e})$ takes $O(\tau + k \cdot \tau_\mu + k \cdot N)$ time.

3 The Minimum Diameter Spanning Tree Problem

In the *minimum diameter spanning tree* (MDST) problem we are given a weighted undirected graph and the goal is to find a spanning tree which minimizes the longest path between any two nodes in that tree (the length of a path is the sum of the weights of its edges). In this section we study the corresponding

mechanism design graph problem. Formally, given a graph $G = (V, E)$, the set $\mathcal{O}(G)$ of feasible solutions consists of all spanning trees; the set of used edges naturally consists of all edges in the tree, and the goal is to find a tree T of minimum *diameter*, that is, a tree such that the length of a maximum-length simple path in T is minimum. We denote this value by $d(G, \mathbf{t})$. Consider the following graph:



For all $t_e \leq 9$, *any* spanning tree is a MDST since, according to the edge weights $\mathbf{t}$ in the picture, the maximum-length simple path is the upper one and this path appears in any spanning tree. Unfortunately, the fact that an algorithm is exact for the MDST problem is not sufficient for obtaining a truthful mechanism. A well-known result by Myerson [13] (see also Archer and Tardos [2]) states that, for our problem, truthfulness can be achieved only if the algorithm is monotone (see Def. 1). It is possible to show that exact algorithms need not lead to truthful mechanisms (see [17] for the details).

In the sequel we will show that there exists an efficient polynomial-time algorithm for the MDST problem which is monotone and such that the payments can be computed efficiently. Both results follow from our main technique (Theorem 1).

3.1 A MIN-Reducible Algorithm for the MDST Problem

The computation of a MDST of a given graph can be reduced to the computation of a shortest paths tree rooted at the *absolute 1-center* (simply *center*, in the following) of G [8]. Loosely speaking, the center of a graph is a point c located on an edge (or on one of its endpoints) such that the distance from c to the farthest node is minimized. In particular, all edges are rectifiable, meaning that any point c on edge $f = (u, v)$ can be specified as a pair $c = (f, \lambda)$ with $\lambda \in [0, 1]$; in this case, we obtain a new graph G_c where edge (u, v) is replaced by two edges (u, c) and (v, c); their weights are $\overline{uc} := \lambda t_e$ and $\overline{vc} := (1 - \lambda)t_e$, respectively. (Notice that we consider each edge as an ordered pair of vertices.) Given a point c on f, one can build a spanning tree T_c of G by computing a shortest paths tree of G_c rooted at c, and then by replacing edges incident to c with the edge (u, v). Trivially, the tree T_c has diameter at most $2h_f^\lambda(\mathbf{t})$.[3] We let $h_f^*(\mathbf{t})$ be the minimum height among all shortest paths trees rooted at some point on f, that is, $h_f^*(\mathbf{t}) := \min_{\lambda \in [0,1]} h_f^\lambda(\mathbf{t})$. Our building block is the following algorithm which computes the *relative center* of edge f for the reported input $\mathbf{r}$, namely, a point $c = (f, \lambda)$ minimizing $h_f^\lambda(\mathbf{r})$:

[3] Formally, the tree T_c is obtained by removing c from the shortest paths tree and by adding back edge (u, v), unless c is sitting on one of the endpoints of (u, v) and is not connected to the other endpoint.

Algorithm CENTER$_f$

- compute the minimum $\lambda \in [0,1]$ such that $h_f^\lambda(\mathbf{r}) = h_f^*(\mathbf{r})$;
- compute the tree T_c for $c = (f, \lambda)$ and edge weights $\mathbf{r}$;
- return $Y = (T_c, c)$. /* return the tree T_c associated with the SPT and the center */

Since it holds that $d(G,\mathbf{r})/2 = h^*(\mathbf{r}) := \min_{f\in E} h_f^*(\mathbf{r})$ [8], we can compute a MDST by searching through all relative centers of G for a best possible position of the center:

$$A_{\mathrm{MDST}} := \mathrm{MIN}_h(\mathrm{CENTER}_{e_1}, \ldots, \mathrm{CENTER}_{e_m}),$$

where $e_1, \ldots, e_m$ denote the edges of G in some arbitrary order (independent of the agents' bids). We stress that a MDST cannot be obtained by restricting the computation of the relative center to one of the endpoints of edge f, that is, by considering only the vertices as possible center locations. This will produce a minimum radius spanning tree, instead, and thus the mechanism in [19] cannot be used here.

The following result, combined with Theorem 1, implies the existence of a truthful mechanism for the MDST (see the full version [17] for the proof).

Theorem 2. *Algorithm A_{MDST} is MIN-reducible and, on input a graph G with edge weights $\mathbf{r}$, it returns a MDST and an absolute center for this input. This computation requires $O(mn\,\alpha(n,n))$ time.*

We need one more step to guarantee that payments can be computed in strongly polynomial time. One of our major technical contributions is to show that the "MIN-reduction" can be done efficiently:

Theorem 3. *Algorithm A_{MDST} is MIN-reducible in $O(mn^2\,\alpha(n,n))$ time.*

Efficient Computations via Upper/Lower Envelopes (Proof Idea of Theorem 3). Let $\delta_{u,v}(G,\mathbf{r})$ be the (shortest path) distance from node u to node v in a graph G with weights $\mathbf{r}$. We compute the distances $\delta_{u,v}(G,\mathbf{r})$ and $\delta_{u,v}(G-e,\mathbf{r}_{-e})$, for all nodes u and v, and for all edges e used by the computed solution. Using the $O(mn\log\alpha(m,n))$-time all-pairs shortest paths algorithm by Pettie and Ramachandran [18], this step takes $O(mn^2\log\alpha(m,n))$ time. (We have n graphs in total since the computed solution uses $n-1$ edges.) This term is dominated by $O(mn^2\alpha(n,n))$.

In the remaining of this section, we fix an edge e, and $\mathbf{r}_{-e}$, and an algorithm $A_i = \mathrm{CENTER}_f$, and we show how to compute the thresholds $\theta_e^i(\mathbf{r}_{-e})$, $\hat{\theta}_e^i(\mathbf{r}_{-e})$ and $\breve{\theta}_e^i(\mathbf{r}_{-e})$ in $O(n\,\alpha(n,n))$ time. This implies Theorem 3 since there are m algorithms and $n-1$ agents/edges e used by the computed solution.

At the heart of the proof is an efficient method for computing, for any edge $f=(u,v)$, the following function in $O(n\,\alpha(n,n))$ time:

$$\hat{F}(\ell) := \inf\{x|\ h_f^*(x,\mathbf{r}_{-e}) \geq \ell\}. \tag{5}$$

This value can be computed by considering the *lower envelope* of n functions $\hat{f}_z(\ell, \lambda)$, one for each node z, defined as follows. The value $\hat{f}_z(\ell, \lambda)$ is the infimum value x for r_e such that the distance from a fixed center $c = (f, \lambda)$ to node z is at least ℓ. (Recall that x is the weight of edge e.) Such distance is the minimum of the following two functions, one for the path through u and one for the path through v:

$$upath_\lambda(x) := \lambda r_f + \min\{x + \delta_{u,z}(G, (0, \mathbf{r}_{-e})), \delta_{u,z}(G - e, \mathbf{r}_{-e})\}; \quad (6)$$

$$vpath_\lambda(x) := (1 - \lambda) r_f + \min\{x + \delta_{v,z}(G, (0, \mathbf{r}_{-e})), \delta_{v,z}(G - e, \mathbf{r}_{-e})\}. \quad (7)$$

These two functions are of the form $\lambda r_f + \min\{x + a, b\}$ and $(1 - \lambda) r_f + \min\{x + a', b'\}$, respectively (see Fig. 1(left)).

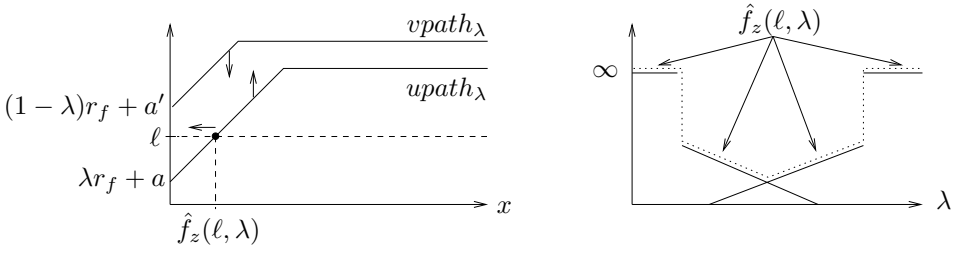


Fig. 1. From shortest paths distances to upper envelopes

We let $\hat{f}_z(\ell, \lambda) = \infty$ (respectively $\hat{f}_z(\ell, \lambda) = 0$) if, for all x, one function is below ℓ (respectively, both functions are not below ℓ). Otherwise, $\hat{f}_z(\ell, \lambda)$ is the x coordinate of the point where the *lowest* (i.e., smaller for $x = 0$) of the functions in (6-7) intersects with the limit ℓ. Notice that, when increasing λ by one unit, the two functions (6-7) move by r_f units as shown in Fig. 1(left). Hence, the point moves accordingly and thus the function $\hat{f}_z(\ell, \lambda)$ can be fully specified by two slanted segments as in Fig. 1(right). Each slanted segment is obtained by considering the intersection of each function in Fig. 1(left) with the limit ℓ. Hence, $\hat{f}_z(\ell, \lambda)$ is the dotted curve in Fig. 1(right) which is given by the upper envelope of the two solid curves in Fig. 1(right) (see [17] for the details).

In order to compute $\hat{F}(\ell)$, we consider $\hat{f}(\ell, \lambda) := \min_z\{\hat{f}_z(\ell, \lambda)\}$ and observe that $\hat{F}(\ell) = \sup_{\lambda \in [0,1]} \hat{f}(\ell, \lambda)$. The actual computation of $\hat{F}(\ell)$ consists in determining the *lower envelope* of all functions $\hat{f}_z(\ell, \cdot)$ and then finding its maximum for $\lambda \in [0, 1]$. Since the functions $\hat{f}_z(\ell, \cdot)$ intersect pairwise in at most one point, this requires $O(n\,\alpha(n, n))$ time using Agarwal and Sharir [1] approach, once the segments of each function have been computed. The latter can be obtained from the pre-computed distances using (6-7). Moreover, these distances allow us to compute the solution of algorithm $A_i = \text{CENTER}_f$ and the values $\overline{g}_i$ and $g_{\min}$, still in $O(n\,\alpha(n, n))$ time. From the first step of CENTER_f and from (2) we obtain the following two identities, respectively: (i) $\theta^i_e(\mathbf{r}_{-e}) = \inf\{x|\ h^*_f(x, \mathbf{r}_{-e}) \geq \overline{g}_i\} = \hat{F}(\overline{g}_i)$; (ii) $\hat{\theta}^i_e(\mathbf{r}_{-e}) = \inf\{x|\ h^*_f(x, \mathbf{r}_{-e}) \geq \overline{g}_{\min}\} = \hat{F}(\overline{g}_{\min})$. Since the

threshold $\breve{\theta}_e^i(\mathbf{r}_{-e})$ can be computed with a very similar approach, each of the $O(mn)$ thresholds can be computed in $O(n\,\alpha(n,n))$ time (see [17] for the details). Hence, Theorem 3 follows.

From Theorems 1, 2, and 3 we obtain the following:

Corollary 1. *There exists an $O(mn^2\,\alpha(n,n))$-time truthful mechanism for the MDST problem.*

4 Conclusions

We have described a general approach for building truthful mechanisms running in *strongly polynomial time* based on the 'MIN' operator defined by Mu'Alem and Nisan [12]. This is similar to what Kao *et al.* [10] propose, though their method for computing the payments assumes that each function $g_{A_i}(x)$ is *monotonically increasing* in $x < \theta_e(\mathbf{r}_{-e})$ (see the assumptions preceding Theorem 10 in [10]). This is too restrictive as the optimization functions used in the MRST and MDST do not fulfill this requirement and payments obtained from [10] do *not* guarantee truthfulness in these cases (in Example 1, their approach would ignore the tie-breaking rule among the algorithms).

Our technique has a very natural application to the MDST problem where the underlying algorithm in [8] optimizing the diameter $d(\cdot)$ can be rewritten as a 'MIN' combination of m algorithms optimizing a *different* function $h(\cdot)$, i.e., the height of a SPT rooted at the relative center of an edge. Although the results have been presented for mechanism design graph problems, they apply to a more general framework in which the agent valuations are either 0 or t_e, that is, to the *known single minded bidders* in [12] or, equivalently, to the *binary demand games* in [10]. The fact that we require plateau-like algorithms (instead of bitonic ones in [12]) does not directly prevent from optimal solutions (any bitonic algorithm minimizing the function $\mu(\cdot)$ is automatically plateau-like). Voluntary participation is guaranteed if optimal algorithms must drop an agent when its cost becomes too high. We can also obtain a strongly polynomial time truthful mechanism for the *p-center* graph problem [20] (in addition to the location of the p centers, we want to compute the associated trees), for any constant p. Notice that the problem is NP-hard for arbitrary p [20]. For the MRST, our method yields a mechanism which improves slightly the running time in [19]. (Details on both these problems are given in the full version [17].)

An interesting future direction is to apply our technique to NP-hard problems to obtain truthful approximation mechanisms (this was done in [10] for problems maximizing the *welfare*, i.e., the sum of all agents costs which obviously meet the "monotone increasing" requirement). According to Theorem 1, it suffices to show that an approximation algorithm is MIN-reducible in polynomial time. An interesting question here is whether the approximation ratio of the "best" approximation polynomial-time algorithm can be attained by some truthful polynomial-time mechanism.

Notice that our positive results cannot be extended to the case in which an agent owns several edges of a graph (these problems can model certain scheduling

problems for which no exact truthful mechanism exists [15,2], while an extension of Theorem 1 would imply such an exact mechanism).

References

1. P.K. Agarwal and M. Sharir. Davenport-Schinzel sequences and their geometric applications. *Cambridge University Press, New York*, 1995.
2. A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. of 42nd FOCS*, pages 482–491, 2001.
3. E.H. Clarke. Multipart Pricing of Public Goods. *Public Choice*, pages 17–33, 1971.
4. J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proceedings of the 6th DIALM*, pages 1–13. ACM Press, 2002.
5. T. Groves. Incentive in Teams. *Econometrica*, 41:617–631, 1973.
6. L. Gualà and G. Proietti. A truthful (2-2/k)-approximation mechanism for the Steiner tree problem with k terminals. In *Proc. of the 11th COCOON*, volume 3595 of *LNCS*, pages 90–400, 2005.
7. L. Gualà and G. Proietti. Efficient truthful mechanisms for the single-source shortest paths tree problem. In *Proc. of the 11th EURO-PAR*, volume 3648 of *LNCS*, pages 941–951, 2005.
8. R. Hassin and A. Tamir. On the minimum diameter spanning tree problem. *Info. Proc. Lett.*, 53(2):109–111, 1995.
9. J. Hershberger and S. Suri. Vickrey prices and shortest paths: what is an edge worth? In *Proc. of the 42nd FOCS*, pages 252–259, 2001.
10. M.-Y. Kao, X.-Y. Li, and W. Wang. Towards truthful mechanisms for binary demand games: A general framework. In *Proc. of ACM EC*, pages 213–222, 2005.
11. E. Koutsoupias and C.H. Papadimitriou. Worst-case equilibria. In *Proc. of STACS*, volume 1563 of *LNCS*, pages 404–413, 1999.
12. A. Mu'Alem and N. Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *Proc. of 18th AAAI*, pages 379–384, 2002.
13. R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6:58–73, 1981.
14. E. Nardelli, G. Proietti, and P. Widmayer. Finding the most vital node of a shortest path. *Theoretical Computer Science*, 296(1):167–177, 2003.
15. N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proc. of the 31st STOC*, pages 129–140, 1999.
16. C.H. Papadimitriou. Algorithms, games, and the Internet. In *Proc. of the 33rd STOC*, pages 749–753, 2001.
17. P. Penna, G. Proietti, and P. Widmayer. Strongly polynomial-time truthful mechanisms in one shot. Technical report, Università di Salerno, Available at http://ec.tcfs.it/Group/Selfish_Agents.html, 2006.
18. S. Pettie and V. Ramachandran. Computing shortest paths with comparisons and additions. In *Proc. of the 13th SODA*, pages 267–276, 2002.
19. G. Proietti and P. Widmayer. A truthful mechanism for the non-utilitarian minimum radius spanning tree problem. In *Proc. of SPAA*, pages 195–202, 2005.
20. B.C. Tansel, R.L. Francis, and T.J. Lowe. Location on networks: a survey. Part I: The p-center and p-median problems. *Management Sciences*, 29:482–497, 1983.
21. W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.