Deterministic Truthful Approximation Mechanisms for Scheduling Related Machines[⋆]

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 consider the problem of scheduling jobs on related machines owned by selfish agents and provide the first *deterministic* mechanisms with constant approximation that are *truthful*; that is, truth-telling is a *dominant strategy* for all agents. More precisely, we present deterministic polynomial-time $(2+\epsilon)$-approximation algorithms and suitable payment functions that yield truthful mechanisms for several NP-hard restrictions of this problem. Our result also yields a family of deterministic polynomial-time truthful $(4+\epsilon)$-approximation mechanisms for any fixed number of machines. The only previously-known mechanism for this problem (proposed by Archer and Tardos [FOCS 2001]) is 3-approximated, *randomized* and truthful under a *weaker* notion of truthfulness.
Up to our knowledge, our mechanisms are the first non-trivial polynomial-time deterministic truthful mechanisms for this NP-hard problem.
To obtain our results we introduce a technique to transform the PTAS by Graham into a deterministic truthful mechanism.

1 Introduction

The Internet is a complex distributed system where a multitude of heterogeneous entities (e.g., providers, autonomous systems, universities, private companies, etc.) offer, use, and even compete with each other for resources. Resource allocation is a fundamental issue for the efficiency of a complex system. Several efficient distributed protocols have been designed for resource allocation. The underlying assumption is that the entities running the protocol are trustworthy; that is, they behave as prescribed by the protocol. This assumption is unrealistic in some settings as the entities owning the resources might try to manipulate the system in order to get some advantages by reporting false information. For example, a router of an autonomous system can report false link status trying to redirect traffic through another autonomous system.

With false information even the most efficient protocol may lead to unreasonable solutions if it is not designed to cope with the selfish behavior of the

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

single entities. The field of *mechanism design* provides an elegant theory to deal with this kind of problems. The main idea of this theory is to pay the agents to convince them to perform strategies that help the system to optimize a global objective function. A *mechanism* $M = (A, P)$ is a combination of two elements: an algorithm A computing a solution and a payment rule P specifying the amount of “money” the mechanism should pay to each entity. Informally speaking, each agent i has a *valuation function* that associates to each solution X some value $v_i(X)$ and the mechanism pays i an amount $P_i(X, r_i)$ based on the solution X and on the *reported* information r_i. A *truthful mechanism* is a mechanism such that the payments guarantee that, when $X = X(r_i)$ is the solution computed by the mechanism, $u_i := P_i(X, r_i) + v_i(X)$ is maximized for r_i equal to the true information (see Sect. 1.3 for a formal definition).

Recently, mechanism design has been applied to several optimization problems arising in computer science, networking and algorithmic questions related to the Internet (see [10] for a survey). In the seminal papers by Nisan and Ronen [8, 9] (see also [11]) it is first pointed out that classical results in mechanism design theory, originated from micro economics and game theory, do not completely fit in a context where computational issues play a crucial role [9].

The main purpose of this paper is to provide polynomial-time approximation truthful mechanisms for the problem of scheduling jobs on parallel related machines ($Q||C_{\max}$).

1.1 Previous Work

The theory of mechanism design dates back to the seminal papers by Vickrey [12], Clarke [4] and Groves [7]. Their celebrated *VCG mechanism* is still the prominent technique to derive truthful mechanisms for many problems (e.g., shortest path, minimum spanning tree, etc.). In particular, when applied to combinatorial optimization problems (see e.g., [8,11]), the VCG mechanisms guarantee the truthfulness under the hypothesis that the optimization function is *utilitarian*[1] and the mechanism is able to compute the optimum.

Unfortunately, none of these hypothesis holds for $Q||C_{\max}$ since we aim at minimizing the *maximum* over all machines of their completion times, and the problem is NP-hard [5].

In [2] the authors characterize those algorithms which can be turned into a truthful mechanism for $Q||C_{\max}$. Their beautiful result brings us back to “pure algorithmic problems” as all we need is to find a good algorithm for the original problem which also satisfies the additional *monotonicity* requirement: increasing the speed of exactly one machine does not make the algorithm decrease the work assigned to that machine (see Sect. 1.3 for a formal definition, and Theorem 7 below). The authors then provide (i) an exact truthful mechanism based on the the algorithm computing the (lexicographically minimal) optimal solutio and

[1] A maximization problem is *utilitarian* if the optimization can be written as the sum of the agents’ valuation functions.

(ii) a *randomized* 3-approximation mechanism that is *truthful in expectation*, a weaker notion of truthfulness.

Nisan and Ronen [8,11] considered the *unrelated* machines case and provide an n-approximation deterministic truthful mechanism for it (n is the number of machines). Rather surprisingly, this mechanism is optimal for the case of $n = 2$. For the case $n > 2$ [8,11] prove that a wide class of "natural" mechanisms cannot achieve a factor better than n, if we require truthfulness. Finally, for $n = 2$ [8, 11] give a *randomized* 7/4-approximation mechanism.

There is a significant difference between the definition of truthfulness used in [8,11] and the one used in [2]. Indeed, the randomized 7/4-approximation algorithm in [8,11] yields a truthful dominant strategy for *any* possible random choice of the algorithm. In [2], instead, the notion of utility is replaced by the *expected utility* one: even though the expected utility is maximized when telling the truth, for some random outcome, there might exist a better (untruthful!) strategy.

This idea is pushed further in [1] where *one parameter agents* are considered for the problem of combinatorial auction. In this work, truthfulness is achieved w.r.t. expected utility and with high probability, that is, the probability that an untruthful declaration improves the agent utility is arbitrarily small.

1.2 Our Contribution

It is natural to ask whether some problems require some relaxation on the definition of truthfulness in order to achieve polynomial-time approximation mechanisms. In this paper we investigate the existence of truthful polynomial-time approximation mechanisms for $Q||C_{\max}$, while maintaining the *strongest* definition of truthfulness: *truth-telling is a dominant strategy over all possible strategies of an agent.*

We first show that, for any fixed number of machines, $Q||C_{\max}$ admits a deterministic truthful $(2+\epsilon)$-approximation mechanism if there exists a monotone allocation algorithm `Gc` whose cost is within an additive factor of $O(t_{\max}/s_1)$ from the cost of `Greedy`, where $t_{\max}$ is the largest job weight and s_1 is the smallest machine speed (see Sect. 2). Our result is a modification of the classical PTAS [6]. Notice that this PTAS cannot be used to construct a truthful mechanism because `Greedy` is not monotone and the allocation produced by the combination of the two algorithms (the optimal and the greedy one) is also not monotone. Our technical contribution here is the analysis of a new algorithm obtained by combining the optimal algorithm and `Gc`, that preserves the monotonicity *and* whose cost is within a factor of 2 of the cost of the PTAS.

We then show that such a monotone algorithm `Gc` exists for the following versions of the problem (see Sect. 3):

– speeds are integer and the largest speed is bounded from above by a constant;
– speeds are *divisible*, that is, they belong to a set $C = \{c_1, c_2, \ldots, c_p, \ldots\}$ such that for each i, c_{i+1} is a multiple of c_i.

Thus, for both these cases, we obtain a family of deterministic truthful $(2+\epsilon)$-approximation mechanisms (see Sect. 4). Observe that all such restrictions remain NP-hard even for two machines [5]. Up to our knowledge, this is the first result in which approximate solutions yield truthful mechanisms, where truthfulness is defined in the strongest sense. Indeed, the mechanism in [2] is only truthful on average. Although our new algorithm is relatively simple, its analysis, in terms of monotonicity and approximability, is far from trivial and goes through several properties of greedy allocations on identical machines.

We emphasize that the importance of an approximating mechanism for the case of divisible speeds is both practical and theoretical. Indeed, on one hand, in many practical applications "speeds" are not arbitrary but they are taken from a pre-determined set of "types", yielding values that are multiple with each other. Moreover, this result implies the existence, for any fixed number of machines, of deterministic truthful $(4+\epsilon)$-approximate mechanisms for the case of *arbitrary* speeds, for any $\epsilon > 0$.

Observe that, also in the case of divisible speeds, existing and natural approximation algorithms are not monotone, and thus they are not suitable for truthful mechanisms (see [3] for a discussion).

Finally, our mechanisms satisfy *voluntary participation* and are able to compute the payments within polynomial time (see Sect. 4). The latter is a property that cannot be directly derived from the results in [2].

Due to lack of space some proofs are omitted or only sketched. We refer the interesting reader to the full version of this work [3].

1.3 Preliminaries

We consider the problem of scheduling on related parallel machines $(Q||C_{\max})$. We are given the *speed vector* $s = \langle s_1, s_2, \ldots, s_n \rangle$, with $s_1 \leq s_2 \leq \ldots \leq s_n$, of the of n machines and a *job sequence* with weights $\sigma = (t_1, t_2, \ldots, t_m)$. In the sequel we simply denote the i-th job with its weight t_i. The largest job weight in σ is denoted by $t_{\max}$. A schedule is a mapping that associates each job to a machine. The amount of time to complete job j on machine i is t_j/s_i. The *work* of machine i, denoted as w_i, is given by the sum of the weights of the jobs assigned to i. The *load* (or finish time) of machine i is given by w_i/s_i. The cost of a schedule is the maximum load over all machines, that is, its *makespan*. Given an algorithm A for $Q||C_{\max}$, $A(\sigma, s)$ denotes the solution computed by this algorithm on input the job sequence σ and the speed vector s. The cost of the solution computed by algorithm A on input σ and s is denoted by $\mathsf{cost}(A, \sigma, s)$. We will also consider scheduling algorithms that take as third input the parameter h. In this case we denote by $A(\sigma, s, h)$ the schedule output and by $\mathsf{cost}(A, \sigma, s)$ its cost.

We consider $Q||C_{\max}$ in the context of selfish agents in which each machine is owned by an agent and the value of s_i is *privately* known to the agent. A mechanism for this problem is a pair $\mathcal{M} = (A, P)$, where A is an algorithm to construct a solution and P is a *payment function*. In particular, the mechanism asks each agent i to report her speed and, based on the *reported* costs, constructs a solution using A and pays the agents according to $P = (P_1, P_2, \ldots, P_n)$. The profit of agent i is defined as $profit_i = P_i - w_i/s_i$, that is, payment minus the cost incurred by the agent in being assigned work w_i.

A *strategy* for an agent i is to declare a value b_i for her speed. Let b_{-i} denote $b_1, b_2, \ldots, b_{i-1}, b_{i+1}, \ldots, b_n$. A strategy $\overline{b_i}$ is a *dominant strategy* for agent i, if $\overline{b_i}$ maximizes $profit_i$ for *any* possible b_{-i}. A mechanism is *truthful* if, for any agent i, declaring her true speed is a dominant strategy. A mechanism satisfies *voluntary participation* if, for any agent i, declaring her true speed yields a non-negative utility.

An algorithm for the $Q||C_{\max}$ problem is *monotone* if, given in input the machine speeds $b_1, b_2, \ldots, b_n$, for any i and fixed b_{-i}, the work w_i is non decreasing in b_i.

Given a sequence σ of m jobs, we denote by σ_h the subsequence consisting of the first h jobs in σ, for any $h \leq m$; moreover, $\sigma \setminus \sigma_h$ denotes the sequence obtained by removing from σ the h first jobs.

The `Greedy` algorithm (also known as the ListScheduling algorithm [6]) processes jobs in the order they appear in σ and assigns a job t_j to the machine i minimizing $(w_i + t_j)/s_i$, where w_i denotes the work of machine i before job t_j is assigned; if more than one machine minimizing the above ratio exists then the one of smallest index is chosen.

An *optimal algorithm* computes a solution of minimal cost $\mathsf{opt}(\sigma, s)$. Throughout the paper we assume that the optimal algorithm always produces the *lexicographically minimal* optimal assignment. As shown in [2], this algorithm is monotone.

An algorithm A is a c-approximation algorithm if, for every instance (σ, s), $\mathsf{cost}(A, \sigma, s) \leq c \cdot \mathsf{opt}(\sigma, s)$. A *polynomial-time approximation scheme* (PTAS) for a minimization problem is a family $\mathcal{A}$ of algorithms such that, for every $\epsilon > 0$ there exists a $(1+\epsilon)$-approximation algorithm $A_\epsilon \in \mathcal{A}$ whose running time is polynomial in the size of the input.

2 Combining Monotone Algorithms with the Optimum

In this section we show how to combine an optimal schedule on a subsequence of the jobs with the one produced by a monotone algorithm on the remaining jobs in order to obtain a good monotone approximation algorithm. Our approach is inspired by the PTAS of R. L. Graham [6] that can be described as follows. First, we optimally assign the h largest jobs. Then, we complete this assignment by running `Greedy` on the remaining jobs according to the work assigned to the machines in the previous phase.

Unfortunately, this PTAS is not monotone. Indeed, even though the first phase is monotone, it is easy to see that `Greedy` is not monotone [2]. Moreover, even if we replace `Greedy` with a monotone algorithm the resulting algorithm is not guaranteed to be monotone. We, instead, propose the following approach.

Let `Gc` be any scheduling algorithm. By `Opt-Gc` we denote the following algorithm.

Algorithm Opt-Gc
Input: a job sequence σ, speed vector s, and parameter h.
Assume that the jobs in σ are ordered in non-increasing order by weight.

A. compute the lexicographically minimal schedule among those that have optimal makespan with respect to job sequence σ_h and speed vector s;
B. run algorithm Gc on job sequence $\sigma \setminus \sigma_h$ and speed vector s assuming that machine i has initial load 0, $i = 1, \cdots, n$;
output the schedule that assigns to machine i the jobs assigned to machine i in Phase A and Phase B.

We have the following lemma.

Lemma 1. *If* Gc *is monotone then* Opt-Gc *is also monotone.*

In the next sections we show that, if Gc has an approximation factor close to the one of the greedy algorithm, then, for each $\epsilon > 0$ and for each number n of machines, it is possible to choose the value of the parameter h so that Opt-Gc outputs a schedule of makespan at most $(2 + \epsilon)$ times the optimal schedule.

We start by defining the notion of a *greedy-close* algorithm.

Definition 1 (greedy-close algorithm). *Let c be a constant. An algorithm* Gc *is* c-greedy-close *if, for any job sequence σ and any machine speed vector $s = \langle s_1, s_2, \ldots, s_n \rangle$,* $\mathsf{cost}(\mathtt{Gc}, \sigma, s) \leq \mathsf{cost}(\mathtt{Greedy}, \sigma, s) + c \cdot t_{max}/s_1$. *An algorithm* Gc *is* greedy-close *if it is c-greedy-close for some constant c.*

2.1 Approximation Analysis of Opt-Gc

In this section, we show that the approximation factor of Opt-Gc is at most twice the approximation factor of PTAS-Gc, where PTAS-Gc computes the optimal schedule on the h largest jobs and then combines it with a greedy-close solution computed using algorithm Gc. Moreover, in order to guarantee a "good" approximability, it makes a *balancing* step in Phase B where jobs are assigned to non-bottleneck machines to reduce the unbalancing, while keeping the solution optimality.

Algorithm PTAS-Gc
Input: a job sequence σ, speed vector s, and parameter h.
Assume that the jobs in σ_h are the h largest jobs of σ.

A. compute the lexicographically minimal schedule among those that have optimal makespan with respect to job sequence σ_h and speed vector s;
let $\mathsf{opt}(\sigma_h, s)$ be the makespan of the schedule produced in this phase;
B. reduce unbalancing without increasing cost by running algorithm Greedy as long as it is possible to add jobs without exceeding $\mathsf{opt}(\sigma_h, s)$ and let h' be the last job considered in this phase;
C. run algorithm Gc on job sequence $\sigma \setminus \sigma_{h'}$ and vector speed s assuming that machine i has initial load 0, for $i = 1, \cdots, n$;

output the schedule that assigns to machine i the jobs assigned to machine i in phases A, B and C.

Let PTAS-Greedy be algorithm PTAS-Gc with $\mathtt{Gc} = \mathtt{Greedy}$. We define the quantity $\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Greedy}, \sigma, s, h) = \mathsf{opt}(\sigma_h, s) + \mathsf{cost}(\mathtt{Greedy}, \sigma \setminus \sigma_{h'}, s)$, where h' is the value computed in Phase B. It is easy to see that $\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Greedy}, \sigma, s, h) \geq \mathsf{cost}(\mathtt{PTAS\text{-}Greedy}, \sigma, s, h)$. Moreover, let Greedy* denote the algorithm that, on input σ and $s = \langle s_1, \dots s_n \rangle$, returns as output the best schedule among those computed by Greedy on input σ and speed vectors $\langle 0, \dots, 0, s_k, \dots, s_n \rangle$ for $k = 1, \dots, n$. Let us also define $\overline{\mathsf{cost}}(\mathtt{Gc}, \sigma, s, \alpha) := \mathsf{cost}(\mathtt{Greedy}^*, \sigma, s) + (1 + c)t_{\max}/\alpha$.

It is then possible to prove the following results: (i) $\mathsf{cost}(\mathtt{Greedy}, \sigma, s) \leq \mathsf{cost}(\mathtt{Greedy}^*, \sigma, s) + t_{max}/s_1$, (ii) $\mathsf{cost}(\mathtt{Gc}, \sigma, s) \leq \overline{\mathsf{cost}}(\mathtt{Gc}, \sigma, s, s_1)$, and (iii) $\overline{\mathsf{cost}}(\mathtt{Gc}, \sigma, \langle 0, \dots, 0, s_k, s_{k+1} \dots, s_n \rangle, s_1) \leq \overline{\mathsf{cost}}(\mathtt{Gc}, \sigma, \langle 0, \dots, 0, s_{k+1} \dots, s_n \rangle, s_1)$.

To upper bound the cost of PTAS-Gc, we consider the following quantity:

$$\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc}, \sigma, s, h) := \mathsf{opt}(\sigma_h, s) + \overline{\mathsf{cost}}(\mathtt{Gc}, \sigma \setminus \sigma_{h'}, s, s_1),$$

where h' is the index of the last job considered in Phase B of PTAS-Gc. Because of (ii) above, we have that $\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc}, \sigma, s, h) \geq \mathsf{cost}(\mathtt{PTAS\text{-}Gc}, \sigma, s, h)$.

The next two lemmas provide an upper bound on $\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc}, \sigma, s, h)$.

Lemma 2. *For any job sequence σ, any h, and any speed vector s of length n*

$$\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Greedy}, \sigma, s, h) \leq \mathsf{cost}(\mathtt{PTAS}, \sigma, s, h) + \frac{\mathsf{opt}(\sigma, s)}{h \cdot s_1} \left(\sum_{i=1}^{n} s_i \right) (n-1).$$

Lemma 3. *If* Gc *is c-greedy-close, then for any job sequence σ, any h, and any speed vector s of length n*

$$\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc}, \sigma, s, h) \leq \overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Greedy}, \sigma, s, h) + \frac{(1+c) \cdot \mathsf{opt}(\sigma, s)}{h \cdot s_1} \sum_{i=1}^{n} s_i.$$

We next provide a bound on the cost of PTAS-Gc in terms of $\mathsf{opt}(\sigma, s)$ and s_n/s_1.

Theorem 1. *If* Gc *is c-greedy-close then, for any job sequence σ, any h, and any speed vector s of length n,*

$$\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc}, \sigma, s, h) \leq \mathsf{opt}(\sigma, s) \left(1 + \frac{f(n) + n^2 + c \cdot n}{h} \frac{s_n}{s_1} \right).$$

Proof. By previous lemmata we have

$$\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc}, \sigma, s, h) \leq \mathsf{cost}(\mathtt{PTAS}, \sigma, s, h) + \frac{\mathsf{opt}(\sigma, s)}{h \cdot s_1} \left(\sum_{i=1}^{n} s_i \right) (n-1)$$

$$
\begin{aligned}
&+ (1+c)\cdot \frac{\mathsf{opt}(\sigma,s)}{h\cdot s_1}\sum_{i=1}^{n} s_i\\
&\leq \mathsf{opt}(\sigma,s)\left(1+\frac{f(n)}{h+1}+\frac{n+c}{h\cdot s_1}\sum_{i=1}^{n} s_i\right)\\
&< \mathsf{opt}(\sigma,s)\left(1+\frac{f(n)+n^2+c\cdot n}{h}\frac{s_n}{s_1}\right),
\end{aligned}
$$

where the last inequality follows from $\mathsf{cost}(\mathtt{PTAS},\sigma,s,h) \leq \mathsf{opt}(\sigma,s)\left(1+\frac{f(n)}{h+1}\right)$ (see [6]) and $s_i \leq s_n$. □

The bound given by Theorem 1 is good for small values of s_n/s_1. When instead, s_n is much larger than s_1 it might be convenient to neglect the machine with speed s_1 and run instead PTAS-Gc only on the remaining $n-1$ machines. In the next theorem, we prove that in this way we can obtain $(1+\epsilon)$ approximation for any value of $\epsilon > 0$. The proof of theorem is based on the following technical lemma.

Lemma 4. *If* Gc *is greedy-close, then for all σ, h and $s = \langle s_1, s_2, \ldots, s_n\rangle$*

$$\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc},\sigma,\langle s_1,s_2,\ldots,s_n\rangle,h) \leq \overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc},\sigma,\langle 0,s_2,\ldots,s_n\rangle,h).$$

Theorem 2. *For any positive integer n and for any $\epsilon > 0$, if* Gc *is a polynomial-time greedy-close algorithm, then there exists an h such that, for all σ and for all speed vectors s of length n,* $\mathsf{cost}(\mathtt{PTAS\text{-}Gc},\sigma,s,h) \leq (1+\epsilon)\mathsf{opt}(\sigma,s)$. *Moreover, the running time of* PTAS-Gc *is polynomial in $m = |\sigma|$.*

Proof. We will prove by induction on n that for any $\epsilon > 0$ there exists an h, depending on ϵ and n only, such that $\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc},\sigma,s,h) \leq (1+\epsilon)\mathsf{opt}(\sigma,s)$. The base case $n = 1$ is trivial. For the inductive step assume that, for any $\epsilon > 0$, there exists h such that $\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc}(\sigma,\langle 0,s_2,\ldots,s_n\rangle,h) \leq (1+\epsilon)\mathsf{opt}(\sigma,\langle 0,s_2,\ldots,s_n\rangle)$. If $s_n/s_1 \leq \epsilon$, then by Theorem 1, it is possible to pick $h = h(n,\epsilon)$ so that $\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc},\sigma,s,h) \leq (1+\epsilon)\mathsf{opt}(\sigma,s)$. Otherwise, pick ϵ' such that $(1+\epsilon')(1+s_1/s_n) \leq (1+\epsilon)$. Then by Lemma 4 and by inductive hypothesis it is possible to choose $h' = h'(n-1,\epsilon')$ such that

$$
\begin{aligned}
\overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc},\sigma,\langle s_1,s_2,\ldots,s_n\rangle,h') &\leq \overline{\mathsf{cost}}(\mathtt{PTAS\text{-}Gc},\sigma,\langle 0,s_2,\ldots,s_n\rangle,h')\\
&\leq (1+\epsilon')\mathsf{opt}(\sigma,\langle 0,s_2,\ldots,s_n\rangle)\\
&\leq (1+\epsilon')(1+s_1/s_n)\mathsf{opt}(\sigma,\langle s_1,s_2,\ldots,s_n\rangle)\\
&\leq (1+\epsilon)\mathsf{opt}(\sigma,s).
\end{aligned}
$$

Finally, the running time is $O(n^{h+2} + m\log m + \mathrm{poly}(m))$. □

We are now ready to prove the main result of this section.

Theorem 3. *For any positive integer n and for any $\epsilon > 0$, if* Gc *is greedy-close, then there exists an $h = h(n, \epsilon)$ such that for all sequences of jobs σ and all speed vectors s of length n,* $\mathsf{cost}(\mathsf{Opt\text{-}Gc}, \sigma, s) \leq (2 + \epsilon)\mathsf{opt}(\sigma, s)$.

Proof Sketch. Fix $\epsilon > 0$. Let $h = h(n, \epsilon)$ be such that $\mathsf{cost}(\texttt{PTAS-Gc}, \sigma, s, h) \leq (1 + \epsilon/2)\mathsf{opt}(\sigma, s)$ (such an h exists by Theorem 2) and let h' be the index of the last job scheduled during phase B by algorithm PTAS-Gc on input σ, s, and h. Construct a new job sequence σ' from σ by adding, just after $t_{h'}$, a copy of the jobs from t_{h+1} to $t_{h'}$. It is possible to prove that the cost of the schedule produced by PTAS-Gc on input σ', s, and h is not less than the cost of the schedule produced by Opt-Gc on σ, s, and h (see [3]).

We observe that the set of new jobs, considered independently from the rest of the sequence, can be scheduled in time $\mathsf{opt}(\sigma, s)$ (using the same schedule computed in phase B of PTAS-Gc) and thus $\mathsf{opt}(\sigma', s) \leq 2\mathsf{opt}(\sigma, s)$. Then, we have

$$\begin{aligned}\mathsf{cost}(\mathsf{Opt\text{-}Gc}, \sigma, s, h) &\leq \mathsf{cost}(\texttt{PTAS-Gc}, \sigma', s, h)\\ &\leq (1 + \epsilon/2)\mathsf{opt}(\sigma', s) \leq (2 + \epsilon)\mathsf{opt}(\sigma, s)\end{aligned}$$

and the theorem follows. □

3 A Monotone Greedy-Close Algorithm

In this section we describe a greedy-close algorithm that is monotone for the case of "divisible" speeds (see Def. 2 below). We present our algorithm for the case of *integer* divisible speeds; this is without loss of generality, as in case the divisible speeds are not integers then they can be scaled to be integers.

Let us consider the following algorithm:

Algorithm `uniform`

Input: a job sequence σ and speed vector $s = \langle s_1, s_2, \cdots, s_n \rangle$, with $s_1 \leq s_2 \leq \cdots \leq s_n$.

A. run algorithm `Greedy` on job sequence σ and $S = \sum_{i=1}^{n} s_i$ identical machines;

B. order the identical machines by nondecreasing load $l_1, \ldots, l_S$;

C. let $g := GCD(s_1, s_2, \ldots, s_n)$ and split the identical machines into g blocks $B_1, \cdots, B_g$ each consisting of S/g consecutive identical machines. For $1 \leq i \leq g$ and $1 \leq k \leq S/g$, denote by $B_i(k)$ the k-th identical machine of the i-th block. Thus identical machine $B_i(k)$ has load $l_{(i-1) \cdot S/g + k}$.

D. for $1 \leq j \leq n$ let $k_j = \sum_{l=1}^{j-1} s_l / g$; then machine j receives the load of identical machines $B_i(k_j + 1), \cdots, B_i(k_j + s_j/g)$, for each block $1 \leq i \leq g$;

As it is described above, algorithm `uniform` does not run in polynomial time as its running time depends on S which, in general, is not polynomially bounded in n and m. However, `uniform` can be easily modified so to obtain the same allocation in $O(n \cdot m + m \log m)$ time.

3.1 Approximation Analysis of `uniform`

Let us denote by w_j^i the work of the s_j/g identical machines from block B_i whose loads are assigned to machine j. Then we have that $w_j^i = \sum_{k=1}^{s_j/g} l_{(i-1)\cdot S/g+k}$.

Theorem 4. *For any job sequence σ and any integer speed vector $s = \langle s_1, s_2, \ldots, s_n\rangle$ it holds that $\mathsf{cost}(\mathtt{uniform}, \sigma, s) \leq \mathsf{opt}(\sigma, s) + t_{\max}/g$, where $g = GCD(s_1, s_2, \ldots, s_n)$.*

When s_1 divides all s_is, we have that $g = s_1$ and the `uniform` algorithm is greedy-close. We then define sequences of speeds that enjoy this property, which will be used below to prove the monotonicity of `uniform`.

Definition 2 (divisible speeds). *Let $C = \{c_1, c_2, \ldots, c_p, \ldots\}$, with the property that c_i divides c_{i+1}. Then a speed vector $s = \langle s_1, s_2, \ldots, s_n\rangle$ is* divisible *if $s \in C^n$. The* restriction to divisible speeds *denotes the problem version in which the set C is known to the algorithm and all declared speeds must be in C.*

We thus have following theorem.

Theorem 5. *Algorithm* `uniform` *is greedy-close when restricted to divisible speeds.*

3.2 Algorithm `uniform` Is Monotone

In order to prove the monotonicity of algorithm `uniform` we first prove some technical results on greedy allocations on identical machines.

Lemma 5. *Let L_i (respectively, l_i) denote the load of the i-th least loaded machine when* `Greedy` *uses N (respectively, $N+1$) identical machines. It then holds that $l_{i+1} \leq L_i$, for all $1 \leq i \leq N$.*

Lemma 6. *Let L_i (respectively, l_i) denote the load of the i-th least loaded machine when* `Greedy` *uses N (respectively, $N' > N$) identical machines. It holds that $L_i \leq l_i + l_{i+1}$, for $1 \leq i \leq N$.*

PROOF. We prove the lemma for $N' = N + 1$ since this implies the same result for any $N' > N$. For any $1 \leq i \leq N$, Lemma 5 yields $\sum_{k=1}^{i-1} L_k \geq \sum_{k=1}^{i-1} l_k$ and $\sum_{k=i+1}^{N} L_k \geq \sum_{k=i+1}^{N} l_{k+1}$, thus implying $L_i \leq l_i + l_{i+1}$. □

Lemma 7. *Let L_i (respectively, l_i) denote the load of the i-th least loaded machine when* `Greedy` *uses N (respectively, $N' > N$) identical machines. For any a, b, b' such that $N - b \leq N' - b'$ it holds that*

$$W(a, b) := \sum_{i=a}^{b} L_i \geq W'(a + b' - b, b') := \sum_{i=a+b'-b}^{b'} l_i.$$

PROOF. Let $d = N' - N$. By repeatedly applying Lemma 5 we obtain $L_i \geq l_{i+d}$, for $1 \leq i \leq N$. Since $b' - b \leq N' - N = d$, it holds that $L_i \geq l_{i+d} \geq l_{i+b'-b}$, for $1 \leq i \leq N$. This easily implies the lemma. □

We can now prove that `uniform` is monotone. Intuitively, if an agent increases her speed, then the overall work assigned to the other agents cannot decrease.

Theorem 6. *Algorithm* `uniform` *is monotone when restricted to divisible speeds.*

4 Polynomial-Time Mechanisms

Computing the payments. We make use of the following result:

Theorem 7 ([2]). *A decreasing output function admits a truthful payment scheme satisfying voluntary participation if and only if $\int_0^\infty b_i w_i(b_{-i}, u)du < \infty$ for all i,b_{-i}. In this case, we can take the payments to be*

$$P_i(b_{-i}, b_i) = b_i w_i(b_{-i}, b_i) + \int_0^\infty b_i w_i(b_{-i}, u)du. \tag{1}$$

We next show how to compute the payments in Eq. (1) in polynomial time when the work curve corresponds to the allocation of `PTAS-uniform`.

Theorem 8. *Let A be a polynomial-time r–approximation algorithm. It is possible to compute the payment functions in Equation (1) in time poly(n, m) when (i) all speeds are integer not greater than some constant M, and (ii) speeds are divisible.*

PROOF. Observe that since A is an r–approximation algorithm there exists a value $\overline{S} \leq r \cdot S$, where $S = \sum_{i=1}^n s_i$, such that on input $(s_{-i}, \overline{S})$, the algorithm assigns all jobs to machine i. Then, in order to compute the work curve of machine i we have only to consider speed values in the interval $[0, \overline{S}]$. Since A runs in polynomial time, if speeds are integer, it is always possible to compute the work curve within time $O(S \cdot poly(n, m))$. When all speeds are not larger than M, we have that $S \in O(n \cdot M)$ and the first part of the theorem follows.

Suppose now that speeds are divisible. In this case all the speeds belong to the interval $[2^{-l}, 2^l]$, where l is the length in bits of the input. Then, there are $O(\log 2^l)$ distinct speed values that machine i can take. So, the computation of the work curve takes $O(l \cdot \text{poly}(n, m)) = O(\text{poly}(n, m))$. □

Truthful approximation mechanisms.

Theorem 9. *There exists a truthful polynomial-time $(2 + \epsilon)$-approximation mechanism for $Q||C_{\max}$ when (i) all speeds are integer bounded above by some constant M, or (ii) speeds are divisible. Moreover, the mechanism satisfies voluntary participation and the payments can be computed in polynomial time.*

Theorem 10. *For every $\epsilon > 0$, there exists a truthful polynomial-time $(4 + \epsilon)$-approximation mechanism for $Q||C_{\max}$. Moreover, the mechanism satisfies voluntary participation and the payments can be computed in polynomial time.*

References

1. A. Archer, C. Papadimitriou, K. Talwar, and E. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proc. of the 14th SODA*, 2003.
2. A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. of FOCS*, pages 482–491, 2001.
3. Vincenzo Auletta, Roberto De Prisco, Paolo Penna, and Pino Persiano. Deterministic truthful approximation mechanisms for scheduling related machines. Technical Report 2, European Project CRESCCO, http://www.dia.unisa.it/~penna/papers/related-MDfull.ps.gz, 2003.
4. E.H. Clarke. Multipart Pricing of Public Goods. *Public Choice*, pages 17–33, 1971.
5. M.R. Garey and D.S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* Freeman, 1979.
6. R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Tech. Journal*, 45:1563–1581, 1966.
7. T. Groves. Incentive in Teams. *Econometrica*, 41:617–631, 1973.
8. N. Nisan and A. Ronen. Algorithmic Mechanism Design. In *Proc. of the 31st STOC*, pages 129–140, 1999.
9. N. Nisan and A. Ronen. Computationally Feasible VCG Mechanisms. In *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC)*, pages 242–252, 2000.
10. C. H. Papadimitriou. Algorithms, Games, and the Internet. In *Proc. of the 33rd STOC*, 2001.
11. A. Ronen. *Solving Optimization Problems Among Selfish Agents.* PhD thesis, Hebrew University in Jerusalem, 2000.
12. W. Vickrey. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, pages 8–37, 1961.