Dynamic Mechanism Design*

Davide Bilò[1], Luciano Gualà[1], and Guido Proietti[1,2]

[1] Dipartimento di Informatica, Università di L'Aquila, Italy
[2] Istituto di Analisi dei Sistemi ed Informatica, CNR, Roma, Italy
{davide.bilo, guala, proietti}@di.univaq.it

Abstract. In this paper we address the question of designing truthful mechanisms for solving optimization problems on dynamic graphs. More precisely, we are given a graph G of n nodes, and we assume that each edge of G is owned by a selfish agent. The strategy of an agent consists in revealing to the system the cost for using its edge, but this cost is not constant and can change over time. Additionally, edges can enter into and exit from G. Among the various possible assumptions which can be made to model how these edge-cost modifications take place, we focus on two settings: (i) the *dynamic*, in which modifications are unpredictable and time-independent, and for a given optimization problem on G, the mechanism has to maintain efficiently the output specification and the payment scheme for the agents; (ii) the *time-sequenced*, in which modifications happens at fixed time steps, and the mechanism has to minimize an objective function which takes into consideration both the quality and the set-up cost of a new solution. In both settings, we investigate the existence of exact and approximate truthful mechanisms. In particular, for the dynamic setting, we analyze the *minimum spanning tree* problem, and we show that if edge costs can only decrease, then there exists an efficient dynamic truthful mechanism for handling a sequence of k edge-cost reductions having runtime $\mathcal{O}(h \log n + k \log^4 n)$, where h is the overall number of payment changes.

Keywords: Algorithmic Mechanism Design, On-line Problems, Dynamic Algorithms, Approximate Mechanisms.

1 Introduction

Algorithmic mechanism design (AMD) is concerned with the computational complexity of implementing, in a centralized fashion, truthful mechanisms for solving optimization problems in multi-agents systems [13]. AMD is by now one of the hottest topic in theoretical computer science, especially since of the game-theoretic nature of Internet. As a result, many classic network optimization problems have been resettled and solved under this new perspective [3,4,7,8,9].

* Work partially supported by the Research Project GRID.IT, funded by the Italian Ministry of Education, University and Research.

P. Spirakis et al. (Eds.): WINE 2006, LNCS 4286, pp. 3–15, 2006.

Apparently, however, the canonical approach is that of dealing with these problems by means of one-shot mechanisms, whose natural computational counterpart are static graph problems. This is in contrast with the intrinsic dynamicity of Internet's infrastructure, where links and node can rapidly appear, disappear, or even change their characteristics. Thus, surprisingly enough, there is a lack of modeling for those situations in which agents need to adapt their strategies over time, according to sudden changes in their owned components. To the best of our knowledge, the only effort towards this direction has been done in the framework of the so-called *on-line mechanism design* (OMD) [6,14]. There, the dynamic aspect resides in the fact that agents arrive and depart once over time, and their strategy consist of a *single* announcement of a bidding value for a time interval included between the arrival and the departing time. However, the limitation of OMD is that agents are not allowed to play different strategies over time, thus preventing to model those situations in which bidding values need to be continuously adjusted.

In this paper, we aim exactly to fill this gap, by exploring the difficulties and the potentialities emerging in this new challenging scenario. In doing that, we combine some of the theoretical achievements of the AMD with techniques which are proper of dynamic and on-line algorithms. The result of this activity is what we call as *dynamic mechanism design* (DMD). As a paradigmatic framework, we consider the situation in which each agent owns an edge of a given underlying graph G of n nodes, and its strategy consists in revealing to the system the cost (which can change over time) for using its edge. We focus on two main realistic scenarios:

1. In the first scenario, we consider the case in which edge costs are subject to sudden changes, due to boundary conditions alterations. In the extreme case, an edge might become unavailable to the system, due to a failure for instance. On the opposite side, some new edge might become available. All these variations are presented *on-line* to the system, which is completely unaware of possible future changes. Moreover, we will assume that each agent is unaware about other agents' types and strategies, and thus it cannot observe the global status of the system.[1] We feel that this is particularly attractive in an Internet setting, where an agent may not even know which other agents are participating to the mechanism. From an algorithmic point of view, this translates into a continuously evolving input graph, over which a solution to a given optimization problem has to be maintained. In other words, we need to design a *fully dynamic* mechanism which updates efficiently both the output specification and the corresponding payment scheme for the agents. In the rest of the paper, we will refer to this as the *dynamic* scenario. What is interesting here is that while classic dynamic graph algorithms can be used for the maintenance of the output specification, as far as the payment scheme updating is concerned, this defines

[1] Notice that the case in which an agent can observe the strategies of the other agents transforms our problem into a *repeated game*, for which the existence of a dominating strategy is unknown.

novel dynamic graph problems, which would make no sense in a canonical centralized framework. In this paper, as a starting point, we deal with a basic graph problem that has served as a case study for several papers on AMD, namely the *minimum spanning tree* (MST) problem. After observing that efficient dynamic MST algorithms in [10] can be turned into an efficient dynamic mechanism for handling a sequence of k edge-cost modifications having runtime $\mathcal{O}(k\,n\log^4 n)$, we will show that for the case in which edges can only become less expensive, then the mechanism runtime drops to $\mathcal{O}(h\log n + k\log^4 n)$, where h is the overall number of payment changes. We emphasize that this edge-cost lowering scenario is interesting because of the competitive nature of Internet.

2. In the second scenario, we consider the case in which the graph evolves in a sequence of time steps, and every agent has a specific cost for using its edge in each of these steps. Here, the time-depending modifications of the graph suggest that the mechanism's goal should now be the composition of two objectives: maintaining a *good* (not necessarily optimal) solution at a *low* (not necessarily minimal) cost of setting it up. Thus, on a sequence of graph changes, the objective function is now given by the overall cost of the sequence of solutions, plus the overall set-up cost. This approach is inspired to that proposed in the past in [11] to model the fact that on an on-line sequence of changes, it is important to take care of the modifications on the structure of the solution, since radical alterations might be too onerous in terms of set-up costs. In the rest of the paper, we will refer to this as the *time-sequenced* scenario. Here, on a positive side, we will show that: (i) if each set-up cost is upper bounded by the initial one and changes are presented *on-line* to the system, then a ρ-approximate monotone algorithm for a given optimization problem Π on G, translates into an approximate truthful mechanism for Π which on a sequence of graph changes of size k has an approximation ratio of $\max\{k, \rho\}$; (ii) if the underlying graph optimization problem is utilitarian and polynomial-time solvable, and changes are presented *off-line* to the system, then there exists a VCG-like truthful mechanism for solving optimally the sequence, which can be computed in polynomial time by means of a dynamic programming technique. On the other hand, on a negative side, we will show that even if graph changes are presented off-line to the system and set-up costs are uniform, then any truthful mechanism which solves the problem by means of a *divide et impera* paradigm (as explained in more detail in Section 6) cannot achieve a better than k approximation ratio.

The paper is organized as follows: in Section 2 we give preliminary definitions; after, in Section 3 we present the dynamic mechanism for the MST problem, while in Section 4 we define formally the time-sequenced model; finally, in the last two sections we give, respectively, positive and negative results on the existence of time-sequenced truthful mechanisms.

2 Preliminaries

Let a communication network be modeled by a graph $G = (V, E)$ with n nodes and m edges. We will deal with the case in which each edge $e \in E$ is controlled by a selfish agent a_e holding a private information t_e, namely the *true type* of a_e. Only agent a_e knows t_e. Each agent has to declare a public *bid* b_e to the mechanism. We will denote by t the vector of types, and by b the vector of bids.

For a given optimization problem Π defined on G, let $\textsc{Sol}(\Pi)$ denote the corresponding set of feasible solutions. We will assume that $\textsc{Sol}(\Pi)$ does not depend on the agents' types. For each $x \in \textsc{Sol}(\Pi)$, an objective function is defined, which depends on the agents' types. A *mechanism* for Π is a pair $\mathcal{M} = \langle g(b), p(b) \rangle$, where $g(b)$ is an algorithm that, given the agents' bids, computes a solution for Π, and $p(b)$ is a scheme which describes the payments provided to the agents. For each solution x, a_e incurs a cost $\nu_e(t_e, x)$ for participating to x (also called *valuation of* a_e *w.r.t.* x). Each agent tries to maximize its *utility*, which is defined as the difference between the payment provided by the mechanism and the cost incurred by the agent w.r.t. the computed solution. On the other hand, the mechanism aims to compute a solution which minimizes the objective function of Π w.r.t. to the agents' types, but of course it does not know t directly. In a *truthful* mechanism this tension between the agents and the system is resolved, since each agent maximizes its utility when it declares its type, regardless of what the other agents do.

Given a positive real function $\varepsilon(n)$ of the input size n, an $\varepsilon(n)$-*approximate* mechanism returns a solution whose measure comes within a factor $\varepsilon(n)$ from the optimum. A mechanism has a runtime of $\mathcal{O}(f(n))$ if $g(\cdot)$ and $p(\cdot)$ are computable in $\mathcal{O}(f(n))$ time. Moreover, a mechanism design problem is called *utilitarian* if the objective function of Π is equal to $\sum_{e \in E} \nu(t_e, x)$. For utilitarian problems, there exists a well-known class of truthful mechanisms, i.e., the *Vickrey-Clarke-Groves (VCG) mechanisms*.

In [2], Archer and Tardos have shown how to design truthful mechanisms for another well-known class of mechanism design problems called *one-parameter*. A problem is said one-parameter if (i) the true type of each agent a_e can be expressed as a single parameter $t_e \in \mathbb{R}$, and (ii) each agent's valuation has the form $\nu_e(t_e, x) = t_e\, \omega_e(b)$, where $\omega_e(b)$ is called the *work curve* for agent a_e, i.e., the amount of work for a_e depending on the output specified by the mechanism algorithm, which in its turn is a function of the bid vector b. When, for each agent a_e, $\omega_e(b)$ can be either 0 or 1, then the problem is also called *binary demand* [12]. In [2] it is shown that for one-parameter problems, a sufficient condition for truthfulness is given by a monotonicity property of the mechanism algorithm. In particular, for a binary demand problem, such property reduces to the following. Let b be the vector of bids of the agents, and let b_{-e} denote the vector of all bids besides b_e; the pair (b_{-e}, b_e) will denote the vector b. If we fix b_{-e}, a *monotone* algorithm $\mathcal{A}$ defines a threshold value $\theta_e(b_{-e}, \mathcal{A})$ such that if a_e bids no more than $\theta_e(b_{-e}, \mathcal{A})$, then e will be selected, while if a_e bids

above $\theta_e(b_{-e}, \mathcal{A})$, e will not be selected.[2] Sometimes, we will write $\theta_e(b_{-e})$ when the algorithm $\mathcal{A}$ is clear from the context. The results in [2] imply that the only truthful mechanism for a binary demand problem using an algorithm $\mathcal{A}$ is the one-parameter mechanism $\mathcal{M} = \langle \mathcal{A}, p^{\mathcal{A}}(\cdot) \rangle$, where $\mathcal{A}$ is required to be monotone, and the payment $p_e^{\mathcal{A}}(b)$ for each agent a_e is defined as its threshold value if it owns a selected edge, and 0 otherwise.

3 An Efficient Dynamic Mechanism for the MST Problem

We start by addressing the problem of designing an efficient mechanism for the fully dynamic MST problem. Since we assume that agents' types change over time, we allow the agents to declare a new bid to the mechanism at any time. Recall that edge-cost changes are presented on-line to the system, which is unaware of possible future changes, and that the agents do not know other agents' bids. The mechanism works as follows. At any time, whenever it receives a new bid from an agent, it computes a new MST w.r.t. the new bid profile, and it updates the payments exactly as the one-parameter mechanism for the MST problem. Concerning the truthfulness of the mechanism, this follows from the truthfulness of the one-parameter mechanism for the MST problem, and from the fact that every agent is completely unaware of other agents' bids.

On the other hand, concerning the time complexity, the mechanism has to maintain: (i) an MST of G, and (ii) the corresponding payments. Moreover, it has to support a payment query in $\mathcal{O}(1)$ time. To dynamically maintain an MST, one can use the algorithm proposed in [10], which takes $\mathcal{O}(k \log^4 n)$ time for processing k edge-cost updates (deletions of edges are simulated by setting to $+\infty$ the cost of an edge). Thus, it remains to manage the payment scheme. We remind that for binary demand problems, the payment provided to a_e is equal to $\theta_e(b_{-e})$ if e is selected, and zero otherwise. This means it suffices to maintain the threshold value $\theta_e(b_{-e})$ for each e belonging to the current solution. We emphasize that the algorithm in [10] can be straightforwardly used to accomplish such a task, and from this it follows that there exists a truthful mechanism for the fully dynamic MST which runs in $\mathcal{O}(k\, n \log^4 n)$ time for processing k updates. Improving this latter result is a challenging open problem. In the following, we show that for the *edge-cost decreasing* case, in which edge costs are only allowed to decrease, a significant improvement is possible. We argue this is not a very special case, as it includes the well-known *partially dynamic* scenario, where only edge insertions are allowed.

How to Maintain the Payments. Let G be a graph, and let T be an MST of G. For each *non-tree edge* $f = (u, v) \in E \setminus E(T)$, $T(f)$ will denote the set of tree edges belonging to the (unique) path in T between u and v. For each $e \in E(T)$,

[2] As usual, we will assume that there always exists a feasible solution not containing e, which implies that $\theta_e(b_{-e}, \mathcal{A})$ is bounded.

let $C_T(e) = \{f \in E \setminus E(T) \mid e \in T(f)\}$. We denote by $swap(e)$ the cheapest non-tree edge in $C_T(e)$.[3] Note that $\theta_e(b_{-e}) = b_{swap(e)}$.

Clearly, if a tree edge decreases its cost, no payment changes. Consider now the situation in which a non-tree edge f decreases its cost from b_f to b'_f. Denote by T' the new MST, i.e., the MST computed w.r.t. the cost profile $b' = (b_{-f}, b'_f)$. We have two cases:

Case 1: $T' = T$. Clearly, only the threshold of edges in $T(f)$ may change, since for each $e' \notin T(f)$, no edge in $C_T(e')$ has changed its cost. Moreover, the threshold of e changes iff $\theta_e(b_{-e}) > b'_f$, and in this case the new threshold value becomes $\theta_e(b'_{-e}) = b'_f$.

Case 2: $T' \neq T$. Clearly $T' = T \setminus \{e\} \cup \{f\}$. Moreover, the payment for a_e becomes 0, while that for a_f will be $\theta_f(b'_{-f}) = b_e$, since $C_{T'}(f) \subseteq C_T(e) \cup \{e\}$.

Lemma 1. *For every $e' \in E(T') \setminus T'(e)$, $\theta_{e'}(b'_{-e'}) = \theta_{e'}(b_{-e'})$.*

Proof. The lemma trivially follows from the fact that for each $e' \in E(T') \setminus T'(e)$, $C_{T'}(e') = C_T(e')$ and $f \notin C_T(e')$. □

Lemma 2. *The threshold of an edge $e' \in T'(e)$ changes iff $\theta_{e'}(b_{-e'}) > b_e$. In this case, $\theta_{e'}(b'_{-e'}) = b_e$.*

Proof. Let $e' \in T'(e)$ be such that $\theta_{e'}(b_{-e'}) > b_e$. Since $e \in C_{T'}(e')$, then $\theta_{e'}(b'_{-e'}) \leq b_e$. We have to show that $\nexists f' \in C_{T'}(e')$ with $b_{f'} < b_e$. For the sake of contradiction, suppose that $\exists f' \in C_{T'}(e')$ such that $b_{f'} < b_e$. Then, we show T was not an MST by proving that $f' \in C_T(e)$. Suppose that $f' \notin C_T(e)$; then $T(f') = T'(f')$, which implies $\theta_{e'}(b_{-e'}) < b_e$.

It remains to show that if $\theta_{e'}(b_{-e'}) \leq b_e$, then $\theta_{e'}(b'_{-e'}) = \theta_{e'}(b_{-e'})$. Notice that if $swap(e') \in C_T(e)$, then $\theta_{e'}(b_{-e'}) \geq b_e$ from the minimality of T, which implies $\theta_{e'}(b_{-e'}) = b_e$. Otherwise, $swap(e') \in C_{T'}(e')$. In both cases $\theta_{e'}(b'_{-e'}) \leq \theta_{e'}(b_{-e'})$. Moreover, since $C_{T'}(e') \subseteq C_T(e') \cup C_T(e) \cup \{e\}$, then

$$\begin{aligned}\theta_{e'}(b'_{-e'}) &= \min_{f \in C_{T'}(e')} \{b_f\} \geq \min_{f \in C_T(e') \cup C_T(e) \cup \{e\}} \{b_f\} \\ &= \min\{b_{swap(e')}, b_e\} = \theta_{e'}(b_{-e'}).\end{aligned}$$

□

Implementation. To update the payments, we use a *top tree*, a data structure introduced by Alstrup *et al.* [1] to maintain information about paths in trees. More precisely, a top tree represents an edge-weighted forest $\mathcal{F}$ with weight function $c(\cdot)$. Some operations defined for top trees are:

- `link`$((u, v), x)$, where u and v are in different trees. It links these trees by adding the edge (u, v) of weight $c(u, v) = x$ to $\mathcal{F}$.
- `cut`(e). It removes the edge e from $\mathcal{F}$.
- `update`(e, x), where e belongs to $\mathcal{F}$. It sets the weight of e to x.
- `max`(u, v), where u and v are connected in $\mathcal{F}$. It returns the edge with maximum weight among the edges on the path between u and v in $\mathcal{F}$.

[3] If there are more than one such cheapest edges, we pick one of them arbitrarily.

In [1,5,15], it is shown how to implement a top tree (by using $\mathcal{O}(n)$ space) for supporting each of the above operations in $\mathcal{O}(\log n)$ time.

To our scopes, we use a top tree $\mathcal{T}$ as follows. $\mathcal{T}$ maintains the current MST where the cost of each edge $e \in E(T)$ is $\theta_e(b_{-e})$. Concerning Case 1, we only need to update the threshold of some edges in $T(f)$. So, let $f = (x, y)$ be the edge which has decreased its cost. While $c(e') > b'_f$, where $e' = \texttt{max}(x, y)$, then we (i) update the payment for $a_{e'}$ to b'_f, and (ii) perform $\texttt{update}(e', b'_f)$. For what concerns Case 2, let $e = (x, y)$ be the edge in T not in T'. First, we update the MST by performing $\texttt{cut}(e)$ and $\texttt{link}(f, b_e)$. Next, we update the payment for a_e (resp., a_f) to 0 (resp., b_e). Finally, while $c(e') > b_e$, where $e' = \texttt{max}(x, y)$, then (i) we update the payment for $a_{e'}$ to b_e, and (ii) we perform $\texttt{update}(e', b_e)$.

The above discussion yields the following:

Theorem 1. *There exists a dynamic mechanism supporting a sequence of k edge-cost decreasing operations in $\mathcal{O}(h \log n + k \log^4 n)$ time, where h is the overall number of payment changes.* □

4 Time-Sequenced Scenario: Problem Statement

Let $G = (V, E)$ be a graph with a positive real weight $w(e)$ associated with each edge $e \in E$. Henceforth, unless stated otherwise, by Π we will denote a *communication network* problem on (G, w), which asks for computing a subgraph $H \in \textsc{Sol}(\Pi)$ of G by minimizing an objective function $\phi(H, w)$ of the form

$$\phi(H, w) = \sum_{e \in E(H)} w(e) \cdot \mu_H(e),$$

where $\mu_H(e)$ depends only on the topology of H. Notice that this definition embraces the quasi-totality of communication network problems, like the MST problem, the *shortest-paths tree* problem, and so on.

Let k be a positive integer. We assume that the type of each agent a_e is $t_e = \langle t_e^1, \ldots, t_e^k \rangle$, while its bid is $b_e = \langle b_e^1, \ldots, b_e^k \rangle$. Intuitively, t_e^i represents the true cost incurred by a_e for using its link e at time i. We will denote by $t^i \in \mathbb{R}^m$ the vector of agents' types at time i, and by t the vector $\langle t^1, \ldots, t^k \rangle$.

Given a communication network problem Π, we want to design a truthful mechanism for the optimization problem that we will denote by $\textsc{Seq}(\Pi)$. This latter problem asks for computing a sequence $\mathcal{H} = \langle H_1, \ldots, H_k \rangle$, where $H_i \in \textsc{Sol}(\Pi)$, $i = 1, \ldots, k$, by minimizing the following objective

$$\Psi(\mathcal{H}, t) = \Phi(\mathcal{H}, t) + \Gamma(\mathcal{H}),$$

where $\Phi(\mathcal{H}, t)$ is a function measuring the quality of the solution $\mathcal{H}$, and $\Gamma(\mathcal{H})$ is a function measuring the overall set-up cost. For a given sequence $\mathcal{H}$, we will naturally assume that the valuation of a_e w.r.t. $\mathcal{H}$ is:

$$\nu_e(\mathcal{H}, t_e) = \sum_{i=1}^{k} \nu_e^i(H_i, t_e^i), \quad \text{where} \quad \nu_e^i(H_i, t_e^i) = \begin{cases} t_e^i & \text{if } e \in E(H_i); \\ 0 & \text{otherwise.} \end{cases}$$

Depending on the cost model to be adopted, the functions $\Phi(\cdot)$ and $\Gamma(\cdot)$ can be defined accordingly. In this paper, we will consider the prominent *additive* cost model, in which

$$\Phi(\mathcal{H},t)=\sum_{i=1}^{k}\phi(H_i,t^i),\qquad \Gamma(\mathcal{H})=\sum_{i=1}^{k}\gamma(i,\mathcal{H}),$$

where

$$\gamma(i,\mathcal{H})=\begin{cases}\gamma_1\in\mathbb{R}^+ & \text{if } i=1;\\ \gamma_i\in\mathbb{R}^+ & \text{if } H_i\neq H_{i-1},\ i=1,\ldots,k;\\ 0 & \text{otherwise.}\end{cases}$$

For any $1\leq i\leq j\leq k$, by $[i,j]$ we will denote the *interval* $\{i,\ldots,j\}$. We will write $[i,j)$ instead of $[i,j-1]$. Given two intervals $[i,j],[i',j']$, we write $[i,j]\prec[i',j']$ if $j<i'$. An *interval vector* $s=\langle I_1,\ldots,I_h\rangle$ is a vector of pairwise disjoint intervals whose union is $\{1,\ldots,k\}$, and such that $I_1\prec\cdots\prec I_h$. Given an interval I, let b^I be the vector defined as $b^I_e=\sum_{i\in I}b^i_e$, for each edge $e\in E$. Moreover, we will denote by H^*_I an optimum solution for Π when the input is (G,b^I). Finally, given two sequences $\mathcal{H}=\langle H_1,\ldots,H_i\rangle,\mathcal{H}'=\langle H'_1,\ldots,H'_j\rangle$, by $\mathcal{H}\odot\mathcal{H}'$ we denote the sequence $\langle H_1,\ldots,H_i,H'_1,\ldots,H'_j\rangle$.

5 Time-Sequenced Mechanisms: Positive Results

In this section we first define the class of *time-sequenced single-parameter (TSSP) mechanisms*, and we prove that any mechanism in this class is truthful for $\textsc{Seq}(\Pi)$. Moreover, for the case in which each set-up cost is upper bounded by γ_1, we show that there exists an on-line $\max\{k,\rho\}$-approximate TSSP mechanism, where ρ is the approximation ratio of a monotone algorithm for Π. Then, we turn our attention to the special case in which Π is utilitarian and polynomial-time solvable, and we show that if the graph changes are presented off-line to the system, then there exists a VCG-like truthful mechanism for solving optimally $\textsc{Seq}(\Pi)$, which can be computed in polynomial time by means of a dynamic programming technique.

5.1 On-Line Sequences with Bounded Set-Up Costs

From now on, by $\tilde{s}$ we will denote the interval vector $\langle[1,1],\ldots,[k,k]\rangle$.

Definition 1. *Given a communication network problem Π, and a monotone algorithm $\mathcal{A}$ for Π, a* TSSP mechanism $\mathcal{M}(s)=\langle g_s(b),p(b)\rangle$ *with interval vector $s=\langle I_1,\ldots,I_h\rangle$ for* $\textsc{Seq}(\Pi)$ *is defined as follows:*

1. *$g_s(\cdot)$ returns a sequence $\mathcal{H}=\mathcal{H}_1\odot\cdots\odot\mathcal{H}_h$, in which*

$$\forall j=1,\ldots,h,\quad \mathcal{H}_j=\langle\hat{H}_j,\ldots,\hat{H}_j\rangle\ \ \textit{has size } |I_j|,$$

where $\hat{H}_j$ is the solution returned by $\mathcal{A}$ with input (G,b^{I_j});

2. For each agent a_e

$$p_e(b) = \sum_{j=1}^{h} p_e^{\mathcal{A}}\big(b^{I_j}\big),$$

where $p_e^{\mathcal{A}}(b^{I_j})$ is the payment provided to a_e by the one-parameter mechanism $\langle \mathcal{A}, p^{\mathcal{A}}(\cdot)\rangle$ for the problem Π when the input is (G, b^{I_j}).

Notice that, by definition, $\mathcal{M}(\tilde{s})$ is the only on-line TSSP mechanism.

Proposition 1. *$\mathcal{M}(s)$ is a truthful mechanism for* $\textsc{Seq}(\Pi)$.

Proof. The mechanism breaks the problem in h instances $(G, b^{I_1}), \ldots, (G, b^{I_h})$ which are independent each other. Then it uses the one-parameter mechanism $\langle \mathcal{A}, p^{\mathcal{A}}(\cdot)\rangle$ for each of them in order to locally guarantee the truthfulness. □

The main result of this section, whose proof is omitted due to lack of space, is the following:

Theorem 2. *Given a ρ-approximate monotone algorithm $\mathcal{A}$ for Π, the mechanism $\mathcal{M}(\tilde{s})$ applied to* $\textsc{Seq}(\Pi)$ *with the assumption that each set-up cost is upper bounded by γ_1, has a performance guarantee of* $\max\{k, \rho\}$. □

5.2 Off-Line Utilitarian Problems

In this section we show how to design an exact off-line mechanism when Π is utilitarian. Before defining our mechanism, we show how to compute an optimal sequence by using dynamic programming.

Let $\mathcal{H}^*$ denote an optimal solution for $\textsc{Seq}(\Pi)$, and let $\mathcal{H}^*_{[1,i]}$ be an optimal solution for $\textsc{Seq}(\Pi)$ when the input is restricted to the interval $[1, i]$, i.e. we have i time steps and the bid vector is $\langle b^1, \ldots, b^i\rangle$. In order to lighten the notation, we will write $\Psi(\mathcal{H}_{[1,i]}, b)$ instead of $\Psi(\mathcal{H}_{[1,i]}, \langle b^1, \ldots, b^i\rangle)$, where $\mathcal{H}_{[1,i]}$ is a solution for $\textsc{Seq}(\Pi)$ restricted to the interval $[1, i]$. Notice that $\mathcal{H}^*_{[1,1]} = \langle H^*_{[1,1]}\rangle$, and $\Psi(\mathcal{H}^*_{[1,1]}, b) = \phi(H^*_{[1,1]}, b^1) + \gamma_1$. Moreover, $\mathcal{H}^*_{[1,k]} = \mathcal{H}^*$.

The dynamic programming algorithm computes $H^*_{[i,j]}$, for every $1 \le i \le j \le k$. Next, starting from $i = 1$ to k, it computes $\mathcal{H}_{[1,i]} = \mathcal{H}_{[1,h_i)} \odot \langle H^*_{[h_i,i]}\rangle$, with

$$h_i = \arg \min_{h=1,\ldots,i} \left\{ \Psi'(b, h, i) := \Psi(\mathcal{H}_{[1,h)}, b) + \phi\big(H^*_{[h,i]}, b^{[h,i]}\big) + \gamma_h \right\},$$

where $\mathcal{H}_{[1,1)}$ is the empty sequence, and $\Psi(\mathcal{H}_{[1,1)}, b)$ is assumed to be 0.

The following lemma, whose proof is omitted due to lack of space, holds:

Lemma 3. *For any $i = 1, \ldots, k$, the dynamic programming algorithm computes a solution $\mathcal{H}_{[1,i]}$ such that $\Psi(\mathcal{H}_{[1,i]}, b) = \Psi(\mathcal{H}^*_{[1,i]}, b)$.* □

We are now ready to define our VCG-like mechanism. Let $\mathcal{M}_{\text{VCG}}$ be a mechanism defined as follows:

1. The algorithmic output specification selects an optimal sequence (w.r.t. the bids b) $\mathcal{H}^*_G$;
2. Let $G-e=(V,E\backslash\{e\})$, and let $\mathcal{H}^*_{G-e}$ be an optimal solution (w.r.t. the bids b) in $G-e$. Then, the payment function for a_e is defined as

$$p_e(b)=\Psi(\mathcal{H}^*_{G-e},b)-\Psi(\mathcal{H}^*_G,b)+\nu_e(\mathcal{H}^*_G,b_e).$$

From the above discussion, it is easy to prove the following

Theorem 3. *Let Π be utilitarian and solvable in polynomial time. Then, $\mathcal{M}_{\text{VCG}}$ is an exact off-line truthful mechanism for* $\textsc{Seq}(\Pi)$ *which can be computed in polynomial time.* □

6 Time-Sequenced Mechanisms: Inapproximability Results

In this section we consider a natural extension of TSSP mechanisms named *adaptive TSSP mechanisms*, and we prove a lower bound of k to the approximation ratio that can be achieved by any truthful mechanism in this class.

Definition 2. *Let δ be a function mapping bid vectors to interval vectors. An* adaptive time-sequenced single-parameter (ATSSP) mechanism *$\mathcal{M}_\delta$ for Π is the mechanism which, for a given vector bid b, is defined exactly as $\mathcal{M}(\delta(b))$.*

Lemma 4. *Let t^i be a type profile for Π, and let $\mathcal{A}$ be an optimal algorithm for Π. Then, $\forall\eta\in\mathbb{R}^+$, $\theta^i_e(\eta\cdot t^i_{-e})=\eta\cdot\theta^i_e(t^i_{-e})$.*

Proof. Observe that $\forall H\in\textsc{Sol}(\Pi)$

$$\phi(H,\eta\cdot t^i)=\sum_{e\in E(H)}\eta\cdot t^i_e\,\mu_H(e)=\eta\sum_{e\in E(H)}t^i_e\,\mu_H(e)=\eta\cdot\phi(H,t^i).$$

□

Theorem 4. *For any mapping function δ, for any optimal algorithm $\mathcal{A}$ for Π, and for any $c<k$, there exists no c-approximate truthful ATSSP mechanism using $\mathcal{A}$ for* $\textsc{Seq}(\Pi)$*, even when set-up costs are uniform.*

Proof. The proof is by contradiction. Let $M=\gamma_1=\cdots=\gamma_k$. Let $\mathcal{M}_\delta$ be a c-approximate truthful ATSSP mechanism for $\textsc{Seq}(\Pi)$. For the sake of clarity, we denote by $H(w)$ an optimum solution for Π with input (G,w). Let $t^1=(t^1_{-e},t^1_e)$, with $t^1_{-e}=\langle 0,\ldots,0\rangle$, and $t^2=(t^2_{-e},0)$ be two type vectors for Π such that the following three conditions hold:

(i) $2t^1_e<\theta^2_e,\ t^1_e>0$, where $\theta^2_e=\theta_e(t^2_{-e})$;
(ii) $\phi(H(t^2_{-e},+\infty),(t^2_{-e},+\infty))\geq(k^2-1)M$;
(iii) $\phi(H(t^2_{-e},x),(t^2_{-e},x))$ does not depend on M, for any $x<\theta^2_e$ not depending on M.

Lemma 5. *There always exist t^1_e and t^2_{-e} satisfying the above conditions.*

Proof. Let $H \in \textsc{Sol}(\Pi)$ be such that $E(H') \not\subset E(H), \forall H' \in \textsc{Sol}(\Pi)$. Let e be an edge of H. Now for each $e' \in E(H) \setminus \{e\}$, let $t^2_{e'} = \frac{1}{\mu_H(e')}$. Moreover, for each $e' \in E \setminus E(H)$, let $t^2_{e'}$ be defined as follows

$$t^2_{e'} = \max_{H' \in \textsc{Sol}(\Pi)} \frac{(k^2-1)M}{\mu_{H'}(e')}.$$

By construction, condition (ii) holds. For M large enough, it is easy to see that θ^2_e is at least $(k^2-1)M - |E(H)| > 0$, from which (i) follows as well. Finally, condition (iii) follows by observing that $\mu_H(e)$ does not depend on M. □

Let t be the type profile defined as follows:

$$\forall i = 1, \ldots, k, \quad t^i = \begin{cases} t^1 & \text{if } i \text{ is odd;} \\ t^2 & \text{otherwise.} \end{cases}$$

Lemma 6. *For M large enough, $\delta(t) \neq \tilde{s}$.*

Proof. The proof is by contradiction. Let $\mathcal{H}$ be the solution computed by the mechanism corresponding to the interval vector $\tilde{s}$. Notice that $\Psi(\mathcal{H}, t) \geq kM$, since $H(t^1) \neq H(t^2)$. Consider now the solution $\mathcal{H}'$ corresponding to the interval vector $\langle [1,k] \rangle$. It is easy to see that for t^1_e small enough, $\Psi(\mathcal{H}', t) = M + \phi\big(H\big(t^{[1,k]}\big), t^{[1,k]}\big) \leq M + k\,\phi(H(t^2_{-e}, t^1_e), (t^2_{-e}, t^1_e))$. It follows that the approximation ratio achieved by the mechanism is at least

$$\frac{\Psi(\mathcal{H}, t)}{\Psi(\mathcal{H}', t)} \geq \frac{kM}{M + k\,\phi(H(t^2_{-e}, t^1_e), (t^2_{-e}, t^1_e))},$$

which, from (iii), goes to k when M goes to $+\infty$. This contradicts the fact that $\mathcal{M}_\delta$ is c-approximate. □

Lemma 7. *For M large enough, the utility of a_e in the solution $g_{\delta(t)}(t)$ computed by the mechanism $\mathcal{M}_\delta$ is less than $\left\lfloor \frac{k}{2} \right\rfloor \theta^2_e$.*

Proof. Let $\delta(t) = \langle I_1, \ldots, I_h \rangle$ be the interval vector computed by δ, and let $\mathcal{H}$ be the corresponding solution. For each $j = 1, \ldots, h$, let $I_j = [x_j, y_j]$ be the j-th interval, and let η_j be the number of occurrences of t^2 in $\langle t^{x_j}, \ldots, t^{y_j} \rangle$. Notice that $t^{I_j} = (\eta_j\, t^2_{-e}, (|I_j| - \eta_j)\, t^1_e)$. It is easy too see that $(|I_j| - \eta_j) \leq \eta_j + 1$. Moreover, notice that e belongs to $H\big(t^{I_j}\big)$ iff $\eta_j > 0$. Indeed, whenever $\eta_j > 0$, $(\eta_j + 1)\, t^1_e < \eta_j\, \theta^2_e$ holds from (i), and from Lemma 4 this implies that e belongs to $H\big(t^{I_j}\big)$. Finally, notice that whenever $|I_j| > 1$, a_e incurs a cost of at least t^1_e.

Then, from Lemma 4, the payment provided to a_e is $\sum_{j=1}^{h} \eta_j\, \theta^2_e = \left\lfloor \frac{k}{2} \right\rfloor \theta^2_e$, while concerning the cost incurred by a_e, it is at least $t^1_e > 0$, since from Lemma 6 there must exist an index j^* such that $|I_{j^*}| > 1$. □

Consider now the following new type profile $\hat{t}$ which is equal to t except for $\hat{t}^i_e$ that is set to $+\infty$ for every odd i.

Lemma 8. *For M large enough, $\delta(\hat{t}\,) = \tilde{s}$.*

Proof. For the sake of contradiction, assume that $\delta(\hat{t}) \neq \tilde{s}$. Then, there must exist an index j for which the solution $\mathcal{H}$ computed by the mechanism does not change at time j. Hence, since either $\hat{t}_e^j = +\infty$ or $\hat{t}_e^{j-1} = +\infty$, from (ii) it must be $\Psi(\mathcal{H}, \hat{t}) \geq k^2 M$. Consider the solution $\mathcal{H}'$ corresponding to the interval vector $\tilde{s}$. Then, the approximation ratio achieved by the mechanism is at least

$$\frac{\Psi(\mathcal{H}, \hat{t})}{\Psi(\mathcal{H}', \hat{t})} \geq \frac{k^2 M}{kM + k\,\phi(H(t^2), t^2)},$$

which, from (iii), goes to k when M goes to $+\infty$. This contradicts the fact that $\mathcal{M}_\delta$ is c-approximate. □

To conclude the proof, observe that when the type profile is t, a_e has convenience to bid b_e defined as

$$\forall i = 1, \ldots, k, \quad b_e^i = \begin{cases} t_e^2 & \text{if } i \text{ is even;} \\ +\infty & \text{otherwise.} \end{cases}$$

Indeed, in this case, from Lemma 8, its utility becomes equal to $\left\lfloor \frac{k}{2} \right\rfloor \theta_e^2$, which is better than the utility it gets by bidding truthfully (see Lemma 7). □

Notice that, since in the uniform set-up cost case each set-up cost is upper bounded by γ_1, and since $\mathcal{M}(\tilde{s})$ belongs to the class ATSSP, then Theorem 4 implies that the upper bound in Theorem 2 is tight, when $\mathcal{A}$ is optimal (i.e., $\rho = 1$).

References

1. S. Alstrup, J. Holm, K. de Lichtenberg, and M. Thorup, Minimizing diameters of dynamic trees, *Proc. 24th International Colloquium on Automata Languages and Programming (ICALP'97)*, Vol. 1256 of LNCS, Springer-Verlag, 270–280.
2. A. Archer and É. Tardos, Truthful mechanisms for one-parameter agents, *Proc. 42nd IEEE Symp. on Foundations of Computer Science (FOCS'01)*, 482–491, 2001.
3. D. Bilò, L. Gualà, and G. Proietti, On the existence of truthful mechanisms for the minimum-cost approximate shortest-paths tree problem, *13th Coll. on Structural Information and Communication Complexity (SIROCCO'06)*, Vol. 4056 of LNCS, Springer-Verlag, 295-309.
4. P. Briest, P. Krysta, and B. Vöcking, Approximation techniques for utilitarian mechanism design, *Proc. 37th Ann. ACM Symp. on Theory of Computing (STOC'05)*, 39–48, 2005.
5. G.N. Frederickson, Data structures for on-line updating of minimum spanning trees, with applications, *SIAM J. Comput.*, 14(4):781–798, 1985.
6. E.J. Friedman and D.C. Parkes, Pricing WiFi at Starbucks: issues in online mechanism design, *4th ACM Conf. on Electronic Commerce (EC'03)*, 240–241, 2003.
7. L. Gualà and G. Proietti, A truthful (2-2/k)-approximation mechanism for the Steiner tree problem with k terminals, *Proc. 11th Int. Computing and Combinatorics Conference (COCOON'05)*, Vol. 3595 of LNCS, Springer-Verlag, 390–400.

8. L. Gualà and G. Proietti, Efficient truthful mechanisms for the single-source shortest paths tree problem, *Proc. 11th Int. Euro-Par Conf. (Euro-Par'05)*, Vol. 3648 of LNCS, Springer-Verlag, 941-951.
9. J. Hershberger and S. Suri, Vickrey prices and shortest paths: what is an edge worth?, *Proc. 42nd IEEE Symp. on Foundations of Computer Science (FOCS'01)*, 252–259.
10. J. Holm, K. de Lichtenberg, and M. Thorup, Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity, *J. of the ACM*, 48(4):723–760, 2001.
11. B. Kalyanasundaram and K. Pruhs, On-line network optimization problems, in *Online Algorithms: The State of the Art*, Vol. 1442 of LNCS, Springer-Verlag, 268–280, 1998.
12. M.-Y. Kao, X.-Y. Li, and W. Wang, Towards truthful mechanisms for binary demand games: a general framework, *Proc. 6th ACM Conf. on Electronic Commerce 2005*, 213–222, 2005.
13. N. Nisan and A. Ronen, Algorithmic mechanism design, *Games and Economic Behaviour* 35:166–196, 2001.
14. D.C. Parkes, S.P. Singh, and D. Yanovsky, Approximately efficient online mechanism design, *Adv. in Neural Information Processing Syst. 17 (NIPS'04)*, 2004.
15. D.D. Sleator and R.E. Tarjan, A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983.