Truthfulness for the Sum of Weighted Completion Times

Eric Angel[1], Evripidis Bampis[2](✉), Fanny Pascual[2], and Nicolas Thibault[3]

[1] IBISC, Université d'Évry Val d'Essonne, Evry, France
[2] Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, Paris, France
evripidis.bampis@lip6.fr
[3] CRED, Université Panthéon-Assas, Paris 2, Paris, France

Abstract. We consider the problem of designing truthful mechanisms for scheduling selfish tasks on a single machine or on a set of m parallel machines. The objective of every selfish task is the minimization of its completion time while the aim of the mechanism is the minimization of the sum of weighted completion times. For the model *without payments*, we prove that there is no $(2-\epsilon)$-approximate deterministic truthful algorithm and no $(\frac{3}{2}-\epsilon)$-approximate randomized truthful algorithm when the tasks' lengths are private data. When both the lengths and the weights are private data, we show that it is not possible to get an α-approximate deterministic truthful algorithm for any $\alpha > 1$. In order to overcome these negative results we introduce a new concept that we call *preventive preemption*. Using this concept, we are able to propose a simple optimal truthful algorithm with no payments for the single-machine problem when the lengths of the tasks are private. For multiple machines, we present an optimal truthful algorithm for the unweighted case. For the weighted-multiple-machines case, we propose a truthful randomized algorithm which is $\frac{3}{2}$-approximate in expectation based on preventive preemption. For the model *with payments*, we prove that there is no optimal truthful algorithm even when only the lengths of the tasks are private data. Then, we propose an optimal truthful mechanism using preventive preemption and appropriately chosen payments.

1 Introduction

A lot of attention has been devoted to scheduling problems in the literature of algorithmic game theory starting from the seminal paper of Koutsoupias and Papadimitriou [18]. Most of these papers consider that the social welfare is expressed as the makespan of the obtained schedule [2–7,9,18,19]. However, in environments where jobs are owned by independent and competing agents for the same resource(s), it is more natural to measure the social welfare using another classical measure of performance, the *average* (weighted) completion time of the tasks [21]. A few papers consider this objective [1,11,12,15], but not in the context of truthfulness (they focus on coordination mechanisms and the price of anarchy). Given the interest of the algorithmic-game-theory community

© Springer International Publishing Switzerland 2016
T.N. Dinh and M.T. Thai (Eds.): COCOON 2016, LNCS 9797, pp. 15–26, 2016.
DOI: 10.1007/978-3-319-42634-1_2

to mechanism design aspects of scheduling problems, it is a natural question to know what is the difficulty of conceiving a truthful mechanism when the social welfare is the weighted completion time of the tasks. In some applications, for ethical or practical reasons, pricing is undesirable and so it is important to conceive mechanisms without payments [8,16]. In other applications however this is not the case. Hence we consider both cases in the sequel. We focus on the following problem: we are given a set of tasks where each task is owned by a selfish agent who is the only one to know the length and/or the weight of his task. The tasks have to be executed on a single-machine or on a set of identical machines. The valuation of each agent/task is the opposite of his completion time. The weight of a task models the importance of the task for the system (and not the agent) and in that case it is more natural to consider that the valuation of the agent is just the completion time of his task[1]. We study this problem both with payments and without payments. When we use payments, the objective of each agent is the maximization of his utility which is defined as the difference between his valuation and his payment. When payments are not allowed, the objective of each agent is the minimization of his (weighted) completion time. Agents may lie concerning their length and/or weight if by doing so, they are able to increase their utility. Our aim is to find a truthful mechanism that minimizes the weighted sum of completion times.

Our contribution. In the first part of the paper, we study the model *without payments*. When the lengths of the tasks are private data, we prove that there is no $(2-\epsilon)$-approximate deterministic truthful algorithm even in the case of a single machine where the weights of all the tasks are unitary. We also show that there is no $(\frac{3}{2}-\epsilon)$-approximate randomized truthful algorithm for the same environment. When both the lengths and the weights are private data, then we show that it is not possible to get an α-approximate deterministic truthful algorithm for any $\alpha > 1$. In order to overcome these negative results we introduce a new concept that we call *preventive preemption*. The intuitive idea behind preventive preemption is simple: whenever a task bids a length smaller than its real length, the scheduler will preempt it at the end of the declared processing time and he will resume it later. Think for instance a planning of a meeting room. Once the schedule of meetings is done, then every meeting has to finish or be interrupted at the planned time. An interrupted meeting could continue only after all other meetings are finished. Notice that as our mechanism is proved to be truthful no task will be interrupted during the constructed schedule. This is in the same vein as the approach used recently by Fotakis et al. [13] where *selective verification* is used as a threat in order to construct a truthful mechanism. Using preventive preemption as a threat, we are able to propose a simple optimal truthful algorithm with no payments for the single-machine problem where the lengths of the tasks are private and the weights are public. For multiple machines, we are able to prove that this approach gives an optimal truthful algorithm for

[1] Notice however that our results can be generalized to the case where the valuation of the tasks is their weighted completion time.

the unweighted case. For the case of multiple machines with weights, given that the problem is NP-hard even if all data are public, we turn our attention to the development of approximate truthful mechanisms. We propose a truthful randomized algorithm which is $\frac{3}{2}$-approximate in expectation based on preventive preemption. We also show that the natural WSPT algorithm of Smith [21] is not truthful. In the second part of the paper, we consider the model *with payments*. For the single-machine case, given that the optimal solution can be computed in polynomial time and the social welfare is utilitarian, one may think that it is sufficient to apply the well known Vickrey-Clarke-Groves (VCG) mechanism [10, 14, 22]. However, in what follows we prove that this is not true even when only the lengths of the tasks are private data. Then, we propose an optimal truthful mechanism for the single-machine case using preventive preemption and appropriately chosen payments. Our results are summarized in Table 1.

Table 1. Summary of the results presented in this paper. *TA* means "truthful algorithm", *det* means "deterministic" and *rand* means "randomized". The number before TA is the approximation ratio. For example, the sentence "$\nexists$ det $(2-\varepsilon)$ TA (thm 1)" in the first cell means that Theorem 1 shows that there does not exist any deterministic truthful algorithm which has an approximation ratio of $2-\varepsilon$ (when payment and preemption are not allowed, and when the lengths of the tasks are private). Unless otherwise specified, the results hold for any number of machines.

	Without preemption	With preventive preemption
Without payment	*Private lengths:* • $\nexists$ det $(2-\varepsilon)$ TA (thm 1) • $\nexists$ rand $(1.5-\varepsilon)$ TA (thm 2)	*Private lengths:* • $m=1$: $\exists$ optimal det TA (thm 4) • $m \geq 2$, identical w: $\exists$ optimal det TA (thm 5) • $m \geq 2$: $\exists$ rand 1.5 TA (thm 6)
	Private lengths and weights: • $\nexists$ det α TA, for all α (thm 3)	*Private lengths and weights:* • $\nexists$ det $(2-\varepsilon)$ TA (thm 8)
With payment	*Private lengths:* • $\nexists$ optimal TA (thm 7)	*Private lengths and weights:* • $m=1$: $\exists$ optimal det TA (thm 9) • $m \geq 2$: $\exists$ rand 1.5 TA (cor 2)

1.1 Formal Definition of the Problem

We consider n agents, $N = \{1, 2, \cdots, n\}$, and a single machine or a set of m parallel identical machines. Each agent i is the owner of a single task and he is the only one to know the private data of his task. The private data of a task can be either its length $t_i > 0$ or both its length $t_i > 0$ and its weight $w_i > 0$. When both the length and the weight of a task are private, we call these data (t_i, w_i), the *agent's true data* or the *agent's type* (if only the length of the task is private, then the agent's type is just t_i). Everything else is public knowledge. From now on in this section, we assume for simplicity that both the length and the weight of the tasks are private data. Each agent will report a pair (b_i, w_i^b) to the mechanism that we call the *agent's bid*. By B, we denote the set of all bids, i.e. $B = \{(b_1, w_1^b), \ldots, (b_n, w_n^b)\}$. We adopt an extension of the *strong model of* *execution* [4] where, once task i starts to be executed, it is executed during t_i units of time, independently of the value of his bid b_i (i.e. even if $b_i \neq t_i$). In the model of [4], the bid value b_i should always be larger than or equal to t_i while here, b_i may get any positive value ($b_i < t_i$ or $b_i \geq t_i$). By C_i, we denote the completion time of task i.

For the model *with payments*, a mechanism is a pair $\mathcal{M} = (A, P)$, where A is an algorithm that finds an output $o(B)$ and P is a payment function: $P(o(B), B) = (p_1, p_2, \ldots, p_n)$. The output $o(B)$ computed by A is a function of the bids, B, of the agents, while the payment is a function of the output $o(B)$ and of the agents' bids B. This means that, contrary to the framework with *verification* introduced by Nisan and Ronen for scheduling problems [19], the payments have to be computed without knowing the true types of the tasks. Let us now define the output of A. Since the true types of the tasks are not known by the mechanism, A is not able to produce a feasible schedule in which the completion time of every task is known in advance. In the case where the preemption of the tasks is not allowed, $o(B)$ is defined as the order in which the tasks will be executed on each machine along with the lengths of the idle-periods that precede the tasks, if such idle periods exist. More formally, in the single-machine case when the preemption of the tasks (the possibility of interrupting and resuming the execution of the task later) is not allowed, we define the output $o(B)$ of algorithm A as a sequence of n pairs (I_i, i) where i is a task and I_i is the length of the idle-period just before task i. Notice that when no idle-periods exist between the tasks, all I_i's will be equal to 0 and we will simply denote the output by a sequence of n tasks. In the case where the preemption of the tasks is allowed, the output $o(B)$ will be defined in a similar way, the only difference being that more than one time-intervals may represent a task, one time-interval for each piece of the preempted task. For multiple machines, the above definitions generalize in the natural way. The objective of the mechanism is to determine a schedule of the tasks minimizing the sum of weighted completion times, or equivalently maximizing the social welfare which is defined as $-\sum_{1 \leq i \leq n} w_i C_i$. For every task i, we define S_i as the set of tasks scheduled before i on the same machine in the output $o(B)$, and T_i as the set of real lengths of the tasks of S_i (i.e. $T_i = \{t_j : j \in S_i\}$). The completion time of task i is $C_i = \sum_{j \in S_i} (I_j + t_j) + I_i + t_i$ and the utility of task i is $u_i(t_i, o(B), B, T_i) = -C_i(t_i, o(B), B, T_i) - p_i(o(B), B)$, where $p_i(o(B), B)$ is the payment, or in other words the amount that i must pay. It is important here to notice that the payments are computed before the real execution of the tasks.

For the model *without payments*, a mechanism for this problem is an algorithm A that determines an output $o(B)$.

In both models, every task/agent i is considered as selfish: the strategy of agent i is to declare a bid (b_i, w_i^b) in order to maximize his utility u_i. Our aim is to propose a *truthful mechanism*, i.e. a mechanism that gives incentive to the agents/tasks to declare their true types. We say that a mechanism is *truthful* if and only if for every i, $1 \leq i \leq n$, and for every bid (b_j, w_j^b), $j \neq i$, the utility u_i of task i reaches its maximum when i bids its true data, i.e. $(b_i, w_i^b) = (t_i, w_i)$.

In other words, a mechanism is truthful if truth-telling is the best strategy for a player i regardless of the strategies adopted by the other players.

2 No Payments

In this section, we consider the problem of designing a truthful mechanism without payments. We start by proving some negative results for truthful deterministic or randomized algorithms. Then, we introduce the notion of preventive preemption, and we show that by using it we are able to design optimal or approximate truthful mechanisms.

2.1 Negative Results: Private Lengths

We first consider deterministic algorithms.

Theorem 1. *Let $\varepsilon > 0$. There is no truthful deterministic $(2-\varepsilon)$-approximate algorithm, even if all the tasks have the same weights.*

Proof. Let $\mathcal{A}$ be a deterministic algorithm which is α-approximate, with $\alpha < 2$. Let us show that $\mathcal{A}$ is not a truthful algorithm.

Let us consider a first instance I_1: a single machine and two tasks T_1 and T_2 of lengths M and M^2 respectively (with $M > 1$). Both tasks have the same weight (in the sequel we will thus consider the criteria $\sum C_i$, which is equivalent to $\sum w_i C_i$ in this case). In an optimal schedule, T_1 is executed at time 0 and T_2 starts when T_1 has been executed, at time M. The cost of such a schedule is $\sum_{i\in\{1,2\}} C_i = M + (M + M^2) = M^2 + 2M$. In such a schedule task T_2 starts at time M.

Let S be a schedule of I_1 in which task T_2 starts *before time M*. In such a schedule task T_1 cannot be completed before the start of T_2. The cost of S is thus larger than or equal to $M^2 + (M^2 + M) = 2M^2 + M$ (in the best case there is no idle time: task T_2 is scheduled at time 0 and task T_1 starts as soon as T_2 is completed, i.e. at time M^2). The ratio between the cost of S and the optimal cost is larger than or equal to $\frac{2M^2+M}{M^2+2M} = \frac{2M+1}{M+2}$, which tends towards 2 when M tends towards the infinity. Since $\mathcal{A}$ is an α-approximate algorithm, with $\alpha < 2$, $\mathcal{A}$ cannot return schedule S. Therefore, in the schedule returned by $\mathcal{A}$ on instance I_1, T_2 starts at the soonest at time M.

Consider now a second instance, I_2: a single machine and two tasks T_1 and T_3 of lengths M and 1 respectively. Both tasks have the same weight. In an optimal schedule T_3 is executed at time 0 and T_1 starts when T_3 has been executed, at time 1. The cost of such a schedule is $1 + (1 + M) = M + 2$.

Let S be a schedule of I_2 in which task T_3 *does not start before time M*. The cost of S is thus larger than or equal to $M+(M+1) = 2M+1$ (in the best case task T_1 is scheduled at time 0 and task T_3 starts as soon as T_1 is completed, i.e. at time M). The ratio between the cost of S and the optimal cost is larger than or equal to $\frac{2M+1}{M+2}$, which tends towards 2 when M tends towards the infinity. Since $\mathcal{A}$ is an α-approximate algorithm, with $\alpha < 2$, $\mathcal{A}$ cannot return schedule S. Therefore, in the schedule returned by $\mathcal{A}$ on instance I_2, T_3 starts before time M.

Let us now consider the following situation: task T_1 bids a length M and task T_2 has a true length of M^2. Given the values bid by T_1, if T_2 bid its true value, then the instance corresponds to instance I_1. As seen above, in the schedule returned by $\mathcal{A}$ on instance I_1, T_2 starts *at the soonest at time* M.

Assume that task T_2 lies and bids a length of 1 instead of M^2. The input of the algorithm is now two tasks of length M and 1: it is instance I_2 (the algorithm cannot know that T_2 lies). As seen above, since $\mathcal{A}$ is an α-approximate algorithm, with $\alpha < 2$, in the schedule returned by $\mathcal{A}$ on instance I_2, T_2 starts *before time* M. Task T_2 decreases its starting time (and thus its completion time) by bidding a false value. Therefore $\mathcal{A}$ is not a truthful algorithm.

If we consider the case of randomized algorithms, we are able to prove the following result (the proof is omitted).

Theorem 2. *Let $\mathcal{A}$ be a (randomized) truthful algorithm which does not introduce idle times between the tasks. Then $\mathcal{A}$ is not α-approximate, with $\alpha < \frac{3}{2}$.*

2.2 Negative Results: Private Lengths and Weights

If both the lengths and the weights of the tasks are private data then it is not possible to obtain a truthful deterministic approximation algorithm.

Theorem 3. *Let $\alpha > 1$. There is no truthful deterministic α-approximate algorithm if both the lengths and the weights of the tasks are private values.*

Proof. Let $\mathcal{A}$ be a deterministic algorithm which is α-approximate. Let us show that $\mathcal{A}$ is not a truthful algorithm. Let $M = 3\alpha$.

Let us consider a first instance I_1: a single machine and two tasks T_1 and T_2. Task T_1 has a length of M^2 and a weight of 1. Task T_2 has a length of M and a weight of M. In an optimal schedule, T_2 is executed at time 0 and T_1 starts when T_2 has been executed, at time M. The cost of such a schedule is $M^2 + (M + M^2) = 2M^2 + M$.

Let S be a schedule of I_1 in which task T_1 starts *before time* M. In such a schedule, task T_2 cannot be completed before the start of T_1: since no preemption is allowed, T_1 is executed before T_2. The cost of S is thus larger than or equal to $M^2 + (M^2 + M)M = M^3 + 2M^2$ (in the best case there is no idle time: task T_1 is scheduled at time 0 and task T_2 starts as soon as T_1 is completed, i.e. at time M^2). The ratio between the cost of S and the optimal cost is thus larger than or equal to $\frac{M^3+2M^2}{2M^2+M} = \frac{M^2+2M}{2M+1} > \frac{M}{3} = \alpha$. Since $\mathcal{A}$ is an α-approximate algorithm, $\mathcal{A}$ cannot return schedule S. Therefore, in the schedule returned by $\mathcal{A}$ on instance I_1, T_1 starts at the soonest at time M.

Let us now consider a second instance, I_2: a single machine and two tasks T_1 and T_2. Task T_1 has a length of 1 and a weight of M^2. Task T_2 has a length of M and a weight of M. In an optimal schedule T_1 is executed at time 0 and

T_2 starts when T_1 has been executed, at time 1. The cost of such a schedule is $M^2 + (1 + M)M = 2M^2 + M$.

Let S be a schedule of I_2 in which task T_1 *does not start before time* M. The cost of S is thus larger than or equal to $M^2 + (M + 1)M^2 = M^3 + 2M^2$ (in the best case task T_2 is scheduled at time 0 and task T_1 starts as soon as T_2 is completed, i.e. at time M). The ratio beween the cost of S and the optimal cost is larger than or equal to $\frac{M^3+2M^2}{2M^2+M} = \frac{M^2+2M}{2M+1} > \frac{M}{3} = \alpha$. Since $\mathcal{A}$ is an α-approximate algorithm, $\mathcal{A}$ cannot return schedule S. Therefore, in the schedule returned by $\mathcal{A}$ on instance I_2, T_1 starts before time M.

Let us now consider the following situation: task T_1 has a length M^2 and weight 1 and task T_2 bids a length M and a weight M. Given the values bid by T_2, if T_1 bids its true values, then the instance corresponds to instance I_1. As seen above, in the schedule returned by $\mathcal{A}$ on instance I_1, T_1 starts *at the soonest at time* M.

Let us now consider that task T_1 lies and bids a length of 1 and a weight of M^2. The input of the algorithm is now identical to instance I_2 (the algorithm cannot know that T_1 lies). As seen above, since $\mathcal{A}$ is an α-approximate algorithm, in the schedule returned by $\mathcal{A}$ on instance I_2, T_1 starts *before time* M. Task T_1 decreases its starting time (and thus its completion time) by bidding false values. Therefore $\mathcal{A}$ is not a truthful algorithm.

2.3 Positive Results: Single Machine with Preventive Preemption

In the remaining of this section, we show that if preventive preemption is used, then it becomes possible to design a truthful mechanism without payments which is optimal with respect to the social welfare. A preemptive schedule on a single machine can be defined as a vector $\sigma = (\rho_1, \ldots, \rho_n)$ where for every task i, $1 \le i \le n$, ρ_i corresponds to the set of time-intervals during which task i is executed, i.e. $\rho_i = [l_i^1, r_i^1) \cup \cdots \cup [l_i^k, r_i^k)$ with $l_i^1 < r_i^1 \le l_i^2 < r_i^2 \le \cdots \le l_i^k < r_i^k$ and $\sum_{j=1}^{k} \left(r_i^j - l_i^j \right) = t_i$, where t_i is the true length of task i. In addition, for every pair of tasks i, j, we have $\rho_i \cap \rho_j = \emptyset$. Hence, in schedule σ, task i starts at time l_i^1, it is preempted at time r_i^1, then its execution continues at time l_i^2, it is again preempted at time r_i^2 and so on until its completion. Clearly, for the considered objective function, i.e. the sum of weighted completion times, any schedule where at least one task is preempted is strictly worse than the optimal non-preemptive schedule. Hence, given that we are interested in obtaining a truthful algorithm which outputs an optimal outcome, we need to design an algorithm which preempts the execution of a task only when the task bids a false value of its length. However, there is no possibility for the mechanism to know *a priori* if a task lies, and the mechanism has to define a (perhaps preliminary) schedule based only on the values that the tasks bid, i.e. *before* their real execution. Our algorithm is the following one: it schedules the tasks following the increasing order of the ratio of the declared length to weight, i.e. following Smith's rule, and it executes each task i during b_i units of time in the time interval $[l_i^1, l_i^1 + b_i)$. Whenever the real length of a task is greater than its declared one, then the task will be preempted at $l_i^1 + b_i$

and restarted after the completion of all the b_i's, $1 \leq i \leq n$, following a round robin policy if more than one tasks are preempted. We now introduce what we will call *preventive preemption.*

Definition 1. *An algorithm uses* preventive preemption *if it constructs a schedule in which a task i is preempted (and resumed later), if and only if, $b_i < t_i$.*

Our algorithm, that we call Weighted Shortest Processing Time with Preventive Preemption (WSPT-PP), uses the concept of preventive preemption. Our algorithm is based on the classical Smith's rule WSPT (Weighted Shortest Processing Time) which is optimal for the sum of the weighted processing times for the single-machine case. As we prove below an important property of WSPT-PP is that it is *truthful* and consequently no task is finally preempted, since for every task i, we have $b_i = t_i$. Let us now define more formally this algorithm[2].

```
Algorithm WSPT-PP
1 Sort all tasks in the WSPT order (i.e. such that $\frac{b_1}{w_1^b} \leq \frac{b_2}{w_2^b} \leq \cdots \leq \frac{b_n}{w_n^b}$).
2 Schedule the first interval $[l_i^1, r_i^1)$ of every task $i$ such that $l_i^1 = \sum_{j=1}^{i-1} b_j$
  and $r_i^1 = l_i^1 + b_i$.
3. After time $t = \sum_{j=1}^{n} b_j$, schedule the tasks which are not already com-
  -pleted using the round robin policy:  For each $x \geq 2$, if Task $i$ is not
  completed at time $\left(\sum_{j=1}^{n} b_j\right) + n(x-2) + i - 1$, schedule this task in
  the time interval $[l_i^x, r_i^x)$, with $l_i^x = \left(\sum_{j=1}^{n} b_j\right) + n(x-2) + i - 1$
  and $r_i^x = \left(\sum_{j=1}^{n} b_j\right) + n(x-2) + i$.
```

Theorem 4. WSPT-PP *is a polynomial-time, optimal and truthful algorithm for the single machine case where the private data of every task is its length and the social welfare is the weighted sum of completion times.*

Proof. Assume that task i bids $b_i > t_i$. By the definition of WSPT-PP, task i will not start earlier than if it bids $b_i = t_i$ (and thus it will not decrease its completion time by lying). On the other hand, if task i bids $b_i < t_i$, again by the definition of WSPT-PP, it will be preempted b_i units of time after its starting time and it will be continued after date $\sum_{j=1}^{n} b_j$. Thus, its completion time will be at least $t_i - b_i + \sum_{j=1}^{n} b_j = t_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} b_j$. If it bids $b_i = t_i$, it will not be preempted and its completion time will be at most $\sum_{j=1}^{n} b_j = t_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} b_j$. In both cases task i has no incentive to lie, and so WSPT-PP is truthful. Thus the obtained schedule is without preemption, i.e. identical to the one obtained by the classical WSPT algorithm. Given the optimality of WSPT, we obtain that WSPT-PP is also optimal. □

Remark. Notice that the previous results hold also if the valuation of each task is defined as its weighted completion time.

[2] Recall that in this section $w_i^b = w_i$.

2.4 Positive Results: Parallel Machines with Preventive Preemption

It is well known that the Shortest Processing Time (SPT) algorithm computes an optimal solution for the problem of minimizing the sum of completion times on identical parallel machines [21]. Based on that, we can apply SPT with preventive preemption (SPT-PP) on identical parallel machines and obtain a polynomial-time optimal and truthful algorithm for the parallel machines case where the social welfare is the minimization of the sum of completion times.

The proof of the truthfulness of SPT-PP is similar than the one of WSPT-PP for the single-machine case and it is omitted here. Given the truthfulness of SPT-PP, it is easy to see that no task will be preempted by SPT-PP and the produced schedule will be the same as the one of SPT.

Theorem 5. SPT-PP *is an optimal and truthful algorithm for the parallel machine case where the private data of every task is its length and the social welfare is the sum of completion times.*

For the multiple machines case with weights, given that the problem is NP-hard even if all data are public, we turn our attention to the development of approximate truthful mechanisms. We propose the following simple algorithm that we call RAND-WSPT-PP: Assign tasks independently and uniformly at random to the machines, and on each machine schedule the tasks using the WSPT rule by applying preventive preemption if necessary. It is easy to see that a task i has no influence on the choice of the machine on which it will be scheduled by lying on its length. In addition, according to the proof of Theorem 4 whatever the machine it is scheduled on, its best strategy is to declare $b_i = t_i$. This means that all the tasks will declare their true lengths and the algorithm will produce a non-preemptive schedule. It has been proved in [20] that this algorithm is 3/2-approximate in expectation. Consequently, we get the following result.

Theorem 6. RAND-WSPT-PP *is a truthful randomized 3/2-approximate in expectation algorithm for the parallel machine case where the private data of every task is its length and the social welfare is the weighted sum of completion times.*

Remark. The derandomization of this algorithm is WSPT-PP: the tasks are sorted according to the non decreasing ratio of b_i/w_i's, and they are scheduled following this order as soon as a machine becomes available [21]. If we impose large penalties on liars, e.g. by starting the exceeding part of a task at a time equal to the sum of all the declared processing times of the tasks, then it is easy to see that preventive preemption guarantees that no agent will lie when we apply WSPT-PP. This gives a $(1+\sqrt{2})/2$-approximation [17]. If however, we impose that the exceeding part is started after the completion of the last task on the same machine or on any machine, then the tasks have incentive to lie. To see this consider the following example.

Example. Consider the following instance: two machines and three tasks: $w_1 = t_1 = 1$, $w_2 = t_2 = 1$, $w_3 = 2$ and $t_3 = 2 + \varepsilon$ (where ϵ is a small positive value, e.g. $\varepsilon = 0.1$). The schedule returned by WSPT-PP is the following one: each task of length 1 is scheduled at time 0 on a machine. Task 3 is scheduled at time 1, after a task of length 0. Its completion time is thus $3+\varepsilon$. Task 3 has incentive to bid $2-\varepsilon$. In this case, WSPT-PP schedules task 3 at time 0, and tasks 1 and 2 are scheduled on the other machine. Since task 3 is alone on its machine, it will be completed at time $2+\epsilon$ even with preventive preemption. Even if we consider a stronger version of preventive preemption, that we may call preventive preemption with migration, where we execute the remaining part of the preempted task on the machine of maximum load, then task 3 will finish at time $2+2\epsilon$ instead of $3-\varepsilon$: task 3 has still incentive to bid a false value.

3 Introducing Payments

3.1 Private Lengths

Let us first prove that the VCG method cannot be applied for the single-machine case without preventive preemption.

Theorem 7. *There is no optimal truthful mechanism with payment for the single machine case even in the unweighted case.*

Proof. By contradiction, assume that there is an optimal truthful mechanism minimizing the sum of completion times of the tasks on a single machine. It is well known that the Shortest Processing Time first (SPT) algorithm, which schedules the tasks in non-decreasing order of their lengths, is the only algorithm that maximizes the social welfare $-\sum_{1\leq i\leq n} C_i$. Given that SPT does not insert any idle time, a schedule can be defined as an ordering of the tasks. Let 1 and 2 be the two tasks to schedule (i.e. $N=\{1,2\}$) and consider the following scenario: when task 2 tells the truth, we have $t_2=b_2>b_1$. In this case, SPT constructs a schedule σ where task 1 is scheduled before task 2 ($\sigma=(1,2)$). Then the utility of task 2 is $u_2=-C_2-p_2=-t_1-t_2-p_2$. On the other hand, when task 2 lies and bids $b'_2<b_1$, SPT constructs σ' where task 2 is scheduled before task 1 ($\sigma'=(2,1)$) and the utility of task 2 becomes $u'_2=-C'_2-p'_2=-t_2-p'_2$. Given that the mechanism is assumed to be truthful, we must have $u_2\geq u'_2$ (i.e. task 2 should not have incentive to lie) and thus $-t_1-t_2-p_2\geq -t_2-p'_2 \Rightarrow p'_2-p_2\geq t_1$. However, since t_1 is not known to the mechanism when the payments are computed, it is clear that there is no any payment function satisfying this property. □

Corollary 1. *The VCG method cannot be applied for the single-machine case.*

3.2 Private Lengths and Weights

In this section, we show that preventive preemption associated with payments helps even when both the length and the weight of the tasks are private data. Since now each agent can lie on his weight, algorithm WSPT-PP is not truthful anymore. Indeed any task i has incentive to bid $b_i=t_i$ and $w_i^b>w_i$ in order to get a smaller ratio $\frac{b_i}{w_i^b}$, and then to decrease its completion time C_i. Moreover, as shown by Theorem 8 below, when both weights and lengths are private values, there is no optimal algorithm even if preemptive preemption is allowed (the proof is omitted due to lack of space). We then propose an optimal truthful algorithm which uses payment and preventive preemption.

Theorem 8. *Let $\varepsilon > 0$. There is no truthful deterministic $(2-\varepsilon)$-approximate algorithm which does not use payment when the weights of the tasks is a private value, even when preventive preemption is allowed.*

Theorem 9. *For every task i, let s_i be the starting time of task i in the schedule obtained by* WSPT-PP*. The mechanism using algorithm* WSPT-PP *and the following payment function $p_i = -s_i + \sum_{j \neq i} b_j$ is polynomial-time computable, optimal and truthful for the single machine case.*

Proof. By the definition of algorithm WSPT-PP, $-s_i + \sum_{j \neq i} b_j$ is a positive value and it can be computed by the scheduler using only the values $(b_1, w_1^b), \ldots, (b_n, w_n^b)$. Thus, $p_i = -s_i + \sum_{j \neq i} b_j$ is a valid payment function. Moreover, for every task i, if i tells the truth, we have $u_i = -C_i - p_i = -(s_i + t_i) - (-s_i + \sum_{j \neq i} b_j) = -t_i - \sum_{j \neq i} b_j$ whereas if i lies, by the definition of algorithm WSPT-PP, it cannot be completed before time $s_i + t_i$ and thus we have $u_i \leq -t_i - \sum_{j \neq i} b_j$. Hence, task i takes no advantage of not telling the truth and so the mechanism is truthful. Moreover, given the truthfulness of the mechanism, WSPT-PP constructs the same schedule as WSPT without preemption. Thus, as WSPT constructs an optimal solution minimizing the sum of the weighted completion times, so does WSPT-PP. □

For applications where the valuation of a task is its weighted completion time, it is also possible to obtain payments that ensure that WSPT-PP is truthful (the details will be given in the full version of the paper).

Multiple machines. Notice that for multiple machines we can use the algorithm RAND-WSPT-PP (see Sect. 2.4) with appropriate payments in order to obtain a randomized truthful approximation algorithm.

Corollary 2. *There exists a truthful $\frac{3}{2}$-approximate in expectation algorithm for the parallel machine case with payments when the private data of every task are its length and its weight.*

Acknowledgments. The work of Evripidis Bampis and Fanny Pascual was partly supported by the French ANR grant ANR-14-CE24-0007-01 “CoCoRICo-CoDec”.

References

1. Abed, F., Correa, J.R., Huang, C.-C.: Optimal coordination mechanisms for multi-job scheduling games. In: Schulz, A.S., Wagner, D. (eds.) ESA 2014. LNCS, vol. 8737, pp. 13–24. Springer, Heidelberg (2014)

2. Ambrosio, P., Auletta, V.: Deterministic monotone algorithms for scheduling on related machines. In: Persiano, G., Solis-Oba, R. (eds.) WAOA 2004. LNCS, vol. 3351, pp. 267–280. Springer, Heidelberg (2005)
3. Andelman, N., Azar, Y., Sorani, M.: Truthful approximation mechanisms for scheduling selfish related machines. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 69–82. Springer, Heidelberg (2005)
4. Angel, E., Bampis, E., Pascual, F.: Truthful algorithms for scheduling selfish tasks on parallel machines. Theoret. Comput. Sci. **369**, 157–168 (2006)
5. Angel, E., Bampis, E., Pascual, F., Tchetgnia, A.: On truthfulness and approximation for scheduling selfish tasks. J. Sched. **12**, 437–445 (2009)
6. Angel, E., Bampis, E., Thibault, N.: Randomized truthful algorithms for scheduling selfish tasks on parallel machines. Theor. Comput. Sci. **414**(1), 1–8 (2012)
7. Archer, A., Tardos, E.: Truthful mechanisms for one-parameter agents. In: FOCS, pp. 482–491 (2001)
8. Braverman, M., Chen, J., Kannan, S.: Optimal provision-after-wait in healthcare. In: ITCS 2014, Princeton, NJ, pp. 541–542 (2014)
9. Christodoulou, G., Gourvès, L., Pascual, F.: Scheduling selfish tasks: about the performance of truthful algorithms. In: Lin, G. (ed.) COCOON 2007. LNCS, vol. 4598, pp. 187–197. Springer, Heidelberg (2007)
10. Clarke, E.: Multipart pricing of public goods. Public Choice **11**(1), 17–33 (1971)
11. Cohen, J., Pascual, F.: Scheduling tasks from selfish multi-tasks agents. In: Träff, J.L., Hunold, S., Versaci, F. (eds.) Euro-Par 2015. LNCS, vol. 9233, pp. 183–195. Springer, Heidelberg (2015)
12. Cole, R., Correa, J.R., Gkatzelis, V., Mirrokni, V.S., Olver, N.: Inner product spaces for minsum coordination mechanisms. In: ACM STOC 2011, pp. 539–548 (2011)
13. Fotakis, D., Tzamos, C., Zampetakis, E.: Who to trust for truthfully maximizing welfare? CoRR abs/1507.02301 (2015)
14. Groves, T.: Incentive in teams. Econometrica **41**(4), 617–631 (1973)
15. Hoeksma, R., Uetz, M.: The price of anarchy for minsum related machine scheduling. In: Solis-Oba, R., Persiano, G. (eds.) WAOA 2011. LNCS, vol. 7164, pp. 261–273. Springer, Heidelberg (2012)
16. Hurst, J., Siciliani, L.: Tackling excessive waiting times for elective surgery: a comparison of policies in 12 OECD countries. Health Policy **72**(2), 201–215 (2005)
17. Kawaguchi, T., Kyan, S.: Worst case bound of an LRF schedule for the mean weighted flow-time problem. SIAM J. Comput. **15**(4), 1119–1129 (1986)
18. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, p. 404. Springer, Heidelberg (1999)
19. Nisan, N., Ronen, A.: Algorithmic mechanism design. In: STOC, pp. 129–140 (1999)
20. Schulz, A.S., Skutella, M.: Scheduling unrelated machines by randomized rounding. SIAM J. Discret. Math. **15**(4), 450–469 (2002)
21. Smith, W.E.: Various optimizers for single stage production. Naval Res. Logist. Q. **3**, 59–66 (1956)
22. Vickrey, W.: Counterspeculation, auctions and competitive sealed tenders. J. Financ. **16**, 8–37 (1961)