The Power of Verification for One-Parameter Agents*

Vincenzo Auletta, Roberto De Prisco, Paolo Penna, and Giuseppe Persiano

Dipartimento di Informatica ed Applicazioni "R.M. Capocelli", Università di Salerno, via S. Allende 2, I-84081 Baronissi (SA), Italy.
{auletta,robdep,penna,giuper}@dia.unisa.it

Abstract. We study combinatorial optimization problems involving one-parameter selfish agents considered by Archer and Tardos [FOCS 2001]. In particular, we show that, if agents can lie in one direction (that is they either overbid or underbid) then *any* (polynomial-time) c-approximation algorithm, for the optimization problem without selfish agents, can be turned into a (polynomial-time) $c(1+\epsilon)$-approximation truthful mechanism, for any $\epsilon > 0$. We then look at the $Q||C_{\max}$ problem in the case of agents owning machines of different speeds. We consider the model in which payments are given to the agents only after the machines have completed the jobs assigned. This means that for each machine that receives at least one job, the mechanism can *verify* if the corresponding agent declared a greater speed. For this setting, we characterize the allocation algorithms A that admit a payment function P such that $M = (A, P)$ is a truthful mechanism. In addition, we give a $(1+\epsilon)$-approximation truthful mechanism for $Q||C_{\max}$ when machine speeds are bounded by a constant. Finally, we consider the classical scheduling problem $Q||\sum w_j C_j$ which does not admit an exact mechanism if verification is not allowed. By contrast, we show that an exact mechanism for $Q||\sum w_j C_j$ exists when verification is allowed.

1 Introduction

Algorithms for solving optimization problems have been studied for decades in several models and 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 the Internet. Here, the various parts of the input are owned by *selfish* (but *rational*) agents and thus the optimization algorithm will have to ask the agents and then work on the *reported* inputs. It is realistic to assume that an agent will lie about her input if this leads to a solution X that she prefers even in spite of the fact that X is not globally optimal.

The field of mechanism design is the branch of Game Theory and Microeconomics that studies ways of inducing the agents to report their true type so

* Work supported by the European Project IST-2001-33135, Critical Resource Sharing for Cooperation in Complex Systems (CRESCCO).

J. Diaz et al. (Eds.): ICALP 2004, LNCS 3142, pp. 171–182, 2004.

that the optimization problem can be solved on the real input. We consider the following conceptual scenario with m agents identified by the integers $1, \cdots, m$. Each agent i has a private type t_i and a public valuation function v_i such that $v_i(X, t_i, \sigma)$ measures how much agent i likes solution X for instance σ. A *mechanism* is a pair $M = (A, P_A)$. A is an algorithm that, on input an instance σ and the reported types $b = (b_1, b_2, \ldots, b_n)$ (b_i is the type reported by agent i), computes a solution $X = A(b, \sigma)$. Moreover, the mechanism awards to each agent i a payment $P_A^i(b, \sigma)$, where $P_A = (P_A^1, \cdots, P_A^m)$. We define the *utility* (or *profit*) $u_M^i(b, \sigma|t_i)$ of agent i on instance σ, when the type of agent i is t_i and $b = (b_1, \cdots, b_m)$ are the declared values of the m agents in the following way

$$u_M^i(b, \sigma|t_i) := P_A^i(b, \sigma) + v^i(X, t_i, \sigma).$$

Each agent *knows* both algorithm A and payment function P_A^i and, being selfish and rational, naturally aims at maximizing her utility u^i. A mechanism is said *truthful with dominant strategies* (or simply *truthful*) if the payments P_A and the algorithm A guarantee that no agent obtains a larger utility when reporting $b_i \neq t_i$, independently of the other agents' reported types; that is, for all instances σ and for all reported types $b_{-i} = (b_1, \cdots, b_{i-1}, b_{i+1}, \cdots, b_m)$ of all the agents except i and for all possible declarations b_i of agent i it holds that

$$u_M^i((t_i, b_{-i}), \sigma|t_i) \geq u_M^i((b_i, b_{-i}), \sigma|t_i).$$

In what follows when M, σ, b_{-i} and t_i are clear from the context we will simply say $u^i(x)$ to denote the utility obtained by agent i by declaring x.

The celebrated truthful VCG mechanisms [6,9,5] are one of the classical results about mechanism design. Unfortunately, the VCG paradigm only applies to so-called *utilitarian* problems. The problem of scheduling jobs on related machines owned by selfish agents in order to minimize the makespan is a natural example of a non-utilitarian problem to which mechanism design theory can be applied. Here, the type t_i of agent i is the inverse of the speed s_i of her machine and a scheduling X has valuation $v^i(X, t_i, \sigma) = -w^i(X) \cdot t_i$ where $w^i(X)$ is the load assigned by scheduling X to machine i. The goal is to design a truthful mechanism that computes a scheduling with a good makespan. Scheduling is a special case of a large class of optimization problems for selfish agents called *one-parameter* agents. Efficient mechanisms for this class of problems have been provided by Archer and Tardos [1] that also characterized the class of allocation algorithms A that admit payments functions P for which (A, P) is a truthful mechanism. Essentially, truthful mechanisms for one-parameter agents must use so called *monotone* algorithms and, in this case, their payment functions are uniquely determined (up to an additive factor).

Summary of results. In this work, we consider mechanisms for one-parameter agents. We start by considering the case in which agents cannot lie arbitrarily, but either can only overbid (i.e., report $b_i \geq t_i$) or can only underbid (i.e., report $b_i \leq t_i$). We call such agents *restricted* agents. We prove that, in this case, for *every* algorithm A there exist payment functions P_A such that $M = (A, P_A)$ is a truthful mechanism (see Theorem 1). This is in contrast with the case of

(unrestricted) one-parameter selfish agents in which only *monotone* algorithms admit payments [1]. We also show that, under mild assumptions about the optimization problem (that, for example, hold for scheduling related machines), *any* c-approximation polynomial-time algorithm A and any $\epsilon > 0$, there exists a polynomial-time algorithm A_ϵ and polynomial-time computable functions $P_{A_\epsilon} = (P^1_{A_\epsilon}, \cdots, P^m_{A_\epsilon})$ such that A_ϵ is $c(1+\epsilon)$-approximate and $(A_\epsilon, P_{A,\epsilon})$ is a truthful mechanism (see Theorem 2).

We then consider *verifiable* agents, that is agents that may lie in reporting their types but the mechanism can verify whether agent i underbids, provided that the work $w^i(X)$ assigned to agent i by the solution X is positive. For scheduling, this naturally models the situation in which it is possible to verify that a machine has underbid (*i.e.*, it has declared to be faster than it actually is) only if at least one job has been assigned to it. This model has been studied by Nisan and Ronen [8] for the case of scheduling unrelated machines. We show that an algorithm A admits payments P_A so that (A, P_A) is truthful for verifiable agents *if and only if* A is *weakly monotone* (see Definition 3) and show that for polynomial-time weakly monotone algorithms the payment functions can be computed in polynomial time (see Theorem 3). This result tells us that the case of verifiable one-parameter selfish agents lies in between the case of restricted selfish agents (for which, by Theorem 2, all algorithms admit a payment function) and the case of (unrestricted) selfish agents in which only a subclass of the class of weakly monotone algorithms (called monotone algorithms and introduced in [1]) admit a payment function. Based on the characterization described above, in Section 4, we present a $(1+\epsilon)$-approximation truthful mechanism for $\mathrm{Q}||\mathrm{C}_{\max}$ where machines are owned by verifiable agents with speeds bounded by a constant. The algorithm is polynomial for any number of machines. This result should be contrasted with the $(3+\epsilon)$-approximation randomized mechanism truthful in expectation [1] and the $(4+\epsilon)$-approximation deterministic truthful one [2] for *unverifiable* agents.

Finally, we consider the classical scheduling problem $\mathrm{Q}||\sum \mathrm{w}_j\mathrm{C}_j$ which does not admit an exact mechanism in the general settings in which verification is not allowed [1]. By contrast, we prove the existence of an exact mechanism for $\mathrm{Q}||\sum \mathrm{w}_j\mathrm{C}_j$ for verifiable agents. This shows that the class of monotone algorithms is a proper subclass of the class of weakly-monotone algorithms.

We would like to mention (and refer the reader to the full version for formal statements of the results and proofs) that similar results about truthful mechanisms with respect to *Bayesian-Nash* equilibrium can be obtained for *quasi one-parameter* agents (also studied in [3]) characterized by a valuation function $v^i(X) := -(w^i(X,\sigma)\cdot t_i + a^i(X, t_{-i}))$, where the additional term $a^i(X, t_{-i})$ does not depend on the type t_i of agent i.

Notation and model. Following the standard notation used in the study of approximation of combinatorial optimization problems (see, e.g., [4]), 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; $\mathsf{m}(X, I)$ is the *measure* of feasible solution X of instance I and goal is either min or max. Thus

the optimization problem consists in finding feasible solution X^* for instance I such that $\mathsf{m}(X^*, I) = \mathsf{opt}(I) := \mathsf{goal}_{X \in \mathsf{sol}(I)} \mathsf{m}(X, I)$.

We consider optimization problems involving *selfish* agents that *privately know* part of the input: every instance I is composed of a *private part* $t = (t_1, t_2, \ldots, t_n)$ (where t_i the *private information* of agent i) and of a *public part* σ. We assume that the set of feasible solutions $\mathsf{sol}(I)$ does not depend on t, that is, for every $I = (\sigma, t)$ and $I' = (\sigma, t')$, $\mathsf{sol}(I) = \mathsf{sol}(I')$.

2 Truthful Mechanisms for Restricted Selfish Agents

In this section we consider the problem of designing truthful mechanisms for selfish restricted agents.

Definition 1 (restricted selfish agent). *An agent i is* overbidding *(respectively,* underbidding*) if her reported type b_i always satisfies $b_i \geq t_i$ (respectively, $b_i \leq t_i$). An agent is* restricted *if it is overbidding or underbidding.*

A mechanism M is truthful for restricted selfish agents if, for every overbidding (respectively, underbidding) agent i, $u^i(t_i) \geq u^i(b_i)$, for any $b_i \in \mathcal{S}^i$ with $b_i \geq t_i$ (respectively, $b_i \leq t_i$).

Not to overburden our notation, when algorithm A, the instance σ and the declarations b_{-i} of all the agents other than i are clear from the context, we will simply write $P^i(x)$ to denote the payment awarded to agent i declaring x. Similarly, we write $w_i(x)$ instead of $w^i_A((x, b_i), \sigma)$.

We first show that any algorithm A admits a payment function P_A such that (A, P_A) is a truthful mechanism for restricted selfish agents. We give a method for computing the payment function and show that it runs in polynomial time if A is polynomial-time and for each i the set $\mathcal{S}^i$ of the possible strategies or agent i has polynomial size in the length of the input.

Theorem 1. *For any algorithm A there exists a payment function $P_A = (P^1_A, P^2_A, \ldots, P^n_A)$ such that $M = (A, P_A)$ is a truthful mechanism for restricted selfish one-parameter agents. The payment functions P^i_A for instance I can be computed in time polynomial in $|\mathcal{S}^i|$ and in the running time of algorithm A on inputs of length $|I|$.*

Proof. We prove the theorem for overbidding agents by explicitly defining the payment $P^i(x)$. Consider agent i and let her strategy set S^i be $(\alpha^i_1 \leq \alpha^i_2 \leq \cdots \leq \alpha^i_l)$. Fix the declarations $b_{-i} = (b_1, \cdots, b_{i-1}, b_{i+1}, \cdots, b_m)$ of the other agents. Since agent i is overbidding to enforce the truthfulness of the mechanism we have to impose that for every $b_i, t_i \in \mathcal{S}^i$, with $b_i \geq t_i$, it holds that

$$P^i(t_i) - w^i(t_i) \cdot t_i \geq P^i(b_i) - w^i(b_i) \cdot t_i.$$

Observe that if t_i is the largest value in S^i then the agent cannot lie and the above condition trivially holds for any payment. For example, we can set $P^i(\alpha^i_l) :=$

$w^i(\alpha_l^i) \cdot \alpha_l^i$ so to guarantee that the utility of agent i is non-negative. Then, for $k = l-1, l-2, \cdots, 1$, we recursively compute $P^i(\alpha_k^i) := w^i(\alpha_k^i) \cdot \alpha_k^i + \delta_k^i$, where

$$\delta_k^i := \max_{j \geq k} \{P^i(\alpha_j^i) - w^i(\alpha_j^i) \cdot \alpha_k^i\}.$$

Suppose now $t_i = \alpha_k^i$ and she declares $b_i = \alpha_j^i$, with $j > k$. By the definition of δ_k^i it follows that $P^i(b_i) - w^i(b_i) \cdot t_i \leq \delta_k^i = P^i(t_i) - w^i(t_i) \cdot t_i$. This proves the theorem.

In the sequel, we describe a class of problems such that, given a c- approximation algorithm A, a small perturbation in the input given to A has a small impact on the quality of the solution produced by the algorithm. Thus, we can consider algorithm A_ϵ that rounds the input of the problem and computes a $c(1+\epsilon)$-approximation solution. We show that if the sets $\mathcal{S}^i$'s are intervals of integers or consist of the reciprocals of an interval of integers, then we can define polynomial-time computable payment functions P_{A_ϵ} such that $(A_\epsilon, P_{A_\epsilon})$ is a truthful mechanism.

Definition 2. *Fix $\epsilon > 0$ and $\gamma > 1$. A minimization problem $(\mathcal{I}, \mathsf{m}, \mathsf{sol}, \mathsf{goal})$ is* (γ, ϵ)–smooth *if for any pair of instances $I = (t, \sigma)$ and $\tilde{I} = (\tilde{t}, \sigma)$ such that $\max\{\frac{\tilde{t}_i}{t_i}, \frac{t_i}{\tilde{t}_i}\} \leq \gamma$, for any $i = 1, 2, \cdots, m$, it holds that*

$$\forall X \in \mathsf{sol}(\sigma), \quad \mathsf{m}(X, (t, \sigma)) \leq (1+\epsilon) \cdot \mathsf{m}(X, (\tilde{t}, \sigma))),$$
$$\mathsf{opt}(t, \sigma) \geq \mathsf{opt}(\tilde{t}, \sigma).$$

A maximization problem is (γ, ϵ)–smooth if the above two inequalities hold with '$\leq$' and '$\geq$' exchanged.

The proof of the following lemma is straightforward.

Lemma 1. *Let Π be a (γ, ϵ)–smooth problem and let $I = (t, \sigma)$ and $\tilde{I} = (\tilde{t}, \sigma)$ be two instances of Π such that $\tilde{t}_i \leq \gamma t_i$, for $i = 1, 2, \ldots, m$. If X is a c-approximate solution for the instance $\tilde{I}$ then X is a $c(1+\epsilon)$-approximate solution for the instance I.*

Let Π be a (γ, ϵ)–smooth minimization problem and let A be an algorithm for Π. We define the mechanism $M_\epsilon = (A_\epsilon, P_{A_\epsilon})$ as follows. For each x let $\mathsf{up}(x) := \min\{y \geq x |\ y = \gamma^a, a \in \mathbf{N}\}$ and $\mathsf{down}(x) := \min\{y \in \mathcal{S}^i |\ \mathsf{up}(y) = \mathsf{up}(x)\}$. The algorithm A_ϵ, on input (b, σ) simply runs algorithm A on the input $(\mathsf{up}(b), \sigma)$.

Consider the set $\mathcal{S}^i$ of strategies of agent i and let $\tilde{\mathcal{S}}^i = \{\tilde{\alpha}_1^i \leq \tilde{\alpha}_2^i \leq \ldots \leq \tilde{\alpha}_h^i\}$ where $\tilde{\alpha}_j^i = \mathsf{up}(x)$ for some $x \in \mathcal{S}^i$. We next consider overbidding agents (underbidding agents can be treated similarly) and define the payment function $P_{A_\epsilon}^i(x)$ for all $x \in \tilde{\mathcal{S}}^i$ and, for $x \notin \tilde{\mathcal{S}}^i$, we set $P_{A_\epsilon}^i(x) := P_{A_\epsilon}^i(\mathsf{up}(x))$.

The values of P^i are defined recursively, starting from the largest value of $\tilde{\mathcal{S}}^i$ to the smallest one. For each $x \in \tilde{\mathcal{S}}^i$ we set

$$P_{A_\epsilon}^i(x) := \begin{cases} x \cdot w_{A_\epsilon}^i(x), & \text{if } x = \tilde{\alpha}_h^i; \\ \max\left\{P_{\mathsf{down}}^i(x), P_{\mathsf{up}}^i(x)\right\}, & \text{otherwise;} \end{cases} \tag{1}$$

where

$$P^i_{\mathsf{up}}(x) := \max_{z \in \tilde{\mathcal{S}}^i, z > x} \{P^i_{A_\epsilon}(z) + \Delta_{A_\epsilon}(x, z) \cdot \mathsf{up}(x)\} \tag{2}$$

$$P^i_{\mathsf{down}}(x) := \max_{z \in \tilde{\mathcal{S}}^i, z > x} \{P^i_{A_\epsilon}(z) + \Delta_{A_\epsilon}(x, z) \cdot \mathsf{down}(x)\}, \tag{3}$$

and

$$\Delta_{A_\epsilon}(x, y) := w^i_{A_\epsilon}(x) - w^i_{A_\epsilon}(y). \tag{4}$$

Next theorem states that $(A_\epsilon, P_{A_\epsilon})$ is an approximating truthful mechanism with respect to restricted agents.

Theorem 2. *Let Π be a (γ, ϵ)–smooth problem and let A be a c–approximate algorithm for Π. $M_\epsilon = (A_\epsilon, P_{A_\epsilon})$ is a $c(1+\epsilon)$–approximate truthful mechanisms with respect to restricted selfish agents for Π. Moreover, the payment functions P_{A_ϵ} can be computed in time $O\left(t_A(k) \cdot \sum_{i=1}^n \log^2 |\mathcal{S}^i|\right)$, where $t_A(k)$ is the worst-case running time of algorithm A on inputs of length k.*

We mention that it is possible to define a class of optimization problems for which we do not need to assume that sets $\mathcal{S}^i$ have finite size. Details are omitted from this extended abstract.

We also have the following corollary.

Corollary 1. *The mechanism $M_\epsilon = (A_\epsilon, P_{A_\epsilon})$ satisfies voluntary participation with respect to restricted overbidding agents.*

Discussion. Let us now briefly discuss the application of Theorems 1 and 2 to the problem of scheduling related machines (which is a special one-parameter problem). Here the set $\mathcal{S}^i$ consists of all speeds that agent i, the owner of the i-th machine, may declare and, in general, $\mathcal{S}^i$ coincides with the set of natural numbers. However, it is easy to see that for each machine i, there exists a threshold τ_i such that if machine i has speed τ_i or higher then the optimal schedule assigns all jobs to machine i. Thus, without loss of generality we can assume that $\mathcal{S}^i$ is the interval $[1, \tau_i]$. Then Theorem 1 tells us that *any* scheduling algorithm A can be equipped with payment functions P such that (A, P) is a truthful mechanism for restricted agents. The payment functions P can be computed in time polynomial in the τ_i's. Theorem 2 instead tells us that, if we round the speeds to the powers of $(1+\epsilon)$ then we lose an extra $(1+\epsilon)$ factor in approximation. But then there exist payment functions P' computable in *polynomial time* (in the length of the τ_i) that yield a truthful mechanism for restricted agents.

3 Truthful Mechanisms for Verifiable Machines

In this section we study the $\mathrm{Q}||\mathrm{C}_{\max}$ problem for the case in which each machine is owned by a selfish agent and machines are verifiable.

Let us quickly review the classical problem $\mathrm{Q}||\mathrm{C}_{\max}$. We are given m machines of speeds $s_1, s_2, \ldots, s_m$ and a set of n jobs of weights $J_1, J_2, \ldots, J_n$. We have to

assign each job to some machine, and a job of weight J_l requires J_l/s time units in order to be processed by a machine of speed s. We want to find an assignment of the jobs to the machines in order to minimize the makespan, that is, we want to minimize $\max_{1\leq j\leq m} w^j/s_j$, where w^j denotes the sum of the weights of all jobs assigned to machine j.

We consider the setting in which agent i owns machine i and knows the speed s_i of the machine and, for convenience, we consider $t_i = 1/s_i$ as the type of agent i. The public information is the set of weights $\sigma = (J_1, J_2, \ldots, J_m)$ of the jobs that need to be scheduled on the machines. The valuation given to a schedule of the jobs of σ by agent i is equal to the finish time of its machine. We assume that machines are *verifiable* and payments are provided *after* the execution of the jobs. If machine i receives at least one job (i.e., $w^i > 0$), then we can verify whether $b_i < t_i$ by checking the release time T_i of its last job.Indeed, if $b_i < T_i/w^i$ then the mechanism discovers that the agent has declared to be faster that it actually is. Since payments are provided *after* the execution of jobs and thus also depend on the actual time T_i, we can make it inconvenient to claim faster speeds then the real one. This is different from the previous case of restricted agents as, if a machine receives no job, there is no way of verifying the bid neither if he underbid nor if he overbid. Thus, in this case there are allocation algorithms for which no payment exists.

Next we characterize the class of algorithms A that admit a payment function P_A for which (A, P_A) is a truthful mechanism with respect to dominant strategies for *verifiable machines*.

Definition 3 (weakly monotone algorithm). *An algorithm for a scheduling problem is* weakly monotone *if, for every i, for every b_{-i} it holds that*

$$w_A^i(b_i, b_{-i}) = 0 \Rightarrow \forall b_i' > b_i, \quad w_A^i(b_i', b_{-i}) = 0. \tag{5}$$

Theorem 3. *An algorithm A for the scheduling problem Π admits a payment function P such that $M = (A, P)$ is a truthful mechanism for verifiable machines if and only if A is weakly monotone.*

Proof. Assume that A is weakly monotone, let P' be payment functions such that $M' = (A, P')$ is a truthful mechanism for overbidding restricted agents that satisfy voluntary participation and define payment functions P in the following way.

$$P_i(b_i, T_i, b_{-i}) := \begin{cases} 0, & \text{if } w_A^i(b_i, b_{-i}) = 0 \text{ or } T_i > b_i w_A^i(b_i, b_{-i}); \\ P_i'(b_i, b_{-i}), & \text{otherwise;} \end{cases} \tag{6}$$

where T_i is the finish time of machine i. Notice that if T_i is greater than expected (that is, the bid b_i of agent i times the load assigned to it) the agent is punished and receives no payment. Let us now verify that (A, P) is truthful. We distinguish two cases.

1. $w_A^i(b_i) = 0$. If $b_i > t_i$, since A is weakly monotone, we have that $w_A^i(t_i) = w_A^i(b_i) = 0$. Eq. 6 implies $P_i(b_i, T_i) = P_i(t_i, T_i) = 0$. Thus the utility is the

same. If $b_i < t_i$, then $T_i > b_i w_A^i(b_i, b_{-i})$, thus the utility of the agent is 0 while by declaring $t_i$he would have a nonnegative utility.
2. $w_A^i(b_i) > 0$. If agent i reports $b_i < t_i$, then her utility is non-positive. If agent i reports $b_i \geq t_i$ it receives payment $P'(b_i)$ for a utility $u(b_i|t_i) = P'(b_i) - t_i w_A^i(b_i)$. On the other hand by reporting true type t_i, the agent would have had utility $u(t_i|t_i) = P'(t_i, b_{-i}) - t_i w_A^i(b_i, b_{-i})$. By the truthfulness of (A, P'), we have that $u(t_i|t_i) \geq u(b_i|t_i)$.
Assume now that there exists a payment function P such that (A, P) is truthful and, for the sake of contradiction, assume that there exist b_i and $b'_i > b_i$ such that $w_A^i(b_i) = 0$ and $w_A^i(b'_i) > 0$. Since $w_A^i(b_i) = 0$, we have no way to verify whether $b_i < t_i$. Moreover, if $t_i < b'_i$, then agent i can make his/her machine finish at time $T'_i = w_A^i(b'_i) \cdot b'_i$. In both cases, we cannot infer anything about the reported values b_i and b'_i. Since $M = (A, P)$ is truthful, then the following two conditions must be fulfilled:

$$t_i = b_i \;\Rightarrow\; P_i(b_i, 0) - w_A^i(b_i) \cdot b_i \geq P_i(b'_i, T'_i) - w_A^i(b'_i) \cdot b_i,$$
$$t_i = b'_i \;\Rightarrow\; P_i(b'_i, T'_i) - w_A^i(b'_i) \cdot b'_i \geq P_i(b_i, 0) - w_A^i(b_i) \cdot b'_i.$$

By using the fact that $w_A^i(b_i) = 0$ and combining the above two inequalities we obtain

$$\begin{aligned} P_i(b_i, 0) &\geq P_i(b'_i, T'_i) - w_A^i(b'_i) \cdot b_i \\ &\geq P_i(b_i, 0) + w_A^i(b'_i) \cdot b'_i - w_A^i(b'_i) \cdot b_i = P_i(b_i, 0) + w_A^i(b'_i)(b'_i - b_i) \end{aligned}$$

thus contradicting the hypothesis.

We mention (see final version for statement and proof) that, for verifiable machines, mechanism without payments perform very poorly.

4 A $(1 + \epsilon)$-Approximation Truthful Mechanism for $Q||C_{max}$

In this section we present a $(1+\epsilon)$-approximation polynomial-time truthful mechanism for $Q||C_{max}$ on selfish verifiable machines when speeds are integer and upper bounded by a constant S. Our mechanism is based on the well-known PTAS for $Q||C_{max}$ on uniform machines due to Hochbaum and Shmoys [7]. We modify their algorithm so to obtain $(1 + \epsilon)$-approximation also for machines of different speeds bounded by a constant and, more importantly, to guarantee weak monotonicity. This, combined with Theorem 3, implies the existence of a truthful polynomial-time $(1 + \epsilon)$-approximation mechanism for verifiable machines.

SCAN$_\epsilon$ takes as input (J, s) where $J = (J_1 \leq \ldots \leq J_n)$ are the jobs in nondecreasing order by weight and $s = (s_1 \leq \ldots \leq s_m)$ are the machine speeds, in nondecreasing order and performs the following steps.

1. $p := 1$;
2. if ORACLE$_\epsilon((1 + \epsilon)^p, J, s) =$ fail then
   while ORACLE$_\epsilon((1 + \epsilon)^p, J, s) =$ fail do $p := p + 1$

3. else while $\text{ORACLE}_\epsilon((1+\epsilon)^p, J, s) = \mathsf{fail}$ do $p := p - 1$;
4. return $(\text{ORACLE}_\epsilon((1+\epsilon)^p, J, s), (1+\epsilon)^p)$

Algorithm ORACLE_ϵ is a polynomial-time algorithm that, for every $\epsilon > 0$, on input $C > 0$ and an instance (J, s) either returns $\mathsf{success}$ along with a feasible solution of cost at most $(1+\epsilon)C$ or it returns fail in which case no feasible solution exists of cost smaller than C.

Algorithm ORACLE_ϵ. Algorithm ORACLE_ϵ receives as input a bound C and an instance (J, s). The algorithm starts by partitioning the jobs J into two sets: the set J^S of the small jobs consisting of all the jobs of size less than ϵC and the set J^L of large jobs containing the remaining jobs. The first phase deals with the large jobs while the second phase schedules the small jobs.

Phase I: SCHEDULING LARGE JOBS. We round the weight of each job $J_h \in J^L$ to J'_h computed as the maximum value $\epsilon C(1+\epsilon)^p$ that is not greater than J_h.

Let $J' = (J'_1, \cdots, J'_n)$ be the sequence of rounded job weights, and let $\mathcal{W} = \{W_1, \cdots, W_k\}$ be the set of distinct values of J'. Denote by n_h the number of jobs in J' whose weight is W_h (clearly $\sum_{h=1}^{k} n_h = n$). We can represent the instance J' of rounded jobs by the k–tuple $(n_1, n_2, \ldots, n_k)$.

Jobs J' are then given in input to a dynamic programming algorithm EXACT that checks whether there exists a scheduling of makespan at most C and returns such a scheduling if it exists. Algorithm EXACT runs in time polynomial in the number of possible values of job weights. By the rounding performed on the weights of the large jobs the number of possible weights of the jobs is $O\left(\log_{1+\epsilon} \frac{s_m}{\epsilon}\right) = O(\log S)$, that is constant.

For $1 \le l \le m$ and for any tuple $(i_1, \ldots, i_k)$ such that $i_h \le n_h$, we define $\mathsf{BINS}(C, i_1, i_2, \ldots, i_k, s, l)$ to take value 0 if there exists a scheduling of cost at most C for the set of jobs $(i_1, i_2, \ldots, i_k)$ using machines of speed $s_l, s_{l+1}, \ldots, s_m$; otherwise, $\mathsf{BINS}(C, i_1, i_2, \ldots, i_k, s, l)$ equals to 1. Clearly, there exists a scheduling for the jobs J' of cost at most C if and only if $\mathsf{BINS}(C, n_1, \ldots, n_k, s, 1) = 0$. To compute this value observe that $\mathsf{BINS}(C, i_1, i_2, \ldots, i_k, s, m) = 0$ if and only if it is possible to execute all jobs $(i_1, i_2, \ldots, i_k)$ on machine m in time not greater than C. Instead, for $l < m$ we have

$$\mathsf{BINS}(C, i_1, i_2, \ldots, i_k, s, l) = \min_{(q_1, q_2, \ldots, q_k)} \mathsf{BINS}(C, i_1 - q_1, i_2 - q_2, \ldots, i_k - q_k, s, l+1)$$

where the minimum is taken over the set $\mathsf{STORE}(C, s_l, i_1, \ldots, i_k)$ of tuples $Q = (q_1, \ldots, q_k)$ such that all jobs of Q can be executed on a machine of speed s_l and $q_h \le i_h$ for $h = 1, \ldots, k$. Observe that the cardinality of set $\mathsf{STORE}(C, s_i, i_1, \ldots, i_k)$ is $O(n^k)$ and we can recursively compute $\mathsf{BINS}(C, i_1, i_2, \ldots, i_k, s, 1)$ in time $O(mn^{2k})$. Moreover, algorithm EXACT also returns a scheduling of the large jobs of cost (with respect to the rounded weights) not greater than C. This is achieved by computing, for each k-tuple $(i_1, \ldots, i_k)$ and for each l, assignment $\mathsf{X}(C, i_1, i_2, \ldots, i_k, s, l)$ to machine l, that is the lexicographically minimal k–tuple (if any) $(q_1, q_2, \ldots, q_k) \in \mathsf{STORE}(C, s_l)$ such that $\mathsf{BINS}(C, i_1 - q_1, i_2 - q_2, \ldots, i_k - q_k, s, l+1) = 0$.

Phase II: SCHEDULING SMALL JOBS. Let X^1 be the allocation of the large jobs computed by the algorithm EXACT and let $w_i(X^1)$ be the work assigned to

machine i by X^1 with respect to the real weights of the jobs. We have now to complete X^1 by allocating the small jobs $J^S = \{J_1, J_2, \ldots, J_k\}$ not considered in the previous phase. We assign job J_i to the machine that minimizes its completion time with respect to the work assigned to it by X^1 and by the allocation of the small jobs $J_1, \ldots J_{i-1}$. If there are more machines that obtain the same completion time we select the slowest one. If the schedule obtained at the end of this phase has cost greater than $C(1+\epsilon)$ (with respect to the real weights of the jobs) then ORACLE$_\epsilon$ returns fail. Otherwise, ORACLE$_\epsilon$ continues with the adjustment phase.

Phase III: ADJUSTMENT. In the final adjustment phase the algorithm partitions the machine into two sets: slow machines and fast machines. The aim of this phase is to enforce that each fast machine will receive positive load. The partition is computed in the following way. We assume without loss of generality that the number of jobs is larger than the number of machines and stress that jobs are ordered by nondecreasing weight and machines are ordered by nondecreasing speed. The first m jobs are assigned each to one machine with machine i getting the J_i. Suppose that the makespan of the one-to-one scheduling obtained is greater than $C(1+\epsilon)$. Then, it is easy to observe that, all schedules that assign positive load to all machine have cost greater than $C(1+\epsilon)$. Thus we repeat the same procedure by considering the $m-1$ fastest machines and the first $m-1$ jobs. The procedure stops when we find a one-to-one scheduling of jobs $J_1, \ldots, J_{m-d}$ to the machines of speed $s_{d+1}, \ldots, s_m$ of cost not greater than $C(1+\epsilon)$. The set of slow machines will then be the set of the first d machines and the remaining machines constitute the set of fast machines. By the discussion above, it is easy to see that any scheduling that assigns positive load to more than $m-d$ machines has cost greater than $C(1+\epsilon)$. More precisely, by denoting with $\mathsf{discard}(u, J, s)$ the number d of slow machines for an instance (J, s) and bound u we have the following lemma.

Lemma 2. *Let $X \in \mathsf{sol}(J, s)$ be a schedule that assigns positive load to more than $m - \mathsf{discard}(u, J, s)$ machines. Then $\mathsf{m}(X, (J, s)) > u$.*

The adjustment phase then continues in the following way. Let X^2 be the schedule computed at the end of Phase II and remove jobs $J_1, \ldots, J_{m-d}$ from X^2. For each $i = 1, \ldots, m-d$, let ℓ_i be the machine to which X^2 assigns job J_i. If machine $(d+i)$ has no load then J_i is assigned to this machine; otherwise J_i is re-assigned to machine ℓ_i. The schedule X^3 obtained at the end of this phase is then given in output by ORACLE$_\epsilon$.

Lemma 3. *If* ORACLE$_\epsilon(C, J, s) \neq$ fail *then the schedule output by* ORACLE$_\epsilon$ *assigns positive loads exactly to the $m - \mathsf{discard}(C, J, s)$ fastest machines.*

The next theorem proves that either ORACLE$_\epsilon$ gives a schedule of cost at most $C(1+\epsilon)$ or no schedule of cost less than C exists.

Theorem 4. *If* ORACLE$_\epsilon(C, J, s) =$ fail*, then $\mathsf{opt}(J, s) \geq C$.*
If, instead, ORACLE$_\epsilon(C, J, s)$ *returns a schedule X then $\mathsf{m}(X, (J, s)) \leq C(1+\epsilon)$.*

Next theorem proves that ORACLE$_\epsilon$ is stable.

Definition 4 (stable algorithm). *An algorithm A is stable if, for every b_i, b_{-i} such that $w_A^i(b_i, b_{-i}) = 0$, it holds that, for every $b_i' > b_i$, $A(b_i', b_i) = A(b_i, b_{-i})$.*

Theorem 5. *Algorithm* ORACLE_ϵ *is stable.*

We are now in a position to prove that algorithm SCAN_ϵ is weakly monotone.

Theorem 6. *The algorithm* SCAN_ϵ *is weakly monotone.*

Proof. Let (J, s) and (J, s') be two instances such that $s' = (s_i', s_{-i})$ and $s_i' < s_i$. Let (C, X) and (C', X') be the output of $\text{SCAN}_\epsilon(J, s)$ and $\text{SCAN}_\epsilon(J, s')$, respectively. We show that if $w^i(X) = 0$ then $w^i(X') = 0$. We consider three cases, depending on the values of C and C'.

Consider first the case $C = C'$. In this case both X and X' are schedules computed by algorithm ORACLE_ϵ with respect to the bound C. By Theorem 5 the algorithm ORACLE_ϵ i stable and thus if $w^i(X) = 0$ then $X = X'$ and in particular $w^i(X') = 0$.

Consider now the case $C > C'$. By Lemma 3 if $w^i(X) = 0$ then $i < \mathsf{discard}(C, J, s)$ and machine i has been classified as a slow machine. Since $\mathsf{discard}(C', J, s') \geq \mathsf{discard}(C, J, s)$ and the machine has declared a slower speed then it is classified as a slow machine also when ORACLE_ϵ runs on (C, J, s'). Then $w^i(X') = 0$.

Finally, consider the case $C < C'$. We observe that this case is not possible because it implies that $\text{ORACLE}_\epsilon(C, J, s')$ returns fail, while $\text{ORACLE}_\epsilon(C, J, s)$ returns the schedule X such that $w^i(X) = 0$. By Theorem 5 $\text{ORACLE}_\epsilon(C, J, s')$ has computed a schedule equal to X and thus it cannot return fail.

Finally we prove SCAN_ϵ is a $(1 + \epsilon)$ approximation scheme for $\text{Q}||\text{C}_{\max}$.

Theorem 7. *Let J be a set of jobs to be scheduled on m selfish machines of speed s. Then for every $\epsilon > 0$, there exists a δ such that the cost of the scheduling computed by algorithm* SCAN_δ *is at most $(1 + \epsilon)\mathsf{opt}(J, s)$.*

Proof. Let $C = (1 + \delta)^i$ be the value returned by the searching phase of the algorithm SCAN_δ and let X be the corresponding allocation of cost at most $C(1 + \delta)^2$. Moreover, since ORACLE_ϵ returned fail on input (C, J, s) we know that $\mathsf{opt}(J, s) \geq C$. Thus, we have that

$$\mathsf{m}(X, (J, s)) \leq (1 + \delta)^2 C \leq (1 + \delta)^2 \mathsf{opt}(J, s) \leq (1 + \epsilon)\mathsf{opt}(J, s),$$

by choosing $\delta = \epsilon/3$.

By combining Theorem 3 with Theorems 6-7, we obtain a family of polynomial-time truthful $(1 + \epsilon)$-approximation mechanisms. Moreover, from the proof of Theorem 3 and from Theorem 2, it follows that also the payments can be computed in polynomial time.

5 The Power of Verification

In this section we show that the ability of verifying the agents' bids leads to approximation guarantees strictly better than what can be achieved without verification for the problem of $Q||\sum w_j C_j$. In the $Q||\sum w_j C_j$, for each job l, we are given a weight w_l and a processing requirement J_l: this job requires J_l/s units of time when processed by a machine of speed s. In addition, jobs must be processed on the same machine without being interrupted. Thus, the completion time of each job C_l also depends on the order in which the machine processed the jobs assigned to it. The goal is to minimize the weighted sum of all jobs completion times $\sum_{l=1}^{n} w_l C_l$. In [1], it is proved that, for $c < 2/\sqrt{3}$, no c-approximation algorithm for $Q||\sum w_j C_j$ is monotone. Consequently, no truthful mechanism can obtain approximation better than $2/\sqrt{3}$. Next we show that any optimal algorithm for $Q||\sum w_j C_j$ is weakly monotone. Thus, by Theorem 3, if the speeds of the machines can be verified there exists an optimal truthful mechanism.

Theorem 8. *Any exact algorithm $A^\star$ for $Q||\sum w_j C_j$ is weakly monotone.*

Proof. Consider two instances $\mathcal{I}$ and $\mathcal{I}'$ which differ only in the speed s_i and s_i' of the i-th machine and let X and X' be the assignments computed by $A^\star$ for the two instances. Assume by contradiction that $s_i > s_i'$ and that the load l_i assigned to machine i by X is 0 whereas X' assigns load $l_i' > 0$. Now observe that since $s_i > s_i'$, we have that $\mathsf{m}(X', \mathcal{I}') > \mathsf{m}(X', \mathcal{I})$ and, since X is optimal for $\mathcal{I}$, we have that $\mathsf{m}(X', \mathcal{I}) \geq \mathsf{m}(X, \mathcal{I})$. Finally, since $l_i = 0$, we have that $\mathsf{m}(X, \mathcal{I}) = \mathsf{m}(X, \mathcal{I}')$ which implies that $\mathsf{m}(X', \mathcal{I}') > \mathsf{m}(X, \mathcal{I})$ contradicting the optimality of $A^\star$.

References

1. A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. of the IEEE Symposium on Foundations of Computer Science*, pages 482–491, 2001.
2. V. Auletta, R. De Prisco, P. Penna, and G. Persiano. Deterministic truthful approximation mechanisms for scheduling related machines. Technical report, To appear in Proceedings of STACS 2004.
3. V. Auletta, R. De Prisco, P. Penna, and G. Persiano. How to tax and route selfish unsplittable traffic. Technical report, To appear in Proceedings of SPAA, 2004.
4. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*. Springer Verlag, 1999.
5. E.H. Clarke. Multipart Pricing of Public Goods. *Public Choice*, pages 17–33, 1971.
6. T. Groves. Incentive in Teams. *Econometrica*, 41:617–631, 1973.
7. D.S. Hochbaum and D.B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *J. of ACM*, 34:144–162, 1987.
8. N. Nisan and A. Ronen. Algorithmic Mechanism Design. In *Proc. of the 31st Annual ACM Symposium on Theory of Computing*, pages 129–140, 1999.
9. W. Vickrey. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, pages 8–37, 1961.