ARTICLE IN PRESS

Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

Theoretical Computer Science

www.elsevier.com/locate/tcs

Truthful algorithms for scheduling selfish tasks on parallel machines☆

Eric Angel, Evripidis Bampis, Fanny Pascual*

LaMI–Université d'Évry Val d'Essonne, CNRS UMR 8042, 523 Place des Terrasses, 91000 Évry, France

Received 18 January 2006; received in revised form 7 July 2006; accepted 28 July 2006

Communicated by M. Mavronicolas

Abstract

We consider the problem of designing truthful mechanisms for scheduling *selfish tasks* (*or agents*)—whose objective is the minimization of their completion times—on parallel identical machines in order to minimize the *makespan*. A truthful mechanism can be easily obtained in this context (if we, of course, assume that the tasks cannot shrink their lengths) by scheduling the tasks following the increasing order of their lengths. The quality of a mechanism is measured by its approximation factor (price of anarchy, in a distributed system) w.r.t. the social optimum. The previous mechanism, known as SPT, produces a $(2 - 1/m)$-approximate schedule, where m is the number of machines. The central question in this paper is the following: "*Are there other truthful mechanisms with better approximation guarantee* (*price of anarchy*) *for the considered scheduling problem*?" This question has been raised by Christodoulou et al. [Coordination mechanisms, in: Proc. of ICALP 2004, Lecture Notes in Computer Science, Vol. 3142, 345–357.] in the context of coordination mechanisms, but it is also relevant in centrally controlled systems. We present (randomized) truthful mechanisms for both the centralized and the distributed settings that improve the (expected) approximation guarantee (price of anarchy) of the SPT mechanism. Our centralized mechanism holds for any number of machines and arbitrary task lengths, while the coordination mechanism holds only for two machines and task lengths that are powers of a certain constant.

Keywords: Algorithmic game theory; Truthful algorithm; Scheduling; Coordination mechanism; Mechanism design

1. Introduction

The Internet is a complex distributed system where many entities wish to maximize their own profits. Protocols organize this network, and their aim is to maximize the social welfare. The underlying assumption is that the agents on which the protocols are applied are trustworthy. This assumption is unrealistic in some settings as the agents might try to manipulate the protocol by reporting false information in order to get some advantages. 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 single entities.

☆ A short version of this paper has been published in the first Workshop on Internet and Network Economics (WINE 2005), LNCS 3828.

* Corresponding author. Tel.: +33 6 88577716.

E-mail addresses: angel@lami.univ-evry.fr (E. Angel), bampis@lami.univ-evry.fr (E. Bampis), fpascual@lami.univ-evry.fr (F. Pascual).

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.07.057

In this paper, we deal with the problem of scheduling tasks on parallel identical machines in order to minimize the *makespan*, (this problem is also known as $P\|C_{\max}$). There are m identical machines and n tasks of arbitrary lengths, where each task is owned by an *agent*. The lengths of the tasks are known to their owners only.

In the first part of the paper, we focus on the following process: at first the agents declare their lengths; then, given these bids, the system allocates the tasks to the machines. The objective of the system is to minimize the makespan, i.e. the date at which the last task finishes its execution. The aim of each agent is to minimize its completion time and thus an agent may lie if by doing so, she can improve its completion time.

The field of Mechanism Design can be useful to deal with the selfishness of the agents. Its main idea is to pay the agents to convince them to perform strategies that help the system to optimize a global objective function. The most famous technique for designing truthful mechanisms is perhaps the Vickrey-Clarke-Groves (VCG) mechanism [10,12,18]. However, when applied to combinatorial optimization problems, this mechanism guarantee the truthfulness under the hypothesis that the objective function is *utilitarian* (i.e. the objective function is equal to the sum of the agents' valuation) and that the mechanism is able to compute the optimum (for instance, it works for the shortest path problem [17]). Archer and Tardos introduce in [2] a method which allows to design truthful mechanisms for several combinatorial optimization problems to which the VCG mechanism does not apply. However, neither approach can be applied to our problem and thus we design a new ad hoc mechanism that is able to retain truthfulness.

In the second part of the paper, we change our setting and we are interested to the development of a truthful *coordination mechanism* [9] for the same scheduling problem. The notion of coordination mechanism has been introduced in order to improve the performance of a system with independent selfish and non-colluding agents. In a *coordination mechanism*, we assume that the system designer can select the scheduling policies of each machine (e.g. each machine schedules its tasks in order of decreasing lengths), but the designer must design the system once and for all (i.e. it should not depend on the values bidded by the tasks). Another important and natural condition is the decentralized nature of the problem: the scheduling on a machine should depend only on the lengths of the tasks assigned to it and should be independent of the tasks' lengths assigned to the other machines. Knowing the coordination mechanism and the values bidded by the other tasks, each task chooses on which machine it will be scheduled, and is then scheduled on this machine according to the policy of the machine.

A truthful mechanism can be easily obtained (if we, of course, assume that the tasks cannot shrink their lengths) by scheduling the tasks following the increasing order of their lengths. This mechanism can also be adapted to a truthful coordination mechanism. This mechanism, known as SPT, produces a $(2-1/m)$-approximate schedule. The central question in this paper is the following: "*Are there other truthful mechanisms with better approximation guarantee* (*price of anarchy*) *for the considered scheduling problem*?"

1.1. Results in this paper

Since there is no deterministic truthful mechanism with an approximation ratio better than the one of SPT, we focus, like in [2], on randomized truthful mechanisms. Thus, we assume that each agent aims to maximize her *expected* profit. A mechanism is then called *truthful* if, for each agent, bidding her true schedule length maximizes her expected profit regardless of what the other agents bid.

In Section 3, we consider the selfish task allocation model and we give a centralized algorithm which is truthful even if the values of the lengths are not restricted, and has an expected approximation ratio of $2-\frac{1}{m+1}(\frac{5}{3}+\frac{1}{3m})$, which is smaller than the one of an SPT schedule (e.g. if $m=2$ its approximation ratio is smaller than 1.39 whereas it is 1.5 for an SPT schedule).

In Section 4, we consider the two-machines case. We first study a *coordination mechanism* in which the first machine always schedules its tasks in order of increasing lengths, and the second machine schedules its tasks with a probability $p \geqslant \frac{2}{3}$ in order of increasing lengths and with probability $(1-p)$ in order of decreasing lengths. The expected approximation ratio of this (randomized) coordination mechanism, that we prove to be $\frac{4}{3}+\frac{p}{6}$, is better than the one of SPT ($\frac{3}{2}$). We show that this coordination mechanism is truthful if the tasks are powers of a constant larger than or equal to $\frac{4-3p}{2-p}$, but not if the values of the task lengths are not restricted. We also show that if $p<\frac{1}{2}$ then this coordination mechanism is not truthful even if the tasks are powers of any integer larger than 1. In Section 4.3, we consider some other randomized coordination mechanisms based on the deterministic coordination mechanisms in which the tasks

Please cite this article as: Eric Angel et al., Truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science (2006), doi: 10.1016/j.tcs.2006.07.057

are scheduled in order of increasing or decreasing lengths (and thus which have expected approximation ratios better than the one of SPT), and give negative results on their truthfulness.

1.2. Related works

Scheduling with selfish agents have been intensively studied these last years, starting with the seminal work of Nisan and Ronen [17], and followed by a series of papers [1–5,7,15]. However, all these works differ from our paper since in their case, the selfish agents were the machines while here we consider that the agents are the tasks. Furthermore, in most of these works the mechanisms have to give a payment to the agents in order to induce them to report their true private values, whereas we wish in this paper to design algorithms which are truthful without giving a payment to the tasks.

However there are some similarities among these works and ours. For example, in [4], the authors study the case where the agents can lie in one direction (that is they either overbid or underbid). However they study this for one-parameter agents (which can be used when the agents are the machines, but not when they are the tasks), and in the context of mechanism with verification (which implies that the mechanism can recognize the true value of an agent, and can punish her). The impact of restrictions on the values that the agents can bid has also been considered in other contexts, such as allocation of goods to competing requests [13,16]. Several works also consider truthful mechanisms if the tasks or the agents values are restricted: Kovacs proved in [15] that the LPT algorithm is truthful and 3-approximate if the machine's speeds are powers of two, and Ambrosio et al. proved in [1] that LPT is not truthful if the machine's speeds are powers of a constant smaller than 1.78.

A more related work is the one of Auletta et al. who considered in [6] the problem of scheduling selfish tasks in a centralized case. Their work differs from ours since they considered that each machine uses a round and robin policy and thus that the completion of each task is the completion time of the machine on which the task is (this model is known as the KP model). They considered that the tasks can lie in both directions, and that there are some payments.

Another closely related work is the one of Christodoulou et al. [9] who considered the model that we study here, but only in the distributed context of coordination mechanisms. They proposed different coordination mechanisms with a price of anarchy better than the one of the SPT mechanism. Nevertheless, these mechanisms are not truthful. In [14], the authors gave coordination mechanisms for the same model for related machines (i.e. machines can have different speeds), but their mechanisms are also not truthful.

2. Preliminaries

We are given m machines (or processors) and n tasks $T_1, \ldots, T_n$. Let l_i denote the execution time (or length) of task T_i. We use the identification numbers to compare tasks of the same lengths: we will say that a task T_i is larger than a task T_j if and only if $l_i > l_j$ or ($l_i = l_j$ and $i > j$). The machines have the same speed, and the length of each task is known by an agent, its owner. Each agent declares a value b greater than or equal to the real length of the task (we make the assumption, like in [9] that the agents cannot shrink their lengths). The aim of each agent is to minimize its completion time, and an agent may lie if by doing so she can improve its completion time.

We consider two different models of execution:

- in the first one, used in Section 3, if T_i bids a value $b > l_i$, then its execution time remains l_i,
- in the second one, used in Section 4, we assume that if T_i bids a value $b > l_i$, then its execution time is b, i.e. T_i (or its owner) will not get the result of its execution before b time units after the beginning of the execution of T_i.

This model of execution is called the *weak model of execution* in what follows.

We adopt the following definition of *randomized mechanism*: A randomized mechanism can be seen as a probability distribution over deterministic mechanisms; for instance given two deterministic mechanisms $M1$ and $M2$, with a probability p the mechanism will be $M1$ and with probability $(1 - p)$ it will be $M2$.

In the *centralized setting* (Section 3), the schedule will be obtained as follows: given the randomized mechanism, the agents will declare their lengths and the system will assign them to the machines following the deterministic mechanism $M1$ with probability p or $M2$ with probability $(1 - p)$.

In the *distributed setting* (Section 4), given the randomized mechanism, each task bids a value which represents its length, and then the selected deterministic coordination mechanism is announced to the tasks (it is $M1$ with probability

Please cite this article as: Eric Angel et al., Truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science (2006), doi: 10.1016/j.tcs.2006.07.057

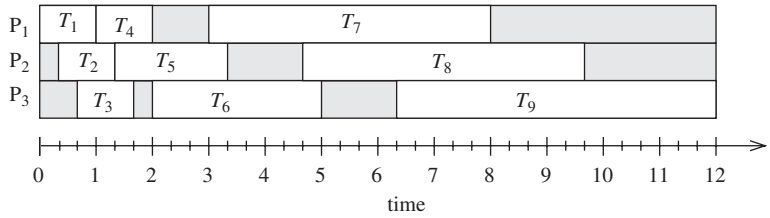

Fig. 1. DSPT schedule.

p and $M2$ with probability $(1-p)$). Each task chooses on which processor it will be scheduled, according to the policies of the processors: it goes on the processor on which it will minimize its expected completion time.

We say that a (randomized) mechanism is truthful if for every task the expected completion time when it declares its true length is smaller than or equal to its expected completion time in the case where it declares a larger value. More formally, we say that a mechanism M is *truthful* if $E_i(l_i) \leqslant E_i(b_i)$, for every i and $b_i \geqslant l_i$, where $E_i(b_i)$ is the expected completion time of task T_i if it declares b_i. In order to evaluate the quality of a randomized mechanism, we use the notion of expected approximation ratio (price of anarchy).

We will refer in the sequel to the list scheduling algorithms LPT and SPT, where LPT (resp. SPT) [11] is the algorithm which greedily schedules the tasks, sorted in order of decreasing (resp. increasing) lengths, as soon as a machine is available. An LPT (resp. SPT) schedule is a schedule returned by the LPT (resp. SPT) algorithm.

3. Truthful centralized mechanism

We give in this section a truthful randomized mechanism for the centralized setting. This mechanism is obtained by using a new deterministic (truthful) algorithm, which will be used with a certain probability p, and the LPT algorithm (which is not truthful but has a good approximation ratio), with a probability $(1-p)$.

3.1. Algorithm: LDS

Let us consider the following algorithm, denoted by DSPT (for Delayed SPT), and which returns an SPT schedule in which a delay (or idle time) may have been inserted before each task:

> Let $\{T_1, T_2, \ldots, T_n\}$ be n tasks to be scheduled on $m \geqslant 2$ identical processors, $\{P_1, P_2, \ldots, P_m\}$. Let us suppose that $l_1 \leqslant l_2 \leqslant \cdots \leqslant l_n$.
> Tasks are scheduled alternatively on $P_1, P_2, \ldots, P_m$, in order of increasing length, and T_{i+1} starts to be executed when exactly $\frac{1}{m}$ of task T_i has been executed. Thus T_1 starts to be scheduled on P_1 at time 0, T_2 is scheduled on P_2 at time $\frac{l_1}{m}$, T_3 is scheduled on P_3 (on P_1 if $m = 2$) when $\frac{1}{m}$ of T_2 has been executed, i.e. at time $\frac{l_1}{m} + \frac{l_2}{m}$, and so forth...

The schedule returned by DSPT will be called a DSPT schedule in the sequel. Fig. 1 shows a DSPT schedule, where $m = 3$.

Lemma 1. *Let us suppose that we have n tasks $\{T_1, \ldots, T_n\}$ to schedule on m machines $\{P_1, \ldots, P_m\}$. Let idle(i) be the idle time added by algorithm DSPT before task i. For each $i \in \{1, \ldots, n\}$, we have idle$(i) \geqslant 0$.*

Proof. Algorithm DSPT adds, at each step $i \in \{1, \ldots, n\}$, task T_i on processor $P_{i \bmod m}$. Let us show that the idle time before each task is larger than or equal to 0. It is trivial that $idle(i) \geqslant 0$ for the m tasks scheduled in the first position. Let T_i be a task which is not scheduled in the first position (i.e. T_i is scheduled after task T_{i-m}). Task T_i starts to be executed at exactly $\frac{1}{m}$ of the execution of task T_{i-1}, which starts to be executed at exactly $\frac{1}{m}$ of the execution of $T_{i-2}, \ldots, T_{i-m+1}$ which starts to be executed at exactly $\frac{1}{m}$ of the execution of T_{i-m}. Thus T_i starts to be executed $\frac{1}{m}(l_{i-1} + l_{i-2} + \cdots + l_{i-m})$ time units after the beginning of T_{i-m}. Since $l_{i-1} \geqslant l_{i-2} \geqslant \cdots \geqslant l_{i-m}$, $\frac{1}{m}(l_{i-1} + l_{i-2} + \cdots + l_{i-m}) \geqslant l_{i-m}$, and so $idle(i) \geqslant 0$. □

Please cite this article as: Eric Angel et al., Truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science (2006), doi: 10.1016/j.tcs.2006.07.057

Theorem 1. *DSPT is* $2-\frac{1}{m}$*-approximate*: *the makespan of a* DSPT *schedule is smaller than or equal to* $(2-\frac{1}{m})OPT$, *where OPT is the makespan of an optimal schedule for the same tasks.*

Proof. We have n tasks $T_1,\ldots,T_n$, such that $l_1\leqslant\cdots\leqslant l_n$, to schedule on m processors. Each task T_i starts to be executed exactly when $\frac{1}{m}$ of T_{i-1} has been executed. So, if $n\leqslant m$, then the makespan of the DSPT schedule is: $\frac{1}{m}(l_1+\cdots+l_{n-1})+l_n\leqslant\frac{1}{m}(n-1)l_n+l_n\leqslant\frac{(2m-1)l_n}{m}\leqslant(2-\frac{1}{m})l_n\leqslant(2-\frac{1}{m})OPT$, since $l_n\leqslant OPT$.

Let us now consider the case where $n>m$. Let $i\in\{m+1,\ldots,n\}$. Task T_i starts to be executed when $\frac{1}{m}$ of T_{i-1} is executed, and T_{i-1} started to be executed when $\frac{1}{m}$ of T_{i-2} was executed, etc., $T_{(i-m)+1}$ started to be executed when $\frac{1}{m}$ of T_{i-m} was executed. So the idle time between T_i and T_{i-m} is $idle(i)=\frac{1}{m}(l_{i-m}+l_{i-m+1}+\cdots+l_{i-1})-l_{i-m}$.

Let $i\in\{2,\ldots,m\}$. The idle time before T_i is equal to $idle(i)=\frac{1}{m}(l_1+\cdots+l_{i-1})$, and there is no idle time before T_1, which starts to be executed at time 0. Thus, the sum of the idle times between tasks is $\sum_{i=2}^{n}idle(i)=\frac{1}{m}((m-1)l_{n-m+1}+(m-2)l_{n-m+2}+\cdots+l_{n-1})$.

Let $j\in\{n-m+1,\ldots,n-1\}$. Let $end(j)$ be the idle time in the schedule after the end of task T_j and before the end of T_n: $end(j)=l_{j+1}-\frac{m-1}{m}l_j+end(j+1)$, where $end(n)=0$. So the sum of the idle times after the last tasks and before the end of the schedule is $\sum_{j=n-m+1}^{n-1}end(j)=(m-1)(l_n-\frac{m-1}{m}l_{n-1})+(m-2)(l_{n-1}-\frac{m-1}{m}l_{n-2})+\cdots+(l_{n-m+2}-\frac{m-1}{m}l_{n-m+1})$.

The sum of the idle times on the processors, from the beginning of the schedule until the makespan, is the sum of the idle times between tasks (and before the first tasks), plus the sum of the idle times after the end of the last task of a processor and before the makespan. It is equal to $\sum_{i=2}^{n}idle(i)+\sum_{j=n-m+1}^{n-1}end(j)=\frac{1}{m}((m-1)l_{n-m+1}+(m-2)l_{n-m+2}+\cdots+l_{n-1})+(m-1)(l_n-\frac{m-1}{m}l_{n-1})+(m-2)(l_{n-1}-\frac{m-1}{m}l_{n-2})+\cdots+(l_{n-m+2}-\frac{m-1}{m}l_{n-m+1})=(m-1)l_n$.

Let ξ be the makespan of a DSPT schedule. ξ is the sum of the tasks plus the sum of the idle times, divided by m: $\xi=\frac{(\sum_{i=1}^{n}l_i)+(m-1)l_n}{m}=\frac{\sum_{i=1}^{n}l_i}{m}+\frac{(m-1)l_n}{m}$. Since $\frac{\sum_{i=1}^{n}l_i}{m}\leqslant OPT$ and $l_n\leqslant OPT$, we have: $\xi\leqslant(2-\frac{1}{m})OPT$. □

Let us consider the following algorithm, denoted by *LDS* in the sequel:

> Let m be the number of processors. With a probability $\frac{m}{m+1}$, the output schedule is a DSPT schedule; and with a probability $\frac{1}{m+1}$, the output schedule is an LPT schedule.

Theorem 2. *The expected approximation ratio of LDS is* $2-\frac{1}{m+1}(\frac{5}{3}+\frac{1}{3m})$.

Proof. The approximation ratio of a DSPT schedule is $2-\frac{1}{m}$ (see Theorem 1), and the approximation ratio of an LPT schedule is $\frac{4}{3}-\frac{1}{3m}$ (see [11]). Thus the expected approximation ratio of *LDS* is $\frac{m}{m+1}(2-\frac{1}{m})+\frac{1}{m+1}(\frac{4}{3}-\frac{1}{3m})=\frac{1}{m+1}(2m-1+\frac{4}{3}-\frac{1}{3m})=\frac{1}{m+1}(2(m+1)-\frac{5}{3}-\frac{1}{3m})=2-\frac{1}{m+1}(\frac{5}{3}+\frac{1}{3m})$. □

3.2. Truthfulness

Theorem 3. *The algorithm LDS is truthful.*

Proof. Let us suppose that we have n tasks $T_1,\ldots,T_n$, ordered by increasing lengths, to schedule on m processors. Let us show that any task T_i does not have incentive to bid a length higher than its true length. Let us suppose that task T_i bids $b>l_i$, and that, by bidding b, T_i is now larger than all the tasks $T_1,\ldots,T_x$, and smaller than T_{x+1}. In the LPT schedule, the tasks T_{x+1} to T_n are scheduled in the same way, whatever T_i bids (l_i or b). By bidding b, T_i can, at best, start $(l_{i+1}+\cdots+l_x)$ time units before than if it had bidded l_i. Thus the expected completion time of T_i in *LDS* decreases by at most $\frac{1}{m+1}(l_{i+1}+\cdots+l_x)$ time units when T_i bids b instead of l_i.

On the other hand, by bidding b instead of l_i, T_i will end later in the DSPT schedule: in this schedule, tasks from T_{i+1} to T_x will be started before T_i. Since a task T_j starts to be scheduled when $\frac{1}{m}$ of its predecessor T_{j-1} is executed, by bidding b, T_i starts $\frac{1}{m}(l_{i+1}+\cdots+l_x)$ time units later than if it had bidded l_i. Thus, the expected completion time of T_i in *LDS* is increased by $\frac{m}{m+1}(\frac{1}{m}(l_{i+1}+\cdots+l_x))=\frac{1}{m+1}(l_{i+1}+\cdots+l_x)$.

Please cite this article as: Eric Angel et al., Truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science (2006), doi: 10.1016/j.tcs.2006.07.057

Thus, as a whole, the expected completion time of T_i cannot decrease when T_i bids a higher value than l_i, and we can deduce that *LDS* is truthful. □

No better approximation ratio can be achieved by choosing probabilities for DSPT and LPT in a different way. Indeed, if we have $m \geqslant 2$ machines, $m-1$ tasks of length m and m tasks of length 1, then, by bidding $1+\varepsilon$ (where ε is a negligible positive value), the task of length 1 which is the last one in the LPT schedule will gain $(m-1)$ time units in the LPT schedule, whereas it will loose $\frac{1}{m}(m-1)$ time units in the DSPT schedule.

Note that in the case where $m=2$, the expected approximation ratio of *LDS* is $\frac{25}{18} < 1.39$. This algorithm is truthful, even in the case where the tasks can take any value, and it has a better approximation ratio than $SSL(p)$ introduced in Section 4 (but *LDS* is not a coordination mechanism because a processor has to know the tasks scheduled on the other processors).

We can also note that, since the approximation ratio of a DSPT schedule is $2-\frac{1}{m}$ (like SPT), and the approximation ratio of an LPT schedule is $\frac{4}{3}-\frac{1}{3m}$, the schedule returned by *LDS* is, in the worst case, $2-\frac{1}{m}$-approximate, which is not worse than the approximation ratio of an SPT schedule.

4. Truthful coordination mechanisms

We focus in this section on truthful coordination mechanisms, in the case where there are two machines.

4.1. Coordination mechanism: $SSL(p)$

Let us first consider the following algorithm, denoted by $SSL(p)$ in the sequel:

> Let $p \in \mathbb{R}$ such that $0 \leqslant p \leqslant 1$. With a probability p, the output schedule is an SPT schedule: the tasks are greedily scheduled in order of increasing lengths. With a probability $(1-p)$, the output schedule is an SPT–LPT schedule: an SPT–LPT schedule is a schedule in which a processor, denoted by P_{SPT}, schedules the tasks in order of increasing lengths, and the other processor, denoted by P_{LPT}, schedules the tasks in order of decreasing lengths. A task T_i is scheduled on P_{SPT} if the total length of the tasks smaller than T_i is smaller than or equal to the total length of the tasks larger than T_i; otherwise it is scheduled on P_{LPT}.

We can easily transform the centralized algorithm $SSL(p)$ into a (randomized) coordination mechanism. Indeed, we can obtain, as showed in [9], an SPT–LPT schedule by having a processor, P_{SPT}, which schedules its tasks in order of increasing sizes and the other processor, P_{LPT}, which schedules its tasks in order of decreasing sizes. Thus, each task T_i will go on P_{SPT} if the total length of the tasks smaller than T_i is smaller than or equal to the total length of the tasks larger than T_i; otherwise T_i will have incentive to go on P_{LPT}. Likewise, we can obtain an SPT schedule by having two processors P_1 and P_2 which schedule tasks in order of increasing sizes, and P_2 which adds a little idle time ε (which we know to be smaller than the length of any task) before its first task, at the very beginning of the schedule. In this way, the smallest task will go on P_1, the second smallest on P_2, and so forth, and we will get the only possible Nash equilibrium, which is an SPT schedule. Hence, the coordination mechanism corresponding to $SSL(p)$ is the following one:

> Let $p \in \mathbb{R}$ such that $0 \leqslant p \leqslant 1$. Let ε be a small number smaller than the length of every task. The first processor P_1 schedules, starting at time 0, its tasks in order of increasing sizes. The second processor P_2 schedules with a probability p its tasks in order of increasing sizes, starting its first task at time ε; and P_2 schedules, with a probability $(1-p)$, its tasks in order of decreasing sizes, starting its first task at time 0.

Theorem 4. *The expected approximation ratio of $SSL(p)$ is $\frac{4}{3}+\frac{p}{6}$.*

Proof. The approximation ratio of an SPT schedule is $\frac{3}{2}$ (see [11]), and the approximation ratio of an SPT–LPT schedule is $\frac{4}{3}$ (see [9]). Thus the expected approximation ratio of $SSL(p)$ is $p\frac{3}{2}+(1-p)\frac{4}{3}$, i.e. $p(\frac{3}{2}-\frac{4}{3})+\frac{4}{3}=\frac{4}{3}+\frac{p}{6}$. □

Please cite this article as: Eric Angel et al., Truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science (2006), doi: 10.1016/j.tcs.2006.07.057

4.2. Truthfulness

In this section, we will use the weak model of execution, as explained in the Preliminaries. When we assume that all the lenghts of the tasks are powers of a constant C, then we assume that a task can only bid a value which is a power of C. If it was not the case (i.e. if a task bids a value which is not a power of C), we could round the value of this task to the nearest higher power of C.

Theorem 5. *Let $p \in \mathbb{R}$ and such that $\frac{2}{3} \leqslant p \leqslant 1$. Algorithm $SSL(p)$ is truthful if the lengths of the tasks are powers of any constant $C \geqslant \frac{4-3p}{2-p}$.*

Proof. Let us suppose that we have n tasks to schedule, and that we know that these tasks are powers of C (and thus that they have to bid a value which is a power of C). Let us suppose that a task T_i, of length l_i, bids l_k ($l_k > l_i$). Let us show that the expected completion time of T_i is smaller when T_i bids l_i rather than l_k. Let $\Gamma = \{T_1, \ldots, T_i, \ldots, T_k, \ldots, T_{n+1}\}$ be $n+1$ tasks (the n tasks that we have to schedule, plus a task T_k of length l_k which represents the task T_i which bids l_k instead of l_i), and let us suppose that $l_1 \leqslant \cdots \leqslant l_i \leqslant \cdots \leqslant l_k \leqslant \cdots \leqslant l_{n+1}$. If T_i bids l_i then the tasks we have to schedule are the tasks $\Gamma \backslash T_k$; if T_i bids l_k, then the tasks to be scheduled are $\Gamma \backslash T_i$ (thus T_k represents T_i in this case). $SSL(p)$ is truthful if, for every i, the expected completion time of T_i is smaller if it bids l_i than if it bids any other value $l_k > l_i$.

Thus, this algorithm is truthful if the *worst* expected completion time of T_i when it bids l_i is always smaller than the *best* expected completion time of T_i when it bids $l_k > l_i$. The worst completion time of T_i which bids l_i in an SPT schedule is $\frac{\sum_{j=1}^{i-1} l_j}{2} + l_i$: this is the case when T_i starts to be executed when all the smaller tasks have already been completed. The best completion time of T_i which bids l_k in an SPT schedule is $\frac{(\sum_{j=1}^{k} l_j) - l_i}{2}$: this is the case when T_k is completed at the same time as T_{k-1}.

There are two cases for T_i in the SPT–LPT schedule: it is either scheduled on P_{SPT} after the tasks which are smaller than l_i, and ends at time $\sum_{j=1}^{i} l_j$ (case 1), or it is scheduled on P_{LPT} after the tasks which are larger than l_i, and then ends at time $(\sum_{j=i}^{n+1} l_j) - l_k$ (case 2). It is the same thing in the case where T_i bids l_k: T_k is either scheduled on P_{SPT} and then ends at time $(\sum_{j=1}^{k} l_j) - l_i$ (case A), or it is scheduled on P_{LPT} and then ends at time $\sum_{j=k}^{n+1} l_j$ (case B). In the SPT–LPT schedule, T_i (resp. T_k) chooses between the cases 1 and 2 (resp. the cases A and B) the one that minimizes its completion time.

$SSL(p)$ is truthful if the worst completion time of T_i which bids l_i in an SPT schedule, times p, plus the completion time of T_i which bids l_i in an SPT–LPT schedule, times $(1-p)$, is smaller than the best completion time of T_i which bids l_k (T_i is then identified by T_k) in an SPT schedule, times p, plus the completion time of T_k in an SPT–LPT schedule, times $(1-p)$. Thus, $SSL(p)$ is truthful if

$$p\left(\frac{\sum_{j=1}^{i-1} l_j}{2} + l_i\right) + (1-p)\left(\min\left\{\sum_{j=1}^{i} l_j \;;\; \left(\sum_{j=i}^{n+1} l_j\right) - l_k\right\}\right)$$

$$\leqslant p\left(\frac{\left(\sum_{j=1}^{k} l_j\right) - l_i}{2}\right) + (1-p)\left(\min\left\{\left(\sum_{j=1}^{k} l_j\right) - l_i \;;\; \sum_{j=k}^{n+1} l_j\right\}\right)$$

$$\Leftrightarrow (1-p)\left(\min\left\{\sum_{j=1}^{i} l_j \;;\; \sum_{j=i}^{n+1} l_j - l_k\right\}\right) \leqslant p\frac{\left(\sum_{j=i}^{k} l_j\right) - 3l_i}{2} + (1-p)\left(\min\left\{\sum_{j=1}^{k} l_j - l_i \;;\; \sum_{j=k}^{n+1} l_j\right\}\right).$$

Please cite this article as: Eric Angel et al., Truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science (2006), doi: 10.1016/j.tcs.2006.07.057

There are now four cases to consider (the four combinations of the two choices of T_i and the two choices of T_k):

- *Case* 1A: In the SPT–LPT schedule, T_i is scheduled on P_{SPT}, and T_k is scheduled on P_{SPT}.

In the SPT schedule, T_k does not end before T_i because $l_k > l_i$ and the lengths of the other tasks are the same ones. Likewise, in the SPT–LPT schedule, since T_i and T_k are both scheduled on P_{SPT}, T_k cannot end before T_i. So the expectation of the completion time of T_k is not smaller than the one of T_i and $SSL(p)$ is truthful in this case.

- *Case* 2A: In the SPT–LPT schedule, T_i is scheduled on P_{LPT}, and T_k is scheduled on P_{SPT}.

This case cannot happen. Indeed, if T_i is scheduled on P_{LPT}, then the length of the tasks before it on P_{LPT} ($\sum_{i+1}^{n+1} l_j - l_k$) is smaller than the length of the tasks which would be scheduled before it on P_{SPT} ($\sum_{1}^{i-1} l_j$). Since $l_k > l_i$, the length of the tasks scheduled before T_k on P_{LPT} ($\sum_{k+1}^{n+1} l_j = (\sum_{i+1}^{n+1} l_j - l_k) - \sum_{i+1}^{k}$) is smaller than the length of the tasks scheduled before T_k on P_{SPT} ($\sum_{1}^{k-1} l_j - l_i = \sum_{1}^{i-1} l_j + \sum_{i+1}^{k-1}$). Thus, the completion time of T_k is smaller on P_{LPT} rather than on P_{SPT}, and the case where T_i is scheduled on P_{LPT} and T_k on P_{SPT} does not occur.

- *Case* 2B: In the SPT–LPT schedule, T_i is scheduled on P_{LPT}, and T_k is scheduled on P_{LPT}.

$SSL(p)$ is truthful if

$$(1-p)\left(\left(\sum_{j=i}^{n+1} l_j\right) - l_k\right) \leqslant p\left(\frac{\left(\sum_{j=i}^{k} l_j\right) - 3l_i}{2}\right) + (1-p)\sum_{j=k}^{n+1} l_j$$

$$\Leftrightarrow (1-p)\left(\left(\sum_{j=i}^{k} l_j\right) - 2l_k\right) \leqslant p\left(\frac{\left(\sum_{j=i}^{k} l_j\right) - 3l_i}{2}\right)$$

$$\Leftrightarrow \left(\sum_{j=i}^{k} l_j\right) - 2l_k \leqslant p\left(\frac{(3\sum_{j=i}^{k} l_j) - 3l_i - 4l_k}{2}\right)$$

$$\Leftrightarrow \frac{3p}{2} l_i \leqslant 2(1-p)l_k + \sum_{j=i}^{k} l_j\left(\frac{3p}{2} - 1\right).$$

There is necessarily a task between T_i and T_k in Γ, otherwise T_i (i.e. T_i which bids l_i) and T_k (i.e. T_i which bids l_k) would start after the same tasks - and so at the same time - in the SPT schedule (and in the SPT–LPT schedule), and then T_i would not decrease its expected completion time by lying on its execution time. So, $\sum_{j=i}^{k} l_j \geqslant 2l_i + l_k$.

Since $p \geqslant \frac{2}{3}$, we know that $(\frac{3p}{2} - 1) \geqslant 0$. Thus, $SSL(p)$ is truthful if

$$\frac{3p}{2} l_i \leqslant 2(1-p)l_k + (2l_i + l_k)\left(\frac{3p}{2} - 1\right)$$

$$\Leftrightarrow \left(\frac{3p}{2} - 2\left(\frac{3p}{2} - 1\right)\right) l_i \leqslant \left(2(1-p) + \left(\frac{3p}{2} - 1\right)\right) l_k$$

$$\Leftrightarrow \left(2 - \frac{3p}{2}\right) l_i \leqslant \left(1 - \frac{p}{2}\right) l_k$$

$$\Leftrightarrow l_i \leqslant \frac{1 - \frac{p}{2}}{2 - \frac{3p}{2}} l_k.$$

$$\Leftrightarrow l_i \leqslant \frac{2-p}{4-3p} l_k.$$

Since $l_k > l_i$, and tasks are powers of $C \geqslant \frac{4-3p}{2-p}$, we know that $l_k \geqslant \frac{4-3p}{2-p} l_i$, and so the above inequality is fulfilled. Thus, if $p \geqslant \frac{2}{3}$, and if the tasks are powers of $C \geqslant \frac{4-3p}{2-p}$, $SSL(p)$ is truthful.

Please cite this article as: Eric Angel et al., Truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science (2006), doi: 10.1016/j.tcs.2006.07.057

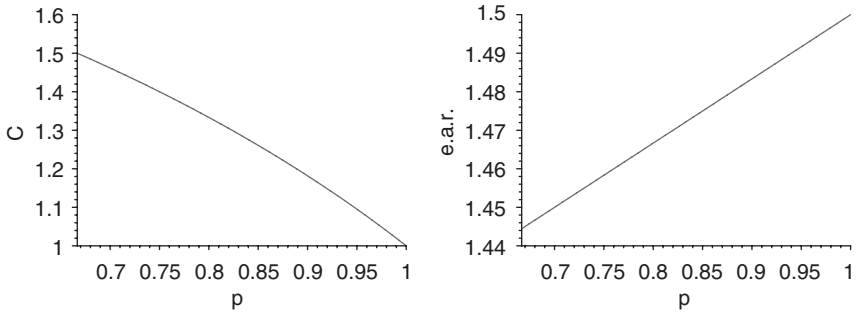

Fig. 2. Left: If the tasks are powers of a constant larger than or equal to $C(p)$ then $SSL(p)$ is truthful. Right: Expectation of the approximation ratio (e.a.r) of $SSL(p)$.

- *Case* 1B: In the SPT–LPT schedule, T_i is scheduled on P_{SPT}, and T_k is scheduled on P_{LPT}.
  $SSL(p)$ is truthful if

$$(1-p)\left(\sum_{j=1}^{i} l_j\right) \leqslant p\left(\frac{\left(\sum_{j=i}^{k} l_j\right) - 3l_i}{2}\right) + (1-p)\sum_{j=k}^{n+1} l_j.$$

In the SPT–LPT schedule, T_i is scheduled on P_{SPT} and not on P_{LPT}, so $\sum_{j=1}^{i} l_j \leqslant (\sum_{j=i}^{n+1} l_j) - l_k$. Thus, $SSL(p)$ is truthful if

$$(1-p)\left(\left(\sum_{j=i}^{n+1} l_j\right) - l_k\right) \leqslant p\left(\frac{\left(\sum_{j=i}^{k} l_j\right) - 3l_i}{2}\right) + (1-p)\sum_{j=k}^{n+1} l_j.$$

This inequality is the same one that in case 2B, so the end of the proof is the same as in the previous case.

Thus, in every case, if $p \geqslant \frac{2}{3}$, and if the tasks are powers of a constant larger than or equal to $\frac{4-3p}{2-p}$, $SSL(p)$ is truthful. □

Fig. 2 (Left) gives an illustration of Theorem 5: if we know that the tasks are powers of a constant larger than or equal to $C(p)$, then $SSL(p)$ is truthful. Fig. 2 (Right) illustrates Theorem 4 and shows the expected approximation ratio of $SSL(p)$.

We saw that $SSL(p)$ is truthful if the tasks are powers of $C = \frac{4-3p}{2-p}$. In fact, the only sufficient condition we have for this algorithm to be truthful is that, for every i, $l_{i+1} = l_i$ or $l_{i+1} \geqslant C \times l_i$. Thus, if we know that the lengths of the tasks belong to a set $S = \{x_1, x_2, \ldots, x_k\}$ such that for each j, $x_{j+1} \geqslant C \times x_j$, then $SSL(p)$ is truthful. However, $SSL(p)$ is not truthful if the possible values of the tasks are not restricted, and it is not truthful if $p < \frac{1}{2}$, even if the tasks are powers of any integer $B > 1$.

Theorem 6. *Let $p \in \mathbb{R}$ be any number such that $0 \leqslant p < 1$. Algorithm $SSL(p)$is not truthful if the tasks can take any value.*

Proof. Let ε be any number such that $0 \leqslant \varepsilon < 1 - p$. Let us consider the following tasks: two tasks T_1 and T_2 of length 1, a task T_3 of length $2 - \varepsilon$ and a task T_4 of length 2. If T_3 bids its true value, $2 - \varepsilon$, it is expected to end at time $p(3 - \varepsilon) + (1 - p)(4 - \varepsilon)$, because it is scheduled after a task of length 1 in the SPT schedule, and after the task of length 2 in the SPT–LPT schedule. If T_3 does not bid its true value, but bids $2 + \varepsilon$ instead, its expectation of completion time is then $p(3 + \varepsilon) + (1 - p)(2 + \varepsilon)$. Indeed, in this case T_3 is scheduled in the SPT schedule after a task of length 1, but it is scheduled at the first position on P_{LPT} in the SPT–LPT schedule.

Please cite this article as: Eric Angel et al., Truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science (2006), doi: 10.1016/j.tcs.2006.07.057

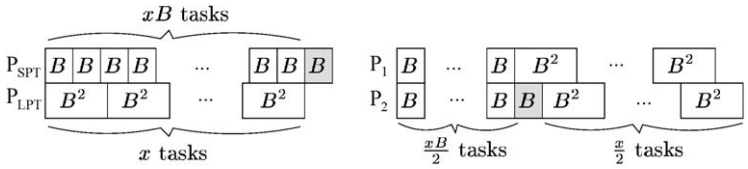

Fig. 3. Schedules returned when T, the task colored in grey, bids B. Left: SPT–LPT schedule, Right: SPT schedule. The number in each task represents its length.

Whatever the value of p is, the expected completion time of T_3 when it bids $2+\varepsilon$ instead of $2-\varepsilon$ is smaller than the expected completion time of T_3 when it bids its true value. Thus this algorithm is not truthful if we do not restrict the possible values of the tasks. □

Theorem 7. *Let $p \in \mathbb{R}$ be any number such that $0 \leqslant p < \frac{1}{2}$. Algorithm $SSL(p)$ is not truthful, even if the tasks are powers of an integer B ($B > 1$), whatever the value of B is.*

Proof. Let $B > 1$ be any integer number, and let p be any positive number smaller than $\frac{1}{2}$. Let us suppose that the tasks are powers of B (and then they must bid a power of B). Let us show that $SSL(p)$ is not truthful, by showing that there is an instance in which a task T can improve its expected completion time by bidding a value larger than its true value.

Let x be an even number larger than $\frac{B^2-1}{(B+p(\frac{1}{2}-2B))}$, and let us consider the following instance: $(xB+1)$ tasks of length B, and x tasks of length B^2. Let T be the last task of length B (i.e. the task of length B which has the largest index: $T = T_{xB+1}$ if the tasks $\{T_1, T_2, \ldots\}$ are labelled by increasing lengths). Let us now show that the expected completion time of T in $SSL(p)$ will be smaller if it bids B^3 instead of B.

The completion time of T which bids B is $\frac{xB}{2} + B$ in the SPT schedule, and $xB^2 + B$ in the SPT–LPT schedule (see Fig. 3). Thus its expected completion time in *LDS* is $p(\frac{xB}{2} + B) + (1-p)(xB^2 + B) = p(\frac{xB}{2} - xB^2) + (xB^2 + B)$.

The completion time of T which bids B^3 is $xB^2 + B^3$ in the SPT schedule (it is on the last position in this schedule, after $\frac{xB}{2}$ tasks of length B and $\frac{x}{2}$ tasks of length B^2), and the completion time of T which bids B^3 is B^3 in the SPT–LPT schedule (it is on the first position on the processor which scheduled its tasks from the largest to the smallest). Thus its expected completion time in *LDS* is $p(xB^2 + B^3) + (1-p)(B^3) = pxB^2 + B^3$.

$SSL(p)$ is not truthful if the expected completion time of T which bids B^3 is smaller than its completion time when it bids B. So it is not truthful if

$$
\begin{aligned}
& pxB^2 + B^3 < p\left(\frac{xB}{2} - xB^2\right) + (xB^2 + B) \\
& \Leftrightarrow B^3 - B < -pxB^2 + \frac{pxB}{2} - pxB^2 + xB^2 \\
& \Leftrightarrow B^3 - B < x\left(B^2 + p\left(\frac{B}{2} - 2B^2\right)\right) \\
& \Leftrightarrow B^2 - 1 < x\left(B + p\left(\frac{1}{2} - 2B\right)\right)
\end{aligned}
$$

$x > \frac{B^2-1}{(B+p(\frac{1}{2}-2B))}$, because $B + p(\frac{1}{2} - 2B) > 0$ since $0 \leqslant p < \frac{1}{2}$.

Since x is an even number larger than $\frac{B^2-1}{(B+p(\frac{1}{2}-2B))}$, x fulfills the above condition and so $SSL(p)$ is not truthful if $p < \frac{1}{2}$. □

4.3. Other coordination mechanisms: negative results

$SL(p)$ is the algorithm where we have with a probability p an SPT schedule, and with a probability $(1-p)$ an LPT schedule. $LSL(p)$ is the algorithm where we have with a probability p an LPT schedule, and with a probability $(1-p)$

Please cite this article as: Eric Angel et al., Truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science (2006), doi: 10.1016/j.tcs.2006.07.057

an SPT–LPT schedule. We saw in Section 4.1 that there exist coordination mechanisms which return an SPT or an SPT–LPT schedule. Likewise, by adding small delays on the processors—which both schedule the tasks in order of decreasing lengths—, the authors showed in [9] a coordination mechanism which returns an LPT schedule (the delays are negligible here since we can fix them as small as we want). Let us now give negative results on the truthfulness of these mechanisms.

Theorem 8. *Let $p \in \mathbb{R}$ be any number such that $0 \leqslant p < 1$. Algorithm $SL(p)$ is not truthful if the tasks can take any value.*

Proof. Let ε be any number such that $0 \leqslant \varepsilon < \frac{1-p}{1+p}$. Let us consider the following tasks: a task T_1 of length 1, a task T_2 of length $1+\varepsilon$ and a task T_3 of length 2. If T_1 bids its true value, 1, it is expected to end at time $t_1 = p + (1-p)(2+\varepsilon)$, because it is scheduled at the beginning of the SPT schedule, and after T_2 in the LPT schedule. If T_1 does not bid its true value, but bids $1+2\varepsilon$ instead, it is expected to end at time $t_2 = (1+2\varepsilon)$, because it is scheduled at the beginning of the SPT and the LPT schedules. Since $\varepsilon < \frac{1-p}{1+p}$, t_2 is smaller than t_1, and so $SL(p)$ is not truthful. □

Theorem 9. *Let $p \in \mathbb{R}$ be any number such that $0 \leqslant p < \frac{1}{2}$. Algorithm $SL(p)$ is not truthful, even if the tasks are powers of a constant B ($B > 1$), whatever the value of B is.*

Proof. Let $B > 1$ be any real number, and let p be any positive number smaller than $\frac{1}{2}$. Let us suppose that the tasks are powers of B (and then they must bid a power of B). Let us show that $SL(p)$ is not truthful, by showing that there is an instance in which a task T can improve its expected completion time by bidding a value larger than its true value.

Let $x = \lfloor \frac{B^2-1}{B(1-2p)} \rfloor + 1$, and let us consider the following instance: $2x$ tasks of length B^2, and one task, T, of length B. Let us now show that the expected completion time of T in $SL(p)$ will be smaller if it bids B^3 rather than B.

The completion time of T which bids B is B in the SPT schedule, and $xB^2 + B$ in the LPT schedule, where T is scheduled after x tasks of lengths B^2. Its expected completion time in $SL(p)$ is then $t_1 = pB + (1-p)(xB^2 + B) = B + (1-p)xB^2$. The completion time of T which bids B^3 is $xB^2 + B^3$ in the SPT schedule (it is scheduled after x tasks of lengths B^2), and B^3 in the LPT schedule. Its expected completion time in $SL(p)$ is then $t_2 = p(xB^2+B^3)+(1-p)(B^3) = B^3 + pxB^2$.

$SL(p)$ is not truthful if $t_1 < t_2$: $t_1 < t_2$

$$\Leftrightarrow B^3 + pxB^2 < B + (1-p)xB^2$$
$$\Leftrightarrow B^3 - B < x(B^2(1-2p))$$
$$\Leftrightarrow x > \frac{B^2-1}{B(1-2p)}.$$

Since $x = \lfloor \frac{B^2-1}{B(1-2p)} \rfloor + 1$ fulfills the above condition, the expected completion time of T in our instance is smaller if T bids B^3 rather than if its bids its true value B, and so $SL(p)$ is not truthful. □

Theorem 10. *Let $p \in \mathbb{R}$ be any number such that $0 \leqslant p \leqslant 1$. Algorithm $LSL(p)$ is not truthful, even if the tasks are powers of a constant B ($B > 1$), whatever the value of B is.*

Proof. Let $B > 1$ be any real number, and p be any number such that $0 \leqslant p \leqslant 1$. Let us suppose that the tasks are powers of B. Let us show that $LSL(p)$ is not truthful, by showing that there is an instance in which a task T can improve its expected completion time by bidding a value larger than its true value.

Let us consider the following instance: $2B+1$ tasks of length B, and $2B^2$ tasks of length 1. Let T be the last task of length B (i.e. the task of length B which has the largest index).

The completion time of T which bids B is $B^2 + B$ in the LPT schedule (T is scheduled after B tasks of length B), and 2, $B^2 + B$ in the SPT–LPT schedule (T is scheduled after $2B^2$ tasks of length 1). The completion time of T which bids B^2 is B^2 in the LPT schedule, and B^2 in the SPT–LPT schedule (T is scheduled at the beginning of both schedules). The expected completion time of T is smaller when T bids B^2 rather than B, so $LSL(p)$ is not truthful. □

Please cite this article as: Eric Angel et al., Truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science (2006), doi: 10.1016/j.tcs.2006.07.057

In the negative results of this section, we used the weak model of execution: we assume that if T_i bids a value $b > l_i$, then its new execution time is b. Of course, these results also hold for the second execution model, in which if T_i bids a value $b > l_i$, then its new execution time will still be l_i (T_i does not have to wait b time units after its start to get its result).

References

[1] P. Ambrosio, V. Auletta, Deterministic monotone algorithms for scheduling on related machines, in: Proc. of WAOA, 2004, pp. 267–280.
[2] A. Archer, E. Tardos, Truthful mechanisms for one-parameter agents, in: Proc. of FOCS, 2001, pp. 482–491.
[3] V. Auletta, R. De Prisco, P. Penna, P. Persiano, Deterministic truthful approximation mechanisms for scheduling related machines, in: Proc. of STACS, 2004, pp. 608–619.
[4] V. Auletta, R. De Prisco, P. Penna, P. Persiano, The power of verification for one-parameter agents, in: Proc. of ICALP, 2004, Lecture Notes in Computer Science, Vol. 3142, pp. 171–182.
[5] V. Auletta, R. De Prisco, P. Penna, P. Persiano, On designing truthful mechanisms for online scheduling, in: Proc. of SIROCCO, 2005, Lecture Notes in Computer Science, Vol. 3499, pp. 3–17.
[6] V. Auletta, P. Penna, R. De Prisco, P. Persiano, How to route and tax selfish unsplittable traffic, in: Proc. of SPAA, 2004, pp. 196–204.
[7] Y. Azar, M. Sorani, Truthful approximation mechanisms for scheduling selfish related machines, in: Proc. of STACS, 2005, pp. 69–82.
[9] G. Christodoulou, E. Koutsoupias, A. Nanavati, Coordination mechanisms, in: Proc. of ICALP 2004, Lecture Notes in Computer Science, Vol. 3142, 2004, pp. 345–357.
[10] E. Clarke, Multipart pricing of public goods, Public Choices, 1971, pp. 17–33.
[11] R. Graham, Bounds on multiprocessor timing anomalies, SIAM Appl. Math. 17 (2) (1969) 416–429.
[12] T. Groves, Incentive in teams, Econometrica 41 (4) (1973) 617–631.
[13] M.T. Hajiaghayi, R.D. Kleinberg, M. Mahdian, D.C. Parkes, Online auctions with re-usable goods, in: Proc. ACM Conf. on Electronic Commerce (EC) 2005, pp. 165–174.
[14] N. Immorlica, L. Li, V.S. Mirrokni, A. Schulz, Coordination mechanisms for selfish scheduling, in: Proc. of WINE, 2005, Lecture Notes in Computer Science, Vol. 3828, pp. 55–69.
[15] A. Kovacs, Fast monotone 3-approximation algorithm for scheduling related machines, in: Proc. of ESA, 2005, Lecture Notes in Computer Science, Vol. 3669, pp. 616–627.
[16] R. Porter, Mechanism design for online real-time scheduling, in: Proc. ACM Conf. Electronic Commerce (EC), 2004, pp. 61–70.
[17] N. Nisan, A. Ronen, Algorithmic mechanism design, in: Proc. of STOC, 1999, pp. 129–140.
[18] W. Vickrey, Counterspeculation, auctions and competitive sealed tenders, J. Finance 16 (1961) 8–37.

Please cite this article as: Eric Angel et al., Truthful algorithms for scheduling selfish tasks on parallel machines, Theoretical Computer Science (2006), doi: 10.1016/j.tcs.2006.07.057