Theoretical Computer Science 410 (2009) 886–899

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Fast payment schemes for truthful mechanisms with verification

Alessandro Ferrante, Gennaro Parlato, Francesco Sorrentino, Carmine Ventre *

Dipartimento di Informatica ed Applicazioni "R.M. Capocelli", Università di Salerno, via Ponte Don Melillo, I-84084 Fisciano (SA), Italy

ARTICLE INFO

Article history:
Received 7 December 2005
Received in revised form 26 January 2007
Accepted 8 December 2008
Communicated by A. Fiat

Keywords:
Algorithmic mechanism design
Mechanisms with verification
One-parameter agents

ABSTRACT

In this paper we study optimization problems with verifiable one-parameter selfish agents introduced by Auletta et al. [V. Auletta, R. De Prisco, P. Penna, P. Persiano, The power of verification for one-parameter agents, in: Proceedings of the 31st International Colloquium on Automata, Languages and Programming, ICALP, in: LNCS, vol. 3142, 2004, pp. 171–182]. Our goal is to allocate load among the agents, provided that the secret data of each agent is a single positive real number: the cost they incur per unit load. In such a setting the payment is given after the load completion, therefore if a positive load is assigned to an agent, we are able to verify if the agent declared to be faster than she actually is. We design truthful mechanisms when the agents' type sets are upper-bounded by a finite value. We provide a truthful mechanism that is $c \cdot (1+\epsilon)$-approximate if the underlying algorithm is c-approximate and weakly-monotone. Moreover, if type sets are also *discrete*, we provide a truthful mechanism preserving the approximation ratio of its algorithmic part. Our results improve the existing ones which provide truthful mechanisms dealing only with finite type sets and do not preserve the approximation ratio of the underlying algorithm. Finally, we give applications for our payment schemes. Firstly, we give a full characterization of the $Q\|C_{\max}$ problem by using our techniques. Even if our payment schemes need upper-bounded type sets, every instance of $Q\|C_{\max}$ can be "mapped" into an instance with upper-bounded type sets preserving the approximation ratio. In conclusion, we turn our attention to binary demand games. In particular, we show that the Minimum Radius Spanning Tree admits an exact truthful mechanism with verification achieving time (and space) complexity of the fastest centralized algorithm for it. This contrasts with a recent truthful mechanism for the same problem [G. Proietti, P. Widmayer, A truthful mechanism for the non-utilitarian minimum radius spanning tree problem, in: Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA, ACM Press, 2005, pp. 195–202] which pays a linear factor with respect to the complexity of the fastest centralized algorithm. Such a result is extended to several binary demand games studied in literature.

1. Introduction

Optimization problems dealing with resource allocation are classical algorithmic problems and they have been studied for decades in several models: centralized vs. distributed algorithms, on-line vs. off-line algorithms and so on. The underlying hypothesis has been that the input is available to the algorithm (either from the beginning in off-line algorithms or during its execution in on-line algorithms). This assumption turns out to be unrealistic in the context of modern networks like the

* Corresponding address: Computer Science Department, University of Liverpool, Ashton Building, Ashton Street, Liverpool, L69 3BX, UK. Tel.: +44 151 7954284.
E-mail addresses: ferrante@dia.unisa.it (A. Ferrante), parlato@dia.unisa.it (G. Parlato), sorrentino@dia.unisa.it (F. Sorrentino), Carmine.Ventre@liverpool.ac.uk (C. Ventre).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.12.024

Internet. Here, the various parts of the input are owned by *selfish* (but *rational*) agents as part of their private information (called the *type*) and thus the optimization algorithm will have to ask the agents for their type and then work on the *reported* types. In this context, it is realistic to assume that an agent will lie about her type if this leads to a solution S that she prefers, even in spite of the fact that S is not globally optimal.

The field of mechanism design is the branch of Game Theory and Microeconomics that studies ways of inducing, through payments, the agents to report their true type so that the optimization problem can be solved on the actual input. In this paper we study the design of algorithms for solving (or approximately solving) combinatorial optimization problems in presence of selfish agents.

Following the standard notation used in the study of approximation of combinatorial optimization problems (see, e.g., [21,2]), we consider problems defined as four-tuples $(\mathcal{I}, \mathsf{m}, \mathsf{sol}, \mathsf{goal})$, where $\mathcal{I}$ is the set of *instances* of the problem; $\mathsf{sol}(I)$ is the set of *feasible solutions* of instance I; $\mathsf{m}(S, I)$ is the *measure* of the feasible solution S of instance I and goal is either min or max. Thus, the optimization problem consists in finding a feasible solution S^* for instance I such that $\mathsf{m}(S^*, I) = \mathsf{opt}(I) := \mathsf{goal}_{S \in \mathsf{sol}(I)} \mathsf{m}(S, I)$. A *c-approximation* algorithm A for $\Pi = (\mathcal{I}, \mathsf{m}, \mathsf{sol}, \mathsf{goal})$ is such that

$$\forall I \in \mathcal{I}, \quad \max\left\{\frac{\mathsf{m}(A(I), I)}{\mathsf{opt}(I)}, \frac{\mathsf{opt}(I)}{\mathsf{m}(A(I), I)}\right\} \leq c.$$

In an optimization problem Π with *selfish agents*, there are m agents which *privately know* part of the input. Thus, every instance $I \in \mathcal{I}$ consists of two parts $I = (T, \sigma)$, where the vector $T = (t_1, t_2, \ldots, t_m)$ is the *private part* of the input and σ is the *public part* of the input. In particular, we assume that t_i is known only to agent i, for $i = 1, 2, \ldots, m$ and we call t_i the *type* of agent i. The *type set* Θ_i of agent i is the set of the possible types of agent i. In this setting, each agent will report some value $b_i \in \Theta_i$ (which can be different from her true type t_i). An algorithm A for the optimization problem Π with selfish agents receives as input the vector of bids $B = (b_1, b_2, \ldots, b_m)$, instead of the true instance T it is supposed to solve. Each selfish agent incurs some monetary *cost*, $\mathsf{cost}_i(S, t_i)$, depending on the feasible solution S and her private data t_i. Since every agent i is selfish, she might declare $b_i \neq t_i$ so to induce A to return a cheaper solution for agent i. Unfortunately, even though A is c-approximating for the instance T, for $B \neq T$ the solution returned by A on input B might have measure, w.r.t. the true instance T, far-off the optimum $\mathsf{opt}(T)$.

Truthful mechanisms. In order to obtain a correct solution, algorithm A is equipped with a *payment scheme* $P = (P_1, \ldots, P_m)$ in order to induce every agent to report her true type. After a solution $S = A(B, \sigma)$ is computed, each agent i is awarded payment $P_i(B, \sigma)$. We assume that each agent i is *rational* in the sense that she picks her type declaration b_i so to maximize her *profit*.

Definition 1. Let Π be an optimization problem with one-parameter selfish agents and A be an algorithm for Π, and P be a payment scheme. The profit function profit of agent i with respect to the pair (A, P) when B is the sequence of bids, σ is the public information, t_i is the true type of agent i, and $S = A(B, \sigma)$, is defined as follows.

$$\mathsf{profit}_i(B, \sigma, t_i) := P_i(B, \sigma) - \mathsf{cost}_i(S, t_i).$$

It is natural to consider mechanisms in which the profit of the i-th agent is maximized when she reports $b_i = t_i$. We have thus the following classical notion of a *truthful mechanism*. In the definition of a truthful mechanism (and in the rest of the paper) the following notation turns out to be useful. Let $X = (x_1, \ldots, x_k)$ be a vector. For any $1 \leq i \leq k$, the writing X_{-i} denotes the vector $X_{-i} := (x_1, \ldots, x_{i-1}, x_{i+1} \ldots, x_k)$ and the writing (y, X_{-i}) denotes the vector $(y, X_{-i}) := (x_1, \ldots, x_{i-1}, y, x_{i+1} \ldots, x_k)$.

Definition 2. The pair $\mathcal{M} = (A, P)$ is a *truthful mechanism for selfish agents* if and only if for all σ, for all agents i, and all type declarations B, it holds

$$\mathsf{profit}_i((t_i, B_{-i}), \sigma, t_i) \geq \mathsf{profit}_i(B, \sigma, t_i).$$

In mechanisms satisfying above definition, every agent maximizes her profit when she is sincere. Thus, we assume that in a truthful mechanism every agent always reports her type faithfully, and algorithm A always works on the true instance T. As a consequence, we say that a truthful mechanism $\mathcal{M} = (A, P)$ is *c-approximating* for an optimization problem Π with selfish agents, if A is a c-approximating algorithm for every instance I of Π. Since, in a truthful mechanism, agents are not sure to have a positive profit, they would not participate in such a mechanism unless they were coerced. This motivates the following definition.

Definition 3. A truthful mechanism satisfies *voluntary participation* condition if agents who bid truthfully never incur a net loss, i.e. for all public information σ, for all agents i, and for all other agents' bids B_{-i},

$$\mathsf{profit}_i((t_i, B_{-i}), \sigma, t_i) \geq 0.$$

One-parameter selfish agents. We now review the concept of optimization problem Π with *one-parameter selfish agents* (as discussed in [7]). Here, each agent i has as private information a single parameter $t_i \in \mathbb{R}$. Moreover, a feasible solution S of an instance I of Π defines, for each agent i, an amount $\mathsf{w}_i(S)$ of assigned work. We call such a solution S *schedule*. Notice that in the definition of one-parameter problem [7] the total amount of work to schedule can depend on the private part of the input B. However, we restrict ourselves, as in wide part of literature, to the case in which the amount of work assigned to all agents depends only on the public information (and not on the agent bids). We denote such an amount of load just as $\mathcal{W} > 0$. The cost function of agent i has the following special form.

Definition 4. Let S be a feasible solution of Π. Then, the cost function $\mathsf{cost}_i(S, t_i)$ is defined as follows.

$$\mathsf{cost}_i(S, t_i) := \mathsf{w}_i(S) \cdot t_i.$$

Scheduling problems for one-parameter selfish machines. Scheduling problems are typical examples of optimization problem for one-parameter selfish agents. In a *scheduling problem*, the input consists of m machine speeds $s = (s_1, s_2, \ldots, s_m)$ and n job weights $W = (w_1, w_2, \ldots, w_n)$. A schedule S is an assignment of jobs to machines. Let $\mathsf{w}_i(S)$ be the sum of the weights of the jobs assigned to machine i by schedule S. In a scheduling problem the task consists in computing a schedule that minimizes a certain cost function associated with the schedule. For instance, in the $Q \parallel C_{\max}$ problem the cost of a schedule S is the *makespan* $MS(S)$ that is the maximum completion time of the machines. Formally, $MS(S) = \max_{1 \le j \le m}\{\frac{\mathbf{w}_j(S)}{s_j}\}$. We consider the setting in which each machine i is owned by a different agent and the speed s_i of machine i is the private information of agent i. To be in a setting of one-parameter selfish agents, we consider $t_i = 1/s_i$ as the type of agent i. The public information σ is the sequence $W = (w_1, w_2, \ldots, w_n)$ of job weights. We recall that $Q \parallel C_{\max}$ problem is NP-hard.

Truthful mechanisms with verification for one-parameter selfish agents. A mechanism with verification $\mathcal{M}$ for one-parameter selfish agents is a pair $\mathcal{M} = (A, P)$ working as follows.

1. The allocation algorithm A takes as input the sequence of bids $B = (b_1, b_2, \ldots, b_m)$ and the public part σ and outputs a schedule $S = A(B, \sigma)$ for the m agents. We recall that $\mathsf{w}_i(S)$ denotes the amount of load assigned to agent i by the schedule S computed by algorithm A on input B and σ.
2. Each agent i is *observed* to complete her assigned load in time $\mathcal{T}_i \ge \mathsf{w}_i(S) \cdot t_i$. Notice that agent i completes the load $\mathsf{w}_i(S)$ assigned to her in time $\mathsf{w}_i(S) \cdot t_i$. Agent i can however delay the release of the works and thus obtain a larger observed completion time and the mechanism has no way of detecting it. However, agent i cannot be observed to finish her load before the actual completion time $\mathsf{w}_i(S) \cdot t_i$. Since $\mathsf{w}_i(S) \cdot t_i$ is the request time for agent i to complete the load $\mathsf{w}_i(S)$, we denote with $s_i = \frac{1}{t_i}$ the *speed* of agent i.
3. Finally, after agent i releases the assigned works, she is awarded payment computed by applying function P_i on arguments B, σ, and the *observed completion time* $\mathcal{T}_i$ of machine i.

We stress that in this setting, payments are provided *after* the execution of the load and thus agents are (partially) *verifiable* in the following sense. If agent i receives an amount of load greater than 0, the mechanism can find out whether agent i has declared to be faster than she actually is (that is, $b_i < t_i$). Indeed, in this case the claimed completion time $\mathsf{w}_i(S) \cdot b_i$ is smaller than the actual completion time $\mathsf{w}_i(S) \cdot t_i$ and thus we have that $\mathcal{T}_i \ge \mathsf{w}_i(S) \cdot t_i > \mathsf{w}_i(S) \cdot b_i$. Since payments are provided *after* the completion of loads, the mechanism can make it inconvenient to claim faster speeds. On the other hand, the mechanism cannot find out if an agent has declared to be slower than she actually is, since the agent can decide to delay some of the jobs.

Henceforward we refer to Π as an optimization problem for verifiable one-parameter selfish agents. Let us now instantiate the definitions of profit and truthful mechanism in this new scenario.

Definition 5. Let A be an algorithm for Π, and P be a payment scheme. The profit function profit of agent i with respect to the pair (A, P), when B is the sequence of bids, σ is the public information, t_i is the true type of agent i, $S = A(B, \sigma)$, and $\mathcal{T}_i$ is the observed completion time of agent i for the load $\mathsf{w}_i(S)$, is defined as follows.

$$\mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) := P_i(B, \sigma|\mathcal{T}_i) - \mathsf{cost}_i(S, t_i).$$

Definition 6. Let A be an algorithm for Π, and P be a payment scheme. A pair $\mathcal{M} = (A, P)$ is a truthful mechanism with verification for Π, if for all σ, for all i, for all bid vectors B, and for all observed completion times $\mathcal{T}_i$, it holds that

$$\mathsf{profit}_i((t_i, B_{-i}), \sigma, t_i|\mathsf{w}_i(S) \cdot t_i) \ge \mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i)$$

where $S = A((t_i, B_{-i}), \sigma)$.

Note that these new definitions are not redundant, since in this case we have to take into account the observed completion time also.

Given a truthful mechanism $\mathcal{M} = (A, P)$ for Π, in [5] the authors give a necessary condition that algorithm A must satisfy.

Definition 7 (*Weakly-Monotone Algorithm*). Let Π be an optimization problem for verifiable one-parameter selfish agents and A be an algorithm for Π. Algorithm A is *weakly-monotone* if and only if, for all σ, for all i, for all declared bid vectors B such that $\mathsf{w}_i(A(B, \sigma)) = 0$ and for all $b'_i \in \Theta_i$ with $b'_i > b_i$ it holds that

$$\mathsf{w}_i(A((b'_i, B_{-i}), \sigma)) = 0.$$

In other words a weakly-monotone algorithm A has the following property. Fix some input (B, σ) for which algorithm A assigns no load to agent i. If agent i declares to be slower (that is, she declares $b'_i > b_i$) and the declared bids of the other agents remain the same, then A assigns no load to agent i.

Lemma 8 (*[5]*). *Let Π be an optimization problem for verifiable one-parameter selfish agents. If $\mathcal{M} = (A, P)$ is a truthful mechanism for Π, then A is a weakly-monotone algorithm.*

2. Previous works and our contribution

2.1. Previous works

The celebrated VCG mechanism [8,11,12,23] is the prominent technique to derive truthful mechanisms for optimization problems. However, this technique applies only to *utilitarian* problems, that are problems where the objective function is equal to the sum of the cost functions of the agent (e.g., shortest path, minimum spanning tree, etc.). In the seminal papers by Nisan and Ronen [17,16] it is pointed out that VCG mechanisms do not completely fit in a context where computational issues play a crucial role since they assume that it is possible to compute an optimal solution of the corresponding optimization problem (maybe a NP-hard problem). Scheduling is a classical optimization problem that is not utilitarian (since we aim at minimizing the maximum over all machines of their completion times) and it is NP-hard. Moreover, scheduling models important features of different allocation and routing problems in communication networks. Thus, it has been the first problem for which non-VCG-based techniques have been introduced.

In [7], is considered the variant of the task scheduling on *uniformly related* machines (in short $Q \parallel C_{\max}$), where each machine i has a speed s_i and the processing time of a task is given by the ratio between the weight of the task and the speed of the machine. They characterized the class of allocation algorithms A for one-parameter problems that admit payment scheme P for which $\mathcal{M} = (A, P)$ is a truthful mechanism. Essentially, truthful mechanisms for one-parameter selfish agents must use *monotone* algorithms and, in this case, the payment scheme is uniquely determined (up to an additive factor). Intuitively, monotonicity means that increasing the speed of exactly one machine does not make the algorithm decrease the work assigned to that machine. The result of [7] reduces the problem of designing a truthful mechanism for $Q \parallel C_{\max}$ to the algorithmic problem of designing a good algorithm which also satisfies the additional monotonicity requirement. Efficient mechanisms for computing scheduling on related machines with small makespan (a special case of one-parameter agents) have been provided by Archer and Tardos [7] and, subsequently by Auletta et al. [4] and by Andelman, Azar and Sorani [1].

Mechanisms with verification. A natural generalization of the scheduling problem discussed above is scheduling on *unrelated machines* (see, e.g., [17]). In such a problem one wants to design a schedule minimizing the makespan with machines having speeds depending on the job they execute: i.e., machine i has a speed s_i^j for executing job j. Due to the difficulty of the problem and to lower bounds on the approximation ratio of "classical" mechanisms without verification (namely, $1 + \Phi$ is the best known lower bound on the approximation of *any* truthful classical mechanism [14]), Nisan and Ronen [17] defined mechanisms with verification. In this model, mechanisms use the actual execution of the jobs to verify agents declarations deferring payments expense to *after* the execution of the jobs.

Afterwards, Auletta et al. [5] have considered optimization problems for verifiable one-parameter problems. They specialized the concept of mechanisms with verification to the case of one-parameter agent (scheduling on unrelated machines does not involve one-parameter agents). They show that, in order to have a truthful mechanism for verifiable one-parameter selfish agents, a necessary condition is that the used algorithm must be *weakly-monotone*. Conversely, when agents type sets are *finite* they have given a (frugal) payment scheme leading to truthful mechanisms with verification for *any* weakly-monotone algorithm. This witnesses the power of verification for one-parameter agents. Indeed, as already said, Archer and Tardos [7] have shown that a mechanism without verification is truthful for one-parameter agents if and only if the algorithm is monotone. On the other hand, the authors of [5] have shown that weak-monotonicity is necessary and sufficient for all finite type sets. Therefore, the class of algorithms that leads to truthful mechanisms with verification is larger than the class of the algorithms part of classical truthful mechanisms. In [3], the authors have shown as the capability of mechanisms with verification of truthfully implementing – an algorithm A is truthfully implemented (with verification) if there exists a payment function P such that (A, P) is a truthful mechanism (with verification) – more algorithms than classical mechanisms do actually gives non-trivial extra power to mechanisms with verification (see, for example, the $(1 + \epsilon)$-approximate mechanism[1] for weighted sum scheduling breaking the 1.54 lower bound for classical mechanisms [7]). Verification is also helpful in more general settings: generalization of one-parameter type sets (see [3] for the class of comparable types), optimal mechanisms for agents bidding from any finite type set [22] and collusion-resistant mechanisms [19].

2.2. Our contribution

In this work, we extend some results given in [5]. The authors were the first to study optimization problems for verifiable one-parameter selfish agents. We recall that intuitively a verifiable agent is an agent that may lie in reporting its type but the mechanism can verify whether agent i underbids (i.e. declares $b_i < t_i$), provided that the load assigned to this agent is positive. For instance, for scheduling problems the mechanism can *verify*, through the observed completion time of agent i, if she declares to be faster then she actually is, provided that at least one job has been assigned to her.

In Section 3.1, we give simple and efficient payment schemes, leading to polynomial-time truthful mechanisms for a wide class of optimization problems with verifiable one-parameter selfish agents. In particular, we provide a payment function

[1] We highlight that such a mechanism uses one of the payment schemes we are going to define.

Table 1
Comparing results (c is the approximation of a given weakly-monotone algorithm and ϵ is related to the smoothness of the problem — see Definition 15).

Problem version	Payments time complexity	Approximation ratio
Θ_i finite [5]	poly($\|\Theta_i\|, m, n$)	c
Smooth problems with finite Θ_i [5]	poly($\log_{1+\epsilon} \|\Theta_i\|, m, n$)	$c \cdot (1+\epsilon)$
Θ_i upper bounded and discrete (not finite)	poly(m, n)	c
Smooth problems with Θ_i upper bounded (continuous)	poly(m, n)	$c \cdot (1+\epsilon)$

$P^{(1)}$ that works for *discrete* and *upper-bounded* type sets. In this setting, we need that agents bid from sets in which there is always a gap between the inverse of two types — in scheduling problems (where types are the inverse of machines' speed), our assumption is satisfied when it is not possible to have machines executing j instructions per second, for every possible $j \in \mathbb{R}$, and this j must be lower bounded by a finite value. Indeed, in the market there are only machines of certain (sufficiently far apart) speeds, and infinitely slow machine does not exist. Thus, our hypothesis applies to many real-life applications.

From a theoretical point of view, our results improve those given in [5]: (i) the class of the discrete and upper-bounded type sets properly includes the class of finite type sets; (ii) our mechanism preserves the approximation ratio c of the algorithm it uses, while the mechanism given in the paper [5] needs that the problem is *smooth* (see Definition 15) to obtain a $c \cdot (1+\epsilon)$-approximation. (This assumption is required to round the input bids to get payments computable in polynomial time.)

In Section 3.2, we give a payment scheme $P^{(2)}$ leading to polynomial-time truthful mechanisms with verification (Theorem 14), for agents having real and upper-bounded type sets. In order to obtain truthful mechanism we round the agents' bid. Using this rounding technique, if the algorithm that mechanism uses is c-approximate, then nothing can be said about the approximation of the same algorithm when it runs on rounded bids. However, if the problem is smooth then the mechanism is $c \cdot (1+\epsilon)$-approximate (see Theorem 17). To best of our knowledge this is the *first* result showing that weak monotonicity of algorithms is a sufficient condition for the existence of truthful mechanisms for optimization problems with verifiable one-parameter selfish agents with continuous type sets. It is open the case when type sets are not upper-bounded. Table 1 summarizes and compares our results and those of [5].

Finally, in Section 4 we show applications our results. In Section 4.1 we study the $Q \parallel C_{\max}$ problem. Such a problem has already been studied in [5] where authors provide a frugal payment scheme for it. But, their scheme suffers from the fact of just working on finite type sets, and it is unknown if it is possible to extend their technique to more general domains. Therefore, we address the question of fully characterizing (i.e., for *every* one-parameter domain) the $Q \parallel C_{\max}$ problem by using the more expensive (with respect to the one in [5]) technique we developed. In particular, we reduce any unbounded instance to a bounded one, obtaining a polynomial-time $c \cdot (1+\epsilon)$-approximate truthful mechanism, provided a c-approximate weakly-monotone polynomial-time algorithm. In particular, if the type sets are discrete, we obtain a polynomial-time truthful mechanism (with verification) preserving the approximation ratio of a given weakly-monotone polynomial-time algorithm. Moreover, we extend our technique to deal with the online version of $Q \parallel C_{\max}$ problem thus obtaining a 12 competitive truthful mechanism with verification for the online $Q \parallel C_{\max}$ problem exploiting an online weakly monotone algorithm presented in [6]. In Section 4.2 we turn our attention to binary demand games. These are problems for which monotone algorithms (in the sense of [7], see Definition 26) are equivalent to weakly-monotone ones (for the details see also [3]). The idea is improving extant truthful mechanisms using the efficiency of our payments. This is the case of the mechanism for the Minimum Radius Spanning Tree (MRST) (presented in [20]). For the latter we provide a truthful mechanism with verification achieving time (and space) complexity of the fastest known centralized algorithm. We extend this technique to several problems studied in literature by showing as our payment schemes help in obtaining performances very close to non-strategic algorithms (i.e., classical algorithms with all the input available).

Features of our payment schemes. Both our payment schemes are bid dependent (as for payment scheme presented in [5]). This seems to be a sufficient condition for payment functions leading to truthful mechanisms with verification. This fact contrasts with a known result for classical truthful mechanisms (without verification) for which suitable payment schemes must be bid independent. In particular, payment functions P leading to classical truthful mechanisms (A, P) depend on the selected outcome: i.e., fixed σ and B_{-i}, agent i declaring two different bids b, b' in Θ_i, such that $A((b, B_{-i}), \sigma) = A((b', B_{-i}), \sigma)$, will get the same amount of money for both declarations. The intuition here is the following: if i's true type is b then i can declare b' and since in both cases the i's cost is the same (being the selected outcome the same) the payment in b must be at least the payment in b'. Same reasoning holds true by considering b' as the i's true type. Thus, the payment has to be the same. This can fail in the verification setting since it can happen that agent i with true type b (respectively b') declaring b' (respectively b) is caught by the verification.

Depending on the shape of the type sets, our payment schemes can become quite expensive. On the contrary, payment schemes in [5] are frugal. But they suffer from several limitations. Firstly, they are able to implement with verification weakly-monotone algorithms just for finite type sets. Further, for general problems, they run in polynomial time if and only if every type set has a size polynomial in the size of the input (i.e., the m bids) (see Table 1). Polynomial-time payment schemes of [5] work just for the subclass of smooth problems. Our payment schemes are always computable in polynomial time and, for general problems, implement any weakly-monotone algorithm for upper-bounded discrete type sets (the class

of upper-bounded discrete type sets includes the class of finite domains and particular infinite domains). Moreover, for the class of smooth problems, they are polynomial-time computable and coupled to a weakly-monotone algorithm lead to a truthful mechanism with verification for all continuous upper-bounded type sets. Therefore, our schemes are interesting from a theoretical point of view (see, for example, their recent applications in [3]) and represent the only general way to obtain truthful mechanisms with verification for general one-parameter agents' type sets.

3. Our payment schemes

In this section we will define our payment schemes. In particular, we start by defining in Section 3.1 the payment scheme for *discrete* and *upper-bounded* types. At the end of the same section it will be explained the intuition behind our schemes. In Section 3.2 we will extend the presented idea to more interesting and natural case: real upper-bounded type sets.

3.1. A payment scheme for discrete types

Facing with a scheduling problem in real life, one is the following scenario: a set of machines of (*sufficiently far apart*) given speeds and a function (of these speeds) to optimize. Looking for optimal solutions one can exclude machines that are *very slow*. Thus, the following definition seems to capture the described real-life scenario.

Definition 9. A set Θ_i is said *discrete* and *upper-bounded* if:

- there exists a value $\Delta_i \in \mathbb{R}^+$ such that, for all $b, \bar{b} \in \Theta_i$, $b \neq \bar{b}$, $|b^{-1} - \bar{b}^{-1}| \geq \Delta_i$ *(discrete)*;
- there exists a finite value $\sup_i \in \mathbb{R}^+$ such that $\sup_i \geq b,\ \forall b \in \Theta_i$ *(upper-bounded)*.

In this section, we will consider only type sets Θ_i discrete and upper-bounded. Next, we define a payment scheme which allows us to construct truthful mechanisms with verification for optimization problems.

Definition 10. Let B be a bid vector, σ be the public part of the input, $\mathcal{T}_i$ be the observed completion time and $c_i^{(1)} \in \mathbb{R}^+$ be a constant (to be given). Given an algorithm A, for each $i = 1, \ldots, m$, we define

$$P_i^{(1)}(B, \sigma|\mathcal{T}_i) := \begin{cases} \frac{w}{b_i} \cdot c_i^{(1)} & \text{if } \mathsf{w}_i(S) \neq 0 \text{ and } \mathcal{T}_i = \mathsf{w}_i(S) \cdot b_i; \\ 0 & \text{otherwise,} \end{cases}$$

where $S = A(B, \sigma)$.

The idea behind the payment $P_i^{(1)}$ is to give to agent i a disincentive to declare to be slower than she actually is. On the other hand, agent i is also discouraged to declare to be faster, if we use verification and weakly-monotone algorithms, as shown in the next theorem.

Theorem 11. *Let Π be an optimization problem for verifiable one-parameter selfish agents and A be a polynomial-time weakly-monotone algorithm for Π. If every Θ_i is upper-bounded by a finite value $\sup_i$ and discrete w.r.t. a known value Δ_i, then for every $1 \leq i \leq m$ there exists a value for the constant $c_i^{(1)}$ such that $\mathcal{M} = (A, P^{(1)})$ is a polynomial-time truthful mechanism with verification for Π. Moreover, $\mathcal{M}$ satisfies voluntary participation condition.*

Proof. Let S_{t_i} be the schedule computed by A when takes as input (t_i, B_{-i}), and S_{b_i} be the one on the input (b_i, B_{-i}). To demonstrate that $\mathcal{M}$ is a truthful mechanism with verification, we show that the following relation holds.

$$\mathsf{profit}_i((t_i, B_{-i}), \sigma, t_i|\mathsf{w}_i(S_{t_i}) \cdot t_i) - \mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) \geq 0; \qquad \forall\, b_i \in \Theta_i, \forall\, \mathcal{T}_i.$$

For sake of readability we denote (t_i, B_{-i}) as $\mathbf{T}$ and $\mathsf{w}_i(S_{t_i}) \cdot t_i$ as $\mathcal{T}_i^*$.
We first consider the case $\mathsf{w}_i(S_{b_i}) = 0$. From Definition 10 we have that $\mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) = 0$.

$$\begin{aligned} \mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) - \mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) &= \mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) \\ &= \frac{\mathcal{W}}{t_i} \cdot c_i^{(1)} - \mathsf{w}_i(S_{t_i}) \cdot t_i \\ &\geq \frac{\mathcal{W}}{t_i} \cdot c_i^{(1)} - \mathcal{W} \cdot t_i \\ &\geq \mathcal{W} \cdot \left(\frac{c_i^{(1)}}{\sup_i} - \sup_i \right) \geq 0 \end{aligned} \tag{1}$$

for all the values $c_i^{(1)} \geq \sup_i^2$.

By the above calculations, we also have that $\mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) \geq 0$, and thus $\mathcal{M}$ satisfies voluntary participation condition. Let $\mathsf{w}_i(S_{b_i}) > 0$. We distinguish two cases.

Case 1 ($b_i > t_i$). Since A is weakly-monotone it holds that $\mathsf{w}_i(S_{t_i}) > 0$. If $\mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) < 0$, from Eq. (1) we have

$$\mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) - \mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) > 0$$

for $c_i^{(1)} \geq \sup_i^2$. Let $\mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) \geq 0$. Then we have:

$$\begin{aligned}\mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) - \mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) &= \mathcal{W} \cdot \left(\frac{1}{t_i} - \frac{1}{b_i}\right) \cdot c_i^{(1)} - (\mathsf{w}_i(S_{t_i}) - \mathsf{w}_i(S_{b_i})) \cdot t_i \\ &\geq \mathcal{W} \cdot \Delta_i \cdot c_i^{(1)} - \mathcal{W} \cdot \sup_i \geq 0\end{aligned}$$

for all the values $c_i^{(1)} \geq \sup_i/\Delta_i$.

Case 2 ($b_i < t_i$). Since $\mathcal{T}_i > \mathsf{w}_i(S_{b_i}) \cdot b_i$, we have that $P_i^{(1)}(B, \sigma|\mathcal{T}_i) = 0$ and $\mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) < 0$. Therefore, from Eq. (1) we have:

$$\mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) - \mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) > \mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) \geq 0$$

for $c_i^{(1)} \geq \sup_i^2$.

Hence, for $c_i^{(1)} \geq \max\{\sup_i^2, \frac{\sup_i}{\Delta_i}\}$, $\mathcal{M}$ is truthful (with verification).

It is straightforward that payment scheme $P^{(1)}$ is computable in polynomial time. □

As argued in Section 1, if A is the algorithm used in a truthful mechanism, then it always works on true types, since every agent always reports her true type. As a consequence, if A is c-approximate and $\mathcal{M} = (A, P)$ is a truthful mechanism then $\mathcal{M}$ is c-approximate as well. Thus, from Theorem 11 we have the following.

Theorem 12. *Let Π be an optimization problem for verifiable one-parameter selfish agents and A be a polynomial-time c-approximating weakly-monotone algorithm for Π. If every Θ_i is upper-bounded by a finite value $\sup_i$ and is discrete w.r.t. a known value Δ_i, then $M = (A, P^{(1)})$ is a polynomial-time c-approximate truthful mechanism with verification for Π, satisfying voluntary participation condition.*

Notice that if a type set is finite then it is discrete and finitely upper-bounded. Conversely, if a type set is discrete and finitely upper-bounded it could contain infinite values. For instance consider the case in which for every $i = 1, \ldots, m$, $\Theta_i = \{i^{-1} | i \in \mathbb{N}\}$. This is a special case of the discrete and upper-bounded type set: $\Delta_i = 1$ and $\sup_i = 1$, for every type set Θ_i, thus implying $c_i^{(1)} = 1$. In this case, for a given schedule S, the payment scheme $P_i^{(1)}$ becomes simply $\mathcal{W} \cdot \frac{1}{b_i}$ (in the case $\mathsf{w}_i(S) \neq 0$ and $\mathcal{T}_i = \mathsf{w}_i(S) \cdot b_i$). A different way to read this payment is to consider $\frac{1}{b_i}$ as the speed of agent i. Hence, to an agent is awarded the total work to schedule times the reported speed. The novelty of the last definition is exactly that the payment completely depends by the reported bid. Indeed, declaring to be very fast (therefore declaring a very little bid) the payment grows up. On the other hand, latter theorem ensures that, using verification, an agent is discouraged to declare to be faster than she actually is.

3.2. A payment scheme for real types

In this section, we show how to extend our payments in order to deal with real type sets which are only upper-bounded by a finite value $\sup_i$. To do that, we apply a rounding technique on types. Given a bid vector B, we denote with B^R the vector obtained by B by replacing each element b_i with a rounded value b_i^R of b_i. If $\alpha^\gamma < b_i^{-1} \leq \alpha^{\gamma+1}$, then $b_i^R = 1/\alpha^{\gamma+1}$ for some $\gamma \in \mathbb{Z}$. Thus, if $B = (b_1, b_2, \ldots, b_m)$ then $B^R = (b_1^R, b_2^R, \ldots, b_m^R)$. Given an algorithm A for Π, we define algorithm A_α as the algorithm that, on input B and σ, simply run algorithm A on input B^R and σ.

Definition 13. Let B be a bid vector, σ be the public part of the input, $\mathcal{T}_i$ be the observed completion time and $c_i^{(2)} \in \mathbb{R}^+$ be a constant (to be given). Given an algorithm A, for each $i = 1, \ldots, m$, we define

$$P_i^{(2)}(B, \sigma|\mathcal{T}_i) := \begin{cases} \frac{w}{b_i^R} \cdot c_i^{(2)} & \text{if } \mathsf{w}_i(S) \neq 0 \text{ and } \mathcal{T}_i = \mathsf{w}_i(S) \cdot b_i; \\ 0 & \text{otherwise,} \end{cases}$$

where $S = A(B, \sigma)$.

The idea behind payment scheme $P^{(2)}$ is similar to the one for $P^{(1)}$. The difference is that we consider the rounded bid b_i^R instead of the declared bid b_i and the used constant $c_i^{(2)}$ is essentially different from $c_i^{(1)}$. In the next theorem, we will better clarify the meaning of constant $c_i^{(2)}$.

Theorem 14. *Let Π be an optimization problem for verifiable one-parameter selfish agents whose types are positive real and let A be a polynomial-time weakly-monotone algorithm for Π. If every Θ_i is upper-bounded by a finite value $\sup_i$, then for every $1 \leq i \leq m$ there exists a value for the constant $c_i^{(2)}$, such that $\mathcal{M} = (A_\alpha, P^{(2)})$ is a polynomial-time truthful mechanism with verification for Π. Moreover, $\mathcal{M}$ satisfies voluntary participation condition.*

Proof. First note that, if A is weakly-monotone then A_α is weakly-monotone as well. Let B be a vector of bids, and S_{t_i} be the schedule computed by algorithm A_α when takes as input (t_i, B_{-i}), and S_{b_i} be the one on input (b_i, B_{-i}). To show that $\mathcal{M}$ is truthful with verification, we prove that

$$\mathsf{profit}_i((t_i, B_{-i}), \sigma, t_i, \mathsf{w}_i(S_{t_i}) \cdot t_i) - \mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) \geq 0; \qquad \forall\, b_i \in \Theta_i, \forall\, \mathcal{T}_i.$$

For sake of readability we denote $\mathbf{T} = (t_i, B_{-i})$ and $\mathcal{T}_i^* = \mathsf{w}_i(S_{t_i}) \cdot t_i$. We firstly consider the case $\mathsf{w}_i(S_{b_i}) = 0$. For some $\gamma \in \mathbb{Z}$, it holds:

$$\begin{aligned}
\mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) - \mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) &= \mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) && (2)\\
&= \frac{\mathcal{W}}{t_i^R} \cdot c_i^{(2)} - \mathsf{w}_i(S_{t_i}) \cdot t_i\\
&\geq \frac{\mathcal{W}}{t_i^R} \cdot c_i^{(2)} - \mathcal{W} \cdot \frac{1}{\alpha^\gamma}\\
&= \mathcal{W} \cdot \left(\alpha^{\gamma+1} \cdot c_i^{(2)} - \frac{1}{\alpha^\gamma}\right). && (3)
\end{aligned}$$

By simple calculations we have that Eq. (3) is greater or equal to 0 when

$$\gamma \geq -\frac{\log_\alpha c_i^{(2)}}{2} - \frac{1}{2}. \tag{4}$$

At the end of the theorem, we discuss how to choose $c_i^{(2)}$ in order $\mathcal{M}$ to be truthful (with verification). From Eq. (2), we also have that $\mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) \geq 0$, thus $\mathcal{M}$ satisfies voluntary participation condition.

It remains to show the case $\mathsf{w}_i(S_{b_i}) > 0$. We distinguish two cases:

Case 1 $(b_i > t_i)$. Since A is weakly-monotone and $b_i^R \geq t_i^R$ it holds that $\mathsf{w}_i(S_{t_i}) > 0$. W.l.o.g., we only consider the case in which $\mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) \geq 0$. We first analyze the case $b_i^R = t_i^R$, i.e. b_i and t_i are rounded to the same power of α.

$$\begin{aligned}
\mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) - \mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) &= \mathcal{W} \cdot \left(\frac{1}{t_i^R} - \frac{1}{b_i^R}\right) \cdot c_i^{(2)} - \left(\mathsf{w}_i(S_{t_i}) - \mathsf{w}_i(S_{b_i})\right) \cdot t_i\\
&= \mathcal{W} \cdot \left(\frac{1}{t_i^R} - \frac{1}{t_i^R}\right) \cdot c_i^{(2)} - \left(\mathsf{w}_i(S_{t_i}) - \mathsf{w}_i(S_{t_i})\right) \cdot t_i = 0.
\end{aligned}$$

Here, we analyze the remaining case in which $b_i^R > t_i^R$. Then, for some $\gamma \in \mathbb{Z}$, it holds:

$$\begin{aligned}
\mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) - \mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) &= \mathcal{W} \cdot \left(\frac{1}{t_i^R} - \frac{1}{b_i^R}\right) \cdot c_i^{(2)} - \left(\mathsf{w}_i(S_{t_i}) - \mathsf{w}_i(S_{b_i})\right) \cdot t_i\\
&\geq \mathcal{W} \cdot \left(\frac{1}{t_i^R} - \frac{1}{b_i^R}\right) \cdot c_i^{(2)} - \mathcal{W} \cdot t_i\\
&\geq \mathcal{W} \cdot \left(\alpha^{\gamma+1} - \alpha^\gamma\right) \cdot c_i^{(2)} - \mathcal{W} \cdot \frac{1}{\alpha^\gamma}\\
&= \mathcal{W} \cdot \left((\alpha^{\gamma+1} - \alpha^\gamma) \cdot c_i^{(2)} - \frac{1}{\alpha^\gamma}\right). && (5)
\end{aligned}$$

By simple calculations we have that Eq. (5) is greater or equal to 0 when:

$$\gamma \geq -\frac{\log_\alpha(c_i^{(2)})}{2} - \frac{\log_\alpha(\alpha - 1)}{2}. \tag{6}$$

As in the previous case, we postpone the discussion of choosing $c_i^{(2)}$ for the end of the theorem.

Case 2 $(b_i < t_i)$. Since $\mathcal{T}_i > \mathsf{w}_i(S_{b_i}) \cdot b_i$, we have that $P_i^{(2)}(B, \sigma|\mathcal{T}_i) = 0$ and $\mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) < 0$, implying that $\mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) - \mathsf{profit}_i(B, \sigma, t_i|\mathcal{T}_i) \geq \mathsf{profit}_i(\mathbf{T}, \sigma, t_i|\mathcal{T}_i^*) \geq 0$.

Here we discuss how to choose the constant $c_i^{(2)}$ in order to satisfy both Eqs. (4) and (6), for any value of γ. In particular, we just show for the case in which $\gamma = \gamma_{\min}$, where $\gamma_{\min}$ is the minimal value that γ can have. Since $\frac{1}{\sup_i} \leq (\frac{1}{\sup_i})^R = \alpha^{\gamma_{\min}}$, then by simple calculation we have $\gamma_{\min} = \lceil \log_\alpha \frac{1}{\sup_i} \rceil$. Since $\log_\alpha(c_i^{(2)})$ can have as value any real number by varying $c_i^{(2)}$, we can compute a value of $c_i^{(2)}$ such that both Eqs. (4) and (6) are satisfied when $\gamma = \gamma_{\min}$. Hence, it is straightforward that payment scheme $P^{(2)}$ is computable in polynomial time. □

Discussion about the rule of constant $c_i^{(2)}$. If in Definition 13 the constant $c_i^{(2)}$ was not used, then from Eqs. (4) and (6) we may observe that in order $\mathcal{M}$ to be truthful with verification, type sets Θ_i must be upper-bounded by a constant which depends on the value $\max\{-\frac{1}{2}, -\frac{\log_\alpha(\alpha-1)}{2}\}$. Thus, $c_i^{(2)}$ allows us to deal with any type set Θ_i that is upper-bounded by any constant $\sup_i$.

In order to have truthful mechanism for the problem at hand, involving agents having type set upper-bounded by a finite value, we round the bids. But what about the approximation? If A is a c-approximation algorithm, then nothing can be said about the approximation of A_α. Next, we define a class of problems for which the loss in approximation guarantee of A due to the rounding is bounded. Henceforth, we restrict our attention only to minimization problems. We stress that similar arguments can be applied for maximization problems as well.

Definition 15. Fix $\epsilon > 0$ and $\delta > 1$. A one-parameter minimization problem $\Pi = (\mathcal{I}, \mathsf{m}, \mathsf{sol}, \min)$ is (δ, ϵ)-*smooth* if, for any pair of instances $I = (T, \sigma)$ and $\tilde{I} = (\tilde{T}, \sigma)$ such that $t_i \leq \tilde{t}_i \leq \delta \cdot t_i$ for $i = 1, 2, \ldots, m$, and for all $S \in \mathsf{sol}(\sigma)$, it holds that

$$\mathsf{m}(S, I) \leq \mathsf{m}(S, \tilde{I}) \leq (1 + \epsilon) \cdot \mathsf{m}(S, I).$$

For instance, observe that $Q \parallel C_{\max}$ is $(\alpha, \alpha - 1)$-smooth for all $\alpha > 1$. From the above definition, the following remark is straightforward.

Remark 16. Let Π be a (δ, ϵ)-smooth one-parameter minimization problem and let $I = (T, \sigma)$ and $\tilde{I} = (\tilde{T}, \sigma)$ be two instances of Π such that $t_i \leq \tilde{t}_i \leq \delta \cdot t_i$, for $i = 1, 2, \ldots, m$. Then, any c-approximate solution S for I is $c \cdot (1+\epsilon)$-approximate for $\tilde{I}$ and any c-approximate solution $\tilde{S}$ for $\tilde{I}$ is $c \cdot (1 + \epsilon)$-approximate for I.

From Theorem 14 and the above remark we have the next theorem.

Theorem 17. *Let Π be a $(\alpha, \alpha - 1)$-smooth optimization problem for verifiable one-parameter selfish agents whose types are positive real and let A be polynomial-time c-approximate weakly-monotone algorithm for Π. If every Θ_i is upper-bounded by a finite value $\sup_i$, then $\mathcal{M} = (A_\alpha, P^{(2)})$ is $(\alpha \cdot c)$-approximate polynomial-time truthful mechanism with verification for Π, satisfying voluntary participation condition.*

4. Applications

Although our payments are very simple, they have the nice property of being computable in constant time (given the reading of the input: the m bids and the n weights). This efficiency turns out to be very useful in many problems studied in literature. In particular, graph problems (and more in general binary demand games) seem to have specific advantages from our schemes. Indeed, in Section 4.2 we will give a specific application to the so-called *Minimum Radius Spanning Tree*(MRST) studied in [20]. Same reasoning is extended to several binary demand games studied in literature — see [13,18]).

While very efficient, our payment schemes have the limitation to require upper-bounded types. We will see that in classical problems, such as $Q \parallel C_{\max}$, this hypothesis can be overcome. This issue (along with applications of this result) will be discussed in Section 4.1. It turns out that our payment schemes can be easily generalized to deal with an online version of $Q \parallel C_{\max}$ problem. Following the approach in [6] we discuss this adaptation in Section 4.1.1.

4.1. $Q \parallel C_{\max}$ problem

In this section we give a non-trivial application of our results to the well known $Q \parallel C_{\max}$ problem. In the case in which type sets are discrete, then given a c-approximate polynomial-time weakly-monotone algorithm for $Q \parallel C_{\max}$ problem, we can construct c-approximate polynomial-time truthful mechanism (with verification) for $Q \parallel C_{\max}$. On the other hand, when we have no constraints on the type sets, then given a c-approximate polynomial-time weakly-monotone algorithm for $Q \parallel C_{\max}$ problem, we can construct $c \cdot (1 + \epsilon)$-approximate polynomial-time truthful mechanism with verification for $Q \parallel C_{\max}$, for every $\epsilon > 0$. Moreover, our possibility results will be enforced showing a polynomial-time algorithm for $Q \parallel C_{\max}$ that is weakly-monotone (and not monotone) for all the type sets.

We refer to Section 1 for the definition of the problem. In the $Q \parallel C_{\max}$ problem with verifiable one-parameter selfish agents,[2] the machines are owned by verifiable selfish agents wishing to maximize their own profit (as discussed in Section 1) disregarding the global makespan minimization. In particular, the job weights $W = (w_1, \ldots, w_n)$, are the public part of the input, and the machine speeds are the private part of the input, that is, each agent i privately knows the speed of her machine. As usual, we assume that the types of the agents are the inverse of the speed.

As shown in Theorems 11 and 14, to apply our payment schemes, type sets must be upper-bounded by a finite value. For $Q \parallel C_{\max}$ problem, since types are the inverse of the speeds, it means that the speed of every agent (the inverse of the

[2] In the rest of the paper, with an abuse of notation, we will simply call $Q \parallel C_{\max}$ problem the one with verifiable one-parameter selfish agents since here we deal only with the latter.

declared bid) has to be lower-bounded by a constant greater than 0, but this could not be the case. Therefore, here we show a method to deal with these cases. We show that it is always possible to reduce any instance of $Q \parallel C_{\max}$ to the one where every type set is upper-bounded by a finite value preserving the optimum of the instance.

The idea is to give a lower bound on the speed (an upper bound on the declared bid) to each agent depending on the declaration of the other agents. Thus, if an agent declares a speed value too small with respect to the other declared speeds, then she can be discarded. More formally, let

$$\hat{s}_i = \begin{cases} \frac{1}{x} & \text{if } \Theta_i \text{ is upper bounded by a value } x > 0 \\ \frac{\hat{w}}{\text{time}_i} & \text{otherwise} \end{cases}$$

where $\hat{w}$ is a maximum weight job and time_i is defined as follows. Let $k(i)$ be a fastest machine in $\{1, \ldots, m\}\backslash\{i\}$ w.r.t. the bid vector B (that is a machine with smallest bid without considering machine i), then time_i is the time taken by agent $k(i)$ to execute all jobs according to her bid $b_{k(i)}$: $\text{time}_i = \mathcal{W} \cdot \text{b}_{\text{k(i)}}$.

Observe that if the machine i declares $b_i > \frac{1}{\hat{s}_i}$, then for any optimum solution OPT, $\text{w}_i(OPT) = 0$, since machine $k(i)$ requires less time to execute all jobs, than the time needed to machine i to complete any job.

Let A be a weakly-monotone algorithm for $Q \parallel C_{\max}$ problem. Now, we describe a weakly-monotone algorithm A' for $Q \parallel C_{\max}$, which uses A as a subroutine. Algorithm A' has the same approximation ratio of A and can be used to deal with machines having unbounded speeds. It takes as input the bid vector B and the weight vector W and outputs a schedule S.

1. Let $\hat{B}$ be the bids vector B without the machines bidding $b_i > \frac{1}{\hat{s}_i}$; let $\hat{S}$ be the schedule returned by A executed on $\hat{B}$ and W;
2. Let S be a schedule equal to $\hat{S}$ for all machines declaring $b_i < \frac{1}{\hat{s}_i}$ assigning 0 to all the other machines; return S as the schedule.

Now, we show that this algorithm, together with our payment schemes, leads to a truthful mechanism with verification. Moreover, it has the same approximation ratio of algorithm A.

Lemma 18. *If A is a weakly-monotone c-approximate algorithm for $Q \parallel C_{\max}$ problem, then A' is a weakly-monotone c-approximate algorithm for $Q \parallel C_{\max}$ problem.*

Proof. We first show that A' is a scheduling algorithm. Since A is a scheduling algorithm, we only have to show that at least one machine is given as input to algorithm A. Now, we show that we never discard the fastest machines. Let i be a fastest machine in $\{1, \ldots, m\}$ and k be a fastest machine in $\{1, \ldots, m\}\backslash\{i\}$. Then, obviously $s_k = \frac{1}{b_k} \leq \frac{1}{b_i} = s_i$. Let T be the time needed by machine k to execute all jobs and let w be a smallest job. From the definition of $\hat{s}_i$, we have $\hat{s}_i = \frac{w}{T} = \frac{w}{\hat{w}} \cdot s_k \leq s_k \leq s_i$. This implies that the fastest machines are surely not discarded. We now prove that A' is weakly-monotone. Fix a bid vector B, and suppose that $\text{w}_i(A'(B, W)) = 0$. We prove that $\text{w}_i(A'((b', B_{-i}), W)) = 0$, for every $b' \geq b_i$. In the case $b' > \frac{1}{\hat{s}_i}$, trivially $\text{w}_i(A'((b', B_{-i}), W)) = 0$, since machine i will be discarded. If $b'_i \leq \frac{1}{\hat{s}_i}$, then machine is not discarded and $\text{w}_i(A'((b', B_{-i}), W)) = 0$, given that algorithm A is weakly-monotone. Finally, to show that the algorithm A' is a c-approximate algorithm, we only prove that the deletion of the "slowest" machines does not modify the optimum. More specifically, let I be the initial instance of the problem and OPT be an optimum solution for I. If $b_i > \frac{1}{\hat{s}_i}$ (i.e. machine i is a discarded machine), then $\text{w}_i(OPT) = 0$. In fact, the time needed by the machine i to complete the smallest job w is greater then the time needed to the fastest machine to complete the overall jobs. □

Algorithm A', using the bounds $\hat{s}_i$, reduces any (potentially unbounded) instance I of $Q \parallel C_{\max}$ to a bounded instance $\hat{I}$ of $Q \parallel C_{\max}$. Thus, we can apply our payment schemes. By Lemma 18 and Theorem 12 we have the following.

Theorem 19. *Let A be a c-approximate polynomial-time weakly-monotone algorithm for $Q \parallel C_{\max}$ problem. If every Θ_i is discrete w.r.t. a known value Δ_i, then there exists a c-approximate polynomial-time truthful mechanism with verification $\mathcal{M} = (A', P^{(1)})$ for $Q \parallel C_{\max}$, satisfying voluntary participation condition.*

By Lemma 18, Theorem 17 and since $Q \parallel C_{\max}$ problem is $(1 + \epsilon, \epsilon)$-smooth we have the following.

Theorem 20. *Let A be a c-approximate polynomial-time weakly-monotone algorithm for $Q \parallel C_{\max}$ problem. Then, for any $\epsilon > 0$, there exists a $c \cdot (1+\epsilon)$-approximate polynomial-time truthful mechanism with verification $\mathcal{M} = (A', P^{(2)})$ for $Q \parallel C_{\max}$, satisfying voluntary participation condition.*

Next, we show that a polynomial-time approximation algorithm for $Q \parallel C_{\max}$ that is weakly-monotone for all the possible type sets actually exists, so having that our possibility results are useful in practice. We highlight that a $(1+\epsilon)$-approximating weakly-monotone algorithm (for infinite type sets) has been given in [3]. Such an algorithm runs in polynomial-time only when the number of agents is constant. We go beyond by showing a polynomial-time weakly-monotone algorithm for the domains we are considering. In particular, we consider the following greedy algorithm for $Q \parallel C_{\max}$ having a (4/3)-approximation ratio (see [11]) that we call *MkSpan*[3]:

[3] Let us recall that in [5] the authors show a PTAS for the $Q \parallel C_{\max}$. Such an algorithm is weakly-monotone for integer speeds that are upper bounded by a constant. This implies that the type set for which their algorithm is weakly-monotone is finite. Thus, it cannot be used here to show that our mechanisms are useful in practice for all the possible type sets.

1. sort the jobs in non-increasing order;
2. assign, in this order, each job to the machine that completes the job in the minimum time.

Theorem 21. *MkSpan is a polynomial-time weakly-monotone approximation algorithm for $Q \parallel C_{\max}$.*

Proof. The proof is trivial. If an agent i does not receive any job when she declares b_i, then it is obvious that she does not receive any job if she declares $b'_i > b_i$ since any job is assigned, in non-increasing order, to the machine that completes it in the minimum time. □

Theorems 21 and 20 imply the following corollary.

Corollary 22. *For any $\epsilon > 0$, there exists a $\frac{4}{3} \cdot (1+\epsilon)$-approximate polynomial-time truthful mechanism with verification $\mathcal{M} = (MkSpan, P^{(2)})$ for $Q \parallel C_{\max}$, satisfying voluntary participation condition.*

If type sets are discrete, using Theorems 21 and 19, we can attain a 4/3-approximate truthful mechanism with verification.

4.1.1. Online version of $Q \parallel C_{\max}$

In this section we study the online version of the makespan problem following the approach in [6]. In the online version of $Q \parallel C_{\max}$, jobs arrive one-by-one and must be scheduled upon their arrival. Moreover, jobs cannot be reallocated. For any (possibly infinite) sequence of jobs $J = J_1 J_2 \ldots$, we let J^k denote the prefix $J_1 J_2 \ldots J_k$ of the first k jobs and we let $w(J_k)$ denote the weight of job J_k, for $1 \leq k \leq |J|$. All notations used so far naturally extend in this case by including the job prefix J^k as input (we stress that, in order to ease our notation, we do not include public information σ in this section).

Before any job appears, each agent declares her type and we denote by $B = (b_1, \ldots, b_m)$ the vector of declared types. An online mechanism for $Q \parallel C_{\max}$ is a pair $\mathcal{M} = (A, P)$ where P is a sequence of payment functions P_i^k, for $i = 1, \ldots, m$ and $k > 0$ such that

- The algorithm A is an online algorithm for $Q \parallel C_{\max}$: that is, A on input J^k and vector B of declared types assigns job J_k to some machine i. We denote, for any $1 \leq j \leq k$, $\mathcal{W}^j = \sum_{l=1}^{j} w(J_l)$ and we set $\mathcal{W}^0 = 0$.
- When the k-th job arrives, it is assigned by A to some machine. Basing on the observed release time of job J_k, that we denote to as $r(J_k)$, the mechanism awards each agent i a non-negative payment $P_i^k(B, J^k | r(J_k))$ if and only if the observed release time is consistent with the declarations. That is, we are not allowed to ask money back from the agents. We call payment functions with such property *PayMon*.

The total payment received by agent i after k jobs is $P_i(B, J^k | r(J^k)) = \sum_{j=1}^{k} P_i^j(B, J^j | r(J_j))$, where $r(J^k) = (r(J_1), \ldots, r(J_k))$.

Definition 23 (*[6]*). We say that an online mechanism is truthful with verification if for any prefix J^k of J, for all B_{-i}, and for all types t_i, the function $\text{profit}_i((b_i, B_{-i}), J^k, t_i | r(J^k))$ is maximized for $b_i = t_i$.

The problem of designing online truthful mechanisms for the $Q \parallel C_{\max}$ problem has been addressed in [6] both for not verifiable and verifiable machines. In the latter case, the authors provide an online weakly-monotone algorithm (an online algorithm is weakly-monotone if it is weakly-monotone w.r.t. *any* job prefix) with competitive ratio 12. They provide a *PayMon* payment scheme. The computation of their payments is possible only for finite type sets and is polynomial if and only if every type set has a polynomial size. We thus augment their online weakly-monotone algorithm with an efficient payment scheme. We thus obtain a more efficient 12 competitive online truthful mechanism with verification. The payment scheme we define is an online generalization of our payment schemes (next definition deals with integer speeds although it is possible to generalize it through constants and rounding as we did for our offline payments).

Definition 24. Let B be a bid vector, J^k a job prefix for a given k, $r(J_k)$ be the observed completion time of job J_k. Given an algorithm A, for each $i = 1, \ldots, m$, we define

$$\tilde{P}_i^k(B, J^k | r(J_k)) := \begin{cases} \frac{\mathcal{W}^k - \mathcal{W}^{\ell_i(k)}}{b_i} & \text{if } J_k \text{ is assigned to } i \text{ and } r(J_k) = w(J_k) \cdot b_i; \\ 0 & \text{otherwise,} \end{cases}$$

where $\ell_i(k) = \text{argmax}_{k' < k}\{A(B, J^{k'})$ assigns job $J_{k'}$ to machine $i\}$ if k is not the first job assigned by A to machine i and 0 otherwise.

Since job weights are non-negative it is easy to see that the scheme above is *PayMon*. Moreover, a simple adaptation of the proof of Theorems 11, 14 shows that payment above leads to truthful mechanisms with verification for the online version of $Q \parallel C_{\max}$ problem. We thus obtain the following result.

Theorem 25. *The $Q \parallel C_{\max}$ problem with verifiable machines admits an online truthful polynomial-time mechanism which is 12-competitive. The running time of this mechanism matches the one of the online weakly-monotone algorithm given in [6].*

4.2. Graph problems and binary demand games

Consider problems in which we are given a graph $G = (V, E)$ and the set of feasible outcomes consists of a suitable set containing certain subgraphs of G. We have one agent per edge and the type t_e of agent e is nothing but the *weight* of edge $e \in E$. If subgraph X of G includes 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 [17], minimum spanning tree [16], shortest-path tree [10], minimum-radius spanning tree [20]. Observe that, as already pointed out in [3], in mechanism design graph problems, we deal with a special case of one-parameter agents in which $w_e(X) = 1$ if edge e is used by X, and $w_e(X) = 0$ otherwise. Hence, an algorithm A is weakly-monotone if and only if A is monotone according to the following definition:

Definition 26 *([7]).* An algorithm A is *monotone* for one-parameter agents if, for all agents e, and for all b_e, B_{-e} and $b'_e < b_e$, it holds that $\mathsf{w}_e(A(b_e, B_{-e})) \leq \mathsf{w}_e(A(b'_e, B_{-e}))$.

Recall that, as stated in [7], a mechanism for one-parameter agent (without verification) is truthful if and only if the used algorithm is monotone. Thus, extant truthful mechanisms *without* verification for graph problems can be turned in truthful mechanisms *with* verification (using the same algorithm) simply relaxing the model (i.e., rewarding agents after the execution of the work). Is there any advantage? One could say: why do you relax the model if we are able to solve the problem in a more general setting? The answer is: efficiency. Our payment schemes are very efficient and can speed up time and space complexities of existing mechanisms as we are going to show for the *Minimum Radius Spanning Tree* studied in [20].

Minimum radius spanning tree [20]. The set of feasible outcomes consists of all spanning trees over a given graph $G = (V, E)$. The goal is to find a rooted tree of minimum height with respect to the edge weights t_e, where $e \in E$. That is, a tree T_u rooted at some $u \in V$ such that the longest path from any v to the root u is as small as possible (the length of a path is the sum of all edge weights t_e of edges e in that path). As observed by the authors of [20], this problem is *not* utilitarian (i.e., the objective function is not the sum of the valuations[4] of the agents) and some algorithms which solve this problem exactly may not lead to truthful mechanisms. The authors then provide a polynomial-time algorithm by sticking to a particular way of computing the optimal solutions which breaks ties in a fixed manner. Let SPT_u be the algorithm computing a shortest-path tree of G rooted at $u \in V$. The minimum-radius spanning tree can be obtained by selecting the best, over all $u \in V$, of the solution returned by SPT_u. It is very simple to show that the algorithm is monotone: when the agent of a selected edge decreases her weight, she continues to be selected; conversely, when the agent of a non-selected edge increases her weight then she is still non-selected). The hard core is in showing that the payments (which are completely determined in [7]) are computable in polynomial time. Indeed, the authors of [20] show that the mechanism is computable in $O(mn\sqrt{n} + n^3 \log n)$ time and $O(n^2\sqrt{n})$ space where n is the number of nodes and m denotes the number of edges of G. Since the fastest centralized algorithm for computing an MRST requires $O(mn + n^2 \log n)$ time and $O(n^2)$ space [9], it follows they pay in terms of complexity an $O(n)$ factor for time and an $O(\sqrt{n})$ factor for space, respectively. The interesting fact is that their algorithm is computed in $O(mn + n^2 \log n)$ time and $O(n^2)$ space which are exactly the complexities of the fastest centralized algorithm. In other words their losses in performance are completely due to the payments computation. Since the problem involves verifiable one-parameter selfish agents and since a monotone algorithm is weakly-monotone as well we can use our payment schemes. The next result simply follows from the results for the payment scheme $P^{(1)}$:

Theorem 27. *If the type sets are discrete and upper-bounded then there exists a polynomial-time exact truthful mechanism with verification $\mathcal{M}$ for the non-utilitarian MRST problem. Moreover, $\mathcal{M}$ is computable in $O(mn + n^2 \log n)$ time and $O(n^2)$ space and $\mathcal{M}$ satisfies voluntary participation.*

Our mechanism uses the algorithm proposed in [20] and payment $P^{(1)}$. Thus, verification improves time and space complexities of the mechanism to the ones of the fastest centralized algorithm. The next result simply follows from the results for the payment scheme $P^{(2)}$ and from the fact that MRST is $(1 + \epsilon, \epsilon)$-smooth:

Theorem 28. *If the type sets are upper-bounded then, for any ϵ, there exists a polynomial-time $(1 + \epsilon)$-approximate truthful mechanism with verification $\mathcal{M}$ for the non-utilitarian MRST problem. Moreover, $\mathcal{M}$ is computable in $O(mn + n^2 \log n)$ time and $O(n^2)$ space and $\mathcal{M}$ satisfies voluntary participation.*

Since we can model[5] the agents of this game as KSM bidders (see [15]) we have that, for any agent, there exists a critical (valuation) value v_C. If the bid on agent e has an associated valuation greater than v_C then e is in the solution, while for valuations less than v_C agent e is out of game. Thus, modeling the agents as KSM bidders, we have that such a finite[6] value

[4] The valuation of agent i, for the solution X, is $-\mathsf{cost}_i(X, t_i)$. This implies defining the profit (for a given solution X) as the received payment *plus* the valuation for that solution.

[5] Indeed, here the valuation can be defined as follows: $-t_e$ if e is in the solution, 0 otherwise. Recall that the weight of agent e, for a given tree, is 1 when e is in the tree and 0 otherwise. See [3] for more details.

[6] Since no agent is indispensable then the graph is 2-connected. Thus, there is no agent included in the solution no matter what she declares, i.e., v_C is finite. See also [20].

Table 2
The price of trust for several binary demand games studied in literature. With n we denote the number of nodes of the graph in input to MRST and MDST problems, while ρ_7 and ρ_8 denote the running time of Algorithm 7 and Algorithm 8 in [13]. Recall that $\alpha(\cdot,\cdot)$ is the classic inverse of the Ackermann's function.

Problem	Price of trust
MRST [20]	$O(n)$
MDST [18]	$O(n\alpha(n,n))$
Set Cover [13]	$O(\rho_7)$
Link Weighted Steiner Tree [13]	$O(\rho_8)$

exists and as it upper-bounds the types (used in any solution) we could use it to deal with unbounded instances (by removing the hypothesis of bounded types). The problem is in computing these values. Actually, the authors of [20] need to compute them, paying a linear factor in the time complexity of the entire mechanism. However, we can use, as an upper-bound for the types, weights encoding useless (or non-existing) links (e.g., in the modern high bandwidth Internet, links slower than 56-kbps are out of date).

Binary demand games. The MRST problem we analyzed above is a special case of binary demand games (see, e.g., [13]). This is a class of games in which every agent has two possible outcomes: being selected or not being selected. For such a class of games, monotone algorithms correspond to weakly-monotone ones and therefore we can use the efficiency of our payment schemes to improve existing truthful mechanisms. In particular, we define the *Price of Trust (PoT)* which measures the price one has to pay to award payments before the execution of the work, thus "trusting" agents' declarations. PoT is defined as the ratio between the best (known) running time of a truthful mechanism without verification and the best (known) running time of a truthful mechanism with verification. Since from any mechanisms without verification (A, P) we can obtain a mechanism with verification using A and our payment schemes, in the PoT definition we don't care about approximation guarantee of the mechanisms. Indeed, their approximation guarantee can differ for at most a small factor $\epsilon > 0$ (if mechanism with verification uses our payment scheme $P^{(2)}$ for smooth problems).

Several problems belonging to the class of binary demand games have been studied in literature: MRST we discussed above, minimum diameter spanning tree [18], set cover and link weighted Steiner tree [13]. Table 2 summarizes the PoT for these problems. We stress that all the best (known) truthful mechanisms with verification use the same algorithm of the classical mechanisms but they use our payment schemes. For the first two rows we also remark that the the running time of mechanisms with verification matches the one of the best non-strategic algorithm (a non-strategic algorithm is a classical algorithm knowing all the input). Notice that by means of our payment schemes, truthful mechanisms with verification gain at least a linear factor with respect to the classical ones (Algorithm 7 and Algorithm 8 in [13] are the algorithms used to solve the corresponding problems). These results suggest that, in order to obtain payments leading to truthful mechanisms without verification, one has at least to read the input of the problem at hand.

5. Conclusions and open problems

We propose new payment schemes for mechanisms with verification. Our results show that weak-monotonicity suffices in more general settings w.r.t. the ones proposed in [5]. Indeed, using weakly-monotone algorithms, we are able to obtain truthful mechanisms with verification for problems in which the types are discrete and upper-bounded. Moreover, weak-monotonicity also suffices in the case in which types are continuous (but upper-bounded). Recall that in [5], it was showed that weak-monotonicity suffices *just* for finite type sets. Our main contribution is in finding a constant-time computable payment function which, in some sense, strengthen the notion of power of verification introduced in [5]. Indeed, we stress the following important points.

Easier payments. Our payments are easy to understand and more importantly to compute: they just need the reading of the input and one computes the payment of an agent in one step. This reduces the problem of finding a truthful mechanism with verification to the problem of designing a weakly-monotone algorithm. On the contrary, designing truthful mechanism (without verification) one has to design a monotone algorithm (in general a harder task than designing a weakly-monotone algorithm) and has to use the payments of [7]. The core is that these payments may be difficulty to compute. Then, much work is needed in showing that the payments are actually computable in polynomial-time.

Preserving algorithms. As we have seen for the MRST (and binary demand games in general) computing payments for truthful mechanisms can also make worse time (and space) complexity of the algorithmic part. Since our payments are constant-time computable we are able to preserve the running time of the algorithm used by a mechanism with verification. Moreover, since a monotone algorithm is weakly-monotone as well, one can also reconsider all existing mechanisms in order to improve their performances (as we have done for the MRST problem). In a truthful

mechanism with verification we are able to preserve, not only the running time of the used algorithm, but also the approximation guarantee of it in many real-life applications.

As we have seen the hypothesis of upper-bounded types can be overcome in natural problems. Nevertheless, it would be interesting finding a payment scheme leading to truthful mechanisms with verification for continuous types (removing the hypothesis of upper-bounded types). Moreover, for mechanisms with verification, it seems there is a trade-off between cheapness and efficiency of the payment schemes. Indeed, the payments introduced in [5] are very cheap (in some sense they are optimal) but not efficient. On the other hand, our payments are efficient but very expensive. Is there a relationship? It would be nice to have an answer to this question.

Acknowledgments

We wish to thank the authors of [5] for providing us with a preliminary full version of their paper. Moreover, we wish to thank Paolo Penna for enlightening discussions and valuable advises.

Fourth author's research was supported by the European Project FP6-15964, Algorithmic Principles for Building Efficient Overlay Computers (AEOLUS) and by DFG grant Kr 2332/1-2 within Emmy Noether Program.

References

[1] N. Andelman, Y. Azar, M. Sorani, Truthful approximation mechanisms for scheduling selfish related machines, in: Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science, STACS, in: LNCS, 2005, pp. 69–82.
[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and Approximation — Combinatorial Optimization Problems and Their Approximability Properties, Springer, 1999.
[3] V. Auletta, R. De Prisco, P. Penna, G. Persiano, C. Ventre, New constructions of mechanisms with verification, in: Proceeding of the 33rd International Colloquium on Automata, Languages and Programming, ICALP, in: Lecture Notes in Computer Science, vol. 4051, Springer, 2006, pp. 596–607.
[4] V. Auletta, R. De Prisco, P. Penna, P. Persiano, Deterministic truthful mechanisms for scheduling on selfish machines, in: Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science, STACS, in: LNCS, vol. 2996, 2004, pp. 608–619.
[5] V. Auletta, R. De Prisco, P. Penna, P. Persiano, The power of verification for one-parameter agents, in: Proceedings of the 31st International Colloquium on Automata, Languages and Programming, ICALP, in: LNCS, vol. 3142, 2004, pp. 171–182.
[6] V. Auletta, R. De Prisco, P. Penna, G. Persiano, On designing truthful mechanisms for online scheduling, in: Proc. of the 12th International Colloquium on Structural Information and Communication Complexity, SIROCCO, in: Lecture Notes in Computer Science, vol. 3499, Springer, 2005, pp. 3–17.
[7] A. Archer, E. Tardos, Truthful mechanisms for one-parameter agents, in: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, FOCS, 2001, pp. 482–491.
[8] E.H. Clarke, Multipart pricing of public goods, Public Choice (1971) 17–33.
[9] D. Dvir, G.H. Handler, The absolute center of a network, Networks 43 (2) (2004) 109–118.
[10] L. Gualà, G. Proietti, Efficient truthful mechanisms for the single-source shortest paths tree problem, in: Proc. of Euro-Par, in: Lecture Notes in Computer Science, vol. 3648, 2005, pp. 941–951.
[11] R.L. Graham, Bounds for certain multiprocessing anomalies, Bell System Technical Journal 45 (1966) 1563–1581.
[12] R.L. Graham, Bounds on multiprocessing timing anomalies, SIAM Journal on Applied Mathematics 17 (2) (1969).
[13] M. Kao, X. Li, W. Wang, Towards truthful mechanisms for binary demand games: A general framework, in: ACM Conference on Electronic Commerce, ACM, 2005, pp. 213–222.
[14] Elias Koutsoupias, Angelina Vidali, A lower bound of $1 + \Phi$ for truthful scheduling mechanisms, in: Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, in: Lecture Notes in Computer Science, vol. 4708, Springer, 2007, pp. 454–464.
[15] A. Mu'Alem, N. Nisan, Truthful approximation mechanisms for restricted combinatorial auctions, in: Proc. of 18th National Conference on Artificial intelligence, AAAI, 2002, pp. 379–384.
[16] N. Nisan, A. Ronen, Computationally feasible VCG mechanisms, in: Proceedings of the 2nd ACM Conference on Electronic Commerce, EC, 2000, pp. 242–252.
[17] N. Nisan, A. Ronen, Algorithmic mechanism design, Games and Economic Behavior 35 (2001) 166–196. Extended abstract in the Proc. of the 31st Annual ACM Symposium on Theory of Computing, STOC, 1999, pp. 129–140.
[18] P. Penna, G. Proietti, P. Widmayer, Polynomial-time truthful mechanisms in one shot, in: Proc. of the 2nd Workshop on Internet and Network Economics, WINE, in: Lecture Notes in Computer Science, vol. 4286, Springer, 2006, pp. 377–388.
[19] P. Penna, C. Ventre, Collusion-resistant mechanisms with verification yielding optimal solutions, in: Proc. of the 16th Annual European Symposium on Algorithms, ESA, in: Lecture Notes in Computer Science, vol. 5193, Springer, 2008, pp. 708–719.
[20] G. Proietti, P. Widmayer, A truthful mechanism for the non-utilitarian minimum radius spanning tree problem, in: Proceedings of the 17th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA, ACM Press, 2005, pp. 195–202.
[21] V. Vazirani, Approximation Algorithms, Springer, 2001.
[22] C. Ventre, Mechanisms with verification for any finite domain, in: Proc. of the 2nd Workshop on Internet and Network Economics, WINE, in: Lecture Notes in Computer Science, vol. 4286, Springer, 2006, pp. 37–49.
[23] W. Vickrey, Counterspeculation, auctions, and competitive sealed tenders, Journal of Finance 16 (1961) 8–37.