Truthful algorithms for scheduling selfish tasks on parallel machines

Eric Angel [1], Evripidis Bampis [1] and Fanny Pascual [1]

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 [1] 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 schedule lengths, while the coordination mechanism holds only for two machines and schedule lengths that are powers of a certain constant.

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.

In this paper, we deal with the problem of scheduling tasks on parallel identical machines in order to minimize the *makespan*, (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 owner 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

[1] LaMI – Université d'Évry Val d'Essonne, CNRS UMR 8042 –, 523 Place des Terrasses, 91000 Évry, France. E-mail: {angel, bampis, fpascual}@lami.univ-evry.fr

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 [8,9,10]. 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 [3]). Archer and Tardos introduce in [4] 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* [1] 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 one 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 she will be scheduled, and she is then scheduled on it, according to the policy of this 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

Any algorithm with the property that increasing task length implies non-decreasing completion time will always be truthful, and this property is necessary for truthtelling to be a dominant strategy. Since there is no dominant strategy with an approximation ratio better than the one of SPT, we focus, like in [4], 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 $2-\frac{1}{m+1}\left(\frac{5}{3}+\frac{1}{3\,m}\right)$, 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 $p>\frac{2}{3}$ 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 $\frac{4-3\,p}{2-p}$, 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 the other randomized coordination mechanisms that combine deterministic coordination mechanisms in which the tasks 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 started with the seminal work of Nisan and Ronen [3] and followed by a series of papers [4], [6], [5], [7]. 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.

The most closely related work is the one by Christodoulou et al [1] who considered the same model 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.

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 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 [1] 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 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 $E_i(l_i) \leq E_i(b_i)$, for every i and $b_i \geq l_i$, 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).

3 Truthful centralized mechanism

We give in this section a randomized mechanism for the centralized setting. The idea is to propose a new deterministic mechanism which when combined with a mechanism scheduling the tasks in the decreasing order of their lengths provides a truthful randomized mechanism.

3.1 Algorithm: LS_δ

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

Let $\{T_1, T_2, \ldots, T_n\}$ be n tasks to be scheduled on $m \geq 2$ identical processors, $\{P_1, P_2, \ldots, P_m\}$. Let us suppose that $l_1 \leq l_2 \leq \cdots \leq l_n$.
Tasks are scheduled alternately 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 SPT$_\delta$ will be called a SPT$_\delta$ schedule in the sequel. Figure 1 shows an SPT$_\delta$ schedule, where $m = 3$.

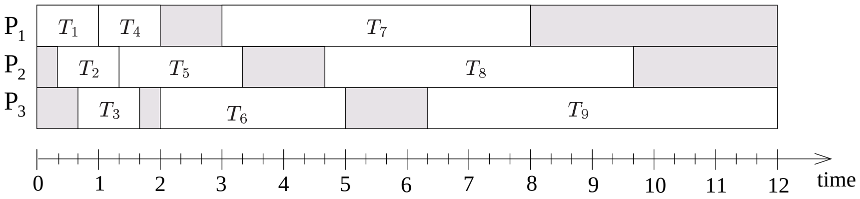

Fig. 1. SPT$_\delta$ schedule

Theorem 1 *SPT$_\delta$ is $2 - \frac{1}{m}$-approximate: the makespan of an SPT$_\delta$ 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 $l_1 \leq \cdots \leq l_n$, 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 $n \leq m$, then the makespan of the SPT$_\delta$ schedule is: $\frac{1}{m}\,(l_1 + \cdots + l_{n-1}) + l_n \leq \frac{1}{m}\,(n-1)\,l_n + l_n \leq \frac{(2m-1)\,l_n}{m} \leq (2 - \frac{1}{m})\,l_n \leq (2 - \frac{1}{m})\,OPT$, since $l_n \leq OPT$.

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 $\sum_{j=n-m+1}^{n-1} end(j) = (m-1)\left(l_n - \frac{m-1}{m}\,l_{n-1}\right) + (m-2)\left(l_{n-1} - \frac{m-1}{m}\,l_{n-2}\right) + \cdots + \left(l_{n-m+2} - \frac{m-1}{m}\,l_{n-m+1}\right)$.

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 $\sum_{i=2}^{n} idle(i)+\sum_{j=n-m+1}^{n-1} end(j) = \frac{1}{m}\left((m-1)l_{n-m+1} + (m-2)l_{n-m+2} + \cdots + l_{n-1}\right) + (m-1)\left(l_n - \frac{m-1}{m}\, l_{n-1}\right) + (m-2)\left(l_{n-1} - \frac{m-1}{m}\, l_{n-2}\right) + \cdots + \left(l_{n-m+2} - \frac{m-1}{m}\, l_{n-m+1}\right) = (m-1)\, l_n$.

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

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

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

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

Proof: The approximation ratio of an SPT$_\delta$ schedule is $2-\frac{1}{m}$ (see Theorem 1), and the approximation ratio of an LPT schedule is $\frac{4}{3} - \frac{1}{3\,m}$ (see [2]). Thus the expected approximation ratio of LS_δ is $\frac{m}{m+1}\left(2-\frac{1}{m}\right) + \frac{1}{m+1}\left(\frac{4}{3}-\frac{1}{3\,m}\right) = \frac{1}{m+1}\left(2m-1+\frac{4}{3}-\frac{1}{3m}\right) = \frac{1}{m+1}\left(2(m+1)-\frac{5}{3}-\frac{1}{3m}\right) = 2-\frac{1}{m+1}\left(\frac{5}{3}+\frac{1}{3\,m}\right)$. □

3.2 Truthfulness

Theorem 3 LS_δ *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 $(T_{i+1} + \cdots + T_x)$ time units before than if it had bidded l_i. Thus the expected completion time of T_i in LS_δ decreases by at most $\frac{1}{m+1}\,(T_{i+1} + \cdots + T_x)$ 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 SPT$_\delta$ 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 $\frac{1}{m}\,(T_{i+1} + \cdots + T_x)$ time units later than if it had bidded l_i. Thus, the expected completion time of T_i in LS_δ is increased by $\frac{m}{m+1}(\frac{1}{m}\,(T_{i+1} + \cdots + T_x)) = \frac{1}{m+1}\,(T_{i+1} + \cdots + T_x)$.
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 LS_δ is truthful. □

Note that in the case where $m = 2$, the expected approximation ratio of LS_δ 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 LS_δ 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 an SPT$_\delta$ 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 LS_δ 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

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 $0 \leq p \leq 1$. With a probability of p, the output schedule is an SPT schedule: the tasks are greedily scheduled in order of increasing length. 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 [1], 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 $0 \leq p \leq 1$. 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 [2]), and the approximation ratio of an SPT-LPT schedule is $\frac{4}{3}$ (see [1]). Thus the expected approximation ratio of $SSL(p)$ is $p\,\frac{3}{2} + (1-p)\,\frac{4}{3}$, i.e. $p(\frac{3}{2} - \frac{4}{3}) + \frac{4}{3} = \frac{4}{3} + \frac{p}{6}$. □

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 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 $\frac{2}{3} < p \leq 1$. Algorithm $SSL(p)$ is truthful if the tasks are powers of any constant $C \geq \frac{4-3p}{2-p}$.*

Proof: Let us suppose that we know that the tasks are powers of C, and thus 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 (n tasks, plus a task T_k which represents the task T_i which bids l_k instead of l_i), and let us suppose that $l_1 \leq \cdots \leq l_i \leq \cdots \leq l_k \leq \cdots \leq l_{n+1}$. If T_i bids l_i then the tasks we have to schedule are the tasks $\Gamma \setminus T_k$; if T_i bids l_k, then the tasks to be scheduled are $\Gamma \setminus T_i$ (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, it is truthful if the *worst* expected completion time of T_i when it bids l_i is always smaller than the *best* completion time of T_i when it bids $l_k > l_i$. The worst expected 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 expected 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:

$$p\left(\frac{\sum_{j=1}^{i-1} l_j}{2}+l_i\right)+(1-p)\left(\min\left\{\begin{array}{l}\sum_{j=1}^{i} l_j\\(\sum_{j=i}^{n+1} l_j)-l_k\end{array}\right.\right)$$
$$\leq p\left(\frac{(\sum_{j=1}^{k} l_j)-l_i}{2}\right)+(1-p)\left(\min\left\{\begin{array}{l}(\sum_{j=1}^{k} l_j)-l_i\\\sum_{j=k}^{n+1} l_j\end{array}\right.\right)$$
$$\Leftrightarrow (1-p)\left(\min\left\{\begin{array}{l}\sum_{j=1}^{i} l_j\\\sum_{j=i}^{n+1} l_j-l_k\end{array}\right.\right)\leq p\,\frac{(\sum_{j=i}^{k} l_j)-3\,l_i}{2}+(1-p)\left(\min\left\{\begin{array}{l}\sum_{j=1}^{k} l_j-l_i\\\sum_{j=k}^{n+1} l_j\end{array}\right.\right)$$

There are now four cases to consider (the four combinations of the two choices of T_i and the two choices of T_k). Due to space limitations, we consider these four cases in the extended version of the paper. □

Figure 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. Figure 2 *Right* illustrates Theorem 4 and shows the expected approximation ratio of $SSL(p)$.

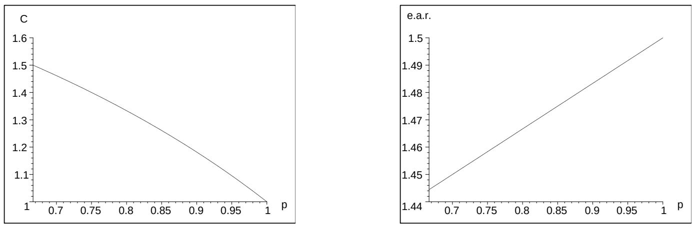

Fig. 2. *Left:* If the tasks are powers of a constant larger than or equal to $C(p)$ then $SSL(p)$ is truthful. *Right:* Expectation of the approximation ratio (e.a.r) of $SSL(p)$.

We saw that $SSL(p)$ is truthful if the tasks are powers of $C=\frac{4-3\,p}{2-p}$. 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 $l_{i+1}\geq C\times l_i$. 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, $x_{j+1}\geq C\times x_j$, 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 $B>1$ (the proofs of Theorems 6 and 7 can be found in the extended version of the paper).

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

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

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 $(1-p)$ 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 [1] a coordination mechanism which returns an LPT schedule (the delays are here negligible since we can fix them as small as we want). Let us now give negative results on the truthfulness of this mechanisms. The proofs of Theorems 8, 9 and 10 can be found in the extended version of the paper.

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

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

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

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. G. Christodoulou, E. Koutsoupias, A. Nanavati *Coordination mechanisms.* In Proc. of ICALP 2004, LNCS 3142, 345-357.
2. R. Graham *Bounds on multiprocessor timing anomalies.* SIAM Jr. on Appl. Math., 17(2), 416-429, 1969.
3. N. Nisan, A. Ronen *Algorithmic mechanism design.* In Proc. STOC 1999, 129-140.
4. A. Archer, E. Tardos *Truthful Mechanisms for One-Parameter Agents.* In Proc. of FOCS 2001, 482-491.
5. P. Ambrosio, V. Auletta *Deterministic Monotone Algorithms for Scheduling on related Machines.* In Proc. of WAOA 2004, 267-280.
6. V. Auletta, R. De Prisco, P. Penna, P. Persiano *Deterministic Truthful Approximation Mechanisms for Scheduling Related Machines.* In Proc. of STACS 2004, 608-619 .
7. Y. Azar, M. Sorani *Truthful Approximation Mechanisms for Scheduling Selfish Related Machines.* In Proc. of STACS 2005, 69-82.
8. W. Vickrey *Counterspeculation, auctions and competitive sealed tenders.* J. Finance, 16:8-37, 1961.
9. E. Clarke *Multipart pricing of public goods.* Linear programming and vickrey auctions. Unpublished manuscript, 2001.
10. T. Groves *Incentive in teams.* Econometrica, 41(4):617-631, 1973.