

# Mechanisms for Scheduling with Single-Bit Private Values

Vincenzo Auletta · George Christodoulou ·  
Paolo Penna

Published online: 31 March 2015  
© Springer Science+Business Media New York 2015

**Abstract** We study the problem of designing truthful mechanisms for makespan minimization in scheduling. In particular, we consider randomized mechanisms for a restriction of the general multi-dimensional domain (i.e., unrelated machines). In a sense, our setting is the simplest multi-dimensional setting, where each machine holds privately only a single-bit of information. Some of the impossibility results for deterministic mechanisms carry over our setting as well. We prove a separation between truthful-in-expectation and universally truthful mechanisms for makespan minimization: We first show how to design an optimal truthful-in-expectation mechanism, and then prove lower bounds on the approximation guarantee of universally truthful mechanisms.

**Keywords** Mechanism Design · Scheduling · Approximation algorithms · Analysis of algorithms

## 1 Introduction

Mechanism design is a field in game theory that studies how to implement good system-wide solutions to problems that involve multiple self-interested agents, each one holding (part of) the inputs and having her own private preferences about

---

V. Auletta (✉) · P. Penna  
Dipartimento di Informatica, Università di Salerno, Salerno, Italy  
e-mail: auletta@unisa.it

P. Penna  
e-mail: penna@dia.unisa.it

G. Christodoulou  
Computer Science Department, University of Liverpool, Liverpool, UK  
e-mail: gchristo@liv.ac.uk

the solutions computed by the mechanism. Algorithmic mechanism design mainly focused on settings where the goal is to maximize the social welfare in auction contexts where the goal is to maximize the auctioneer revenue. Recently, other optimization goals have been considered, such as those that incorporate fairness criteria. Among these alternative goals the problem of minimizing the makespan in the context of machine scheduling is certainly one of the most extensively studied.

Designing truthful mechanisms for scheduling problems was first suggested in the seminal paper by Nisan and Ronen [17], as a paradigm to demonstrate the applicability of Mechanism Design to an optimization problem. In its general form, where the machines are *unrelated*, there are  $n$  jobs to be assigned to  $m$  machines. The time needed by a machine  $i$  to process job  $k$  is described by a nonnegative real value  $t_{ik}$ . Given such an input matrix, a standard task from the algorithm designer's point of view, is to allocate the jobs in a way such that some global objective is optimized; a typical objective is to minimize the maximum completion time (i.e. the makespan).

In a game-theoretic setting, it is assumed that each entry of this matrix is not known to the designer, but instead it is a *private* value held by a selfish agent that controls the machine. We call this private value the agent's *type*. The mechanism has to ask each agent for her type and the agent can misreport her type to the designer if this is advantageous to her. Mechanism design suggests using monetary compensation to incentivize agents to report truthfully. Truthful implementation in *dominant strategies* is desired, because it facilitates the prediction of the outcome and at the same time simplifies the agents' way of reasoning. The challenge is to design truthful mechanisms that optimize/approximate the makespan.

When the entries of the matrix  $t$  are unrelated, the type domain for each machine  $i$  is an  $n$ -valued vector  $t_i$ . For this multi-dimensional domain, the constraints imposed by truthfulness make the problem hard. Nisan and Ronen [17], showed that it is impossible to design a truthful mechanism with approximation factor better than 2, even for two machines. Later this bound was further improved to 2.41 [6] for 3 machines, and to 2.618 [11] for many machines. In [17], it was also shown that applying the VCG mechanism [7, 10, 20] achieves an approximation ratio of  $m$ , and it has been conjectured that this bound is tight. This conjecture still remains open, but it was further strengthened by Ashlagi et al. [1], who proved the conjecture for the intuitively very natural case of anonymous mechanisms (where roughly the allocation algorithm does not base its decisions on the machines' ids).

Randomization provably helps for this problem. There are two notions of truthfulness for randomized mechanisms. Roughly, a mechanism is *universally truthful* if it is defined as a probability distribution over deterministic truthful mechanisms, while it is *truthful-in-expectation*, if in expectation no player can benefit by lying. Already in [17], a universally truthful mechanism was suggested for the case with two machines. The mechanism was extended to the case of  $m$  machines by Mu'alem and Schapira [16] with an approximation guarantee of  $0.875m$ , and this was further improved in [14] to  $0.837m$ . Lu and Yu [15] showed a truthful-in-expectation mechanism with an approximation guarantee of  $(m+5)/2$ . In [16] a lower bound of  $2 - 1/m$  for both randomized versions was shown, while in [5] the lower bound was extended to fractional mechanisms, and an upper bound of  $(m+1)/2$  was provided.

Surprisingly, even for the special case of two machines a tight answer on the approximation factor of truthful randomized mechanisms has not been given. Currently, the lower bound is 1.5 [16], while the best upper bound is 1.5963 due to [15].

Setting restrictions to the input domain can make the problem easier. The single-dimensional counterpart of the problem is scheduling on *related* machines. In this case it is assumed that machine  $i$  has speed  $s_i$  and  $t_{ik} = w_k/s_i$ , where the weights  $w_k$  of the jobs are known to the designer. Notice that the only information missing to the designer is the speed of the machines. Here, the constraints imposed by truthfulness seem harmless; the optimal allocation is truthfully implementable [2], although it takes exponential running time, while the best possible approximation guarantee, a PTAS, can be achieved by polynomial time truthful mechanisms [4, 8]. In fact, a recent paper [9] shows that monotone PTASs can be achieved for a wide range of objective functions. An immediate conclusion is that when one restricts the domain, then truthfulness becomes less and less stringent.

A prominent approach suggested by Lavi and Swamy [12], is to restrict the input domain, but still keep the multi-dimensional flavour. They assumed that each entry in the input matrix can take only two possible values  $L, H$ , that are publicly known to the designer. In this case, a very elegant deterministic mechanism achieves an approximation factor of 2, that is a great improvement comparing to the  $m$  upper bound that is the best known for the general problem. Surprisingly, even for this special case the lower bound is  $11/10$ .

### 1.1 Our Contribution

The focus of this work is to study selfish scheduling problems on restricted but multi-dimensional input domains for which optimized/efficient mechanisms can be given. Following [12], we consider domains where each agent's type can take only one of two possible values  $L, H$ , with  $L < H$ , that are publicly known, but we even restrict the way these values are placed in a player input vector. We assume that the designer is given for each machine some publicly known partition of the tasks into two sets such that all jobs in the same set take the same execution time. Thus, the only information missing to the designer is which set of the partition contains jobs taking time  $L$  and which set contains jobs taking time  $H$ . Therefore, the only missing information is a *single bit* for each player.<sup>1</sup> The lower bound given in [12] is still valid for our setting. It is important to emphasize that all the aforementioned lower bounds are due to truthfulness, and hold even for exponential running time algorithms. We explore the effects of truthfulness (both randomized and deterministic) in this restricted setting. Our contributions are the following:

---

<sup>1</sup>Notice that the information missing is just a single bit, much less than that of the related machines case, where the missing information is a positive real number. However, ours is not a single-dimensional domain. We refer the reader to Chapter 9 and 12 of [18] for the precise definition of a single-dimensional domain.

(1) *Power of truthful-in-expectation mechanisms* There is a class of scheduling problems on two-values domains with publicly known partitions (see Section 1.2) for which every algorithm (thus including optimal ones) can be turned into a truthful-in-expectation mechanism with the same approximation guarantee (Theorem 4). On the contrary, universally truthful randomized mechanisms cannot achieve an approximation better than  $31/30$  (Theorem 22), and the  $11/10$  lower bound for deterministic mechanisms in [12] also applies.<sup>2</sup>

(2) *Mechanisms for two machines* For the special case of two machines with uniform partitions (see Section 1.2), we present an exact truthful-in-expectation mechanism (Theorem 19).

We also give some evidence that two-values domains are easier than the general case, even for this simple case of two machines. In fact, we show that the lower bound of 2 for deterministic mechanisms for unrelated machines still hold (Theorem 24) when we admit three-values domains. On the other hand, a  $3/2$ -approximation deterministic truthful mechanisms exists for two-values domains with publicly known partitions (Theorem 20), that are still a restriction of the two-values domains.

## 1.2 Preliminaries

*The scheduling domain* We have  $n$  jobs to be scheduled on  $m$  machines. Each job must be assigned to exactly one machine. In the unrelated-machines setting, each machine  $i$  has a vector of processing times or *type*  $t_i = (t_{ih})_h$ , where  $t_{ih} \in \mathfrak{R}_{\geq 0}$  is  $i$ 's processing time for job  $h$ .

In the two-values domains by Lavi and Swamy [12], the time for executing job  $h$  on machine  $i$  is either  $L$  (low) or  $H$  (high), with  $H > L$  (the case  $L = H$  is trivial). Given a partition of the jobs  $(S, \bar{S})$ , we say that machine  $i$  is an  $L_S$ -machine (respectively,  $H_S$ -machine) if all jobs in  $S$  take time  $L$  (respectively,  $H$ ), and all jobs not in  $S$  take time  $H$  (respectively,  $L$ ). That is, the type  $t_i$  of an  $L_S$ -machine  $i$  is such that for any job  $h$

$$t_{ih} = L_S^h := \begin{cases} L & \text{if } h \in S \\ H & \text{otherwise} \end{cases}$$

and similarly for  $H_S$ -machines.

In this work, we consider the two-values domains with publicly known partitions, that are special cases of the two-values domains defined in [12]. For each machine  $i$  we are given a publicly known subset  $S_i$  and the private information is whether  $i$  is an  $L_{S_i}$ -machine or an  $H_{S_i}$ -machine. Hence, the type  $t_i \in \{L_{S_i}, H_{S_i}\}$ . Intuitively, a type  $t_i = L_{S_i}$  indicates that machine  $i$  is “good” for the jobs in  $S_i$  and “bad” for other jobs, while for  $t_i = H_{S_i}$  it is the other way around (notice that  $H_{S_i} = L_{\bar{S}_i}$  where

<sup>2</sup>Notice that such a separation was not known for the general problem since, although Lu [13], showed a lower bound higher than 1.5 for universally truthful mechanisms, the result holds only for scale-free mechanisms. This is arguably a very natural assumption, but it is still needed to be proven that it is without loss of generality.

$\bar{S}_i = [n] \setminus S_i$ ). We shall further distinguish between three restrictions (of increasing difficulty) of the domain with publicly known partitions:

1. **identical partitions**, where all subsets  $S_i$  are identical;
2. **uniform partitions**, where all subsets  $S_i$  have size  $s$  for some  $s \geq 0$ ;
3. **(unrestricted) publicly known partitions**, which impose no restriction on the subsets  $S_i$ .

We say that job  $h$  is an  $L$ -job (respectively,  $H$ -job) for machine  $i$  if  $t_{ih} = L$  (respectively,  $t_{ih} = H$ ), with  $t_i$  being the type of machine  $i$ . We represent an allocation by a matrix  $x = (x_{ih})$ , where  $x_{ih} \in \{0, 1\}$  and  $x_{ih} = 1$  iff job  $h$  is assigned to machine  $i$  (since every job is assigned to exactly one machine,  $\sum_i x_{ih} = 1$ ).

Given an allocation  $x$  and machine types  $t$ , we define the *load* of machine  $i$  as the set of jobs allocated to  $i$  in  $x$  and denote by

$$C_i(x, t) := \sum_h x_{ih} t_{ih}$$

the *cost* of the allocation  $x$  for machine  $i$ . The *makespan* of  $x$  with respect to  $t$  is the maximum cost of any machine, i.e.,  $\max_i C_i(x, t)$ .

An *exact* or *optimal* allocation is an allocation that, for the given input  $t$ , minimizes the makespan. A  $c$ -approximation is an allocation whose makespan is at most  $c$  times that of the optimal allocation. A deterministic algorithm  $A$  outputs an allocation  $x = A(t)$ . For a randomized algorithm  $A^{rand}$ ,  $A^{rand}(t)$  is a probability distribution over all possible allocations; we call  $A^{rand}(t)$  a randomized allocation.

*Mechanism Design* In order to characterize truthful mechanisms, we consider the allocations that are given in output for two inputs which differ only in one machine’s type. We let  $(\hat{t}_i, t_{-i})$  denote the vector  $(t_1, \dots, t_{i-1}, \hat{t}_i, t_{i+1}, \dots, t_m)$  obtained from  $t$  by replacing  $t_i$  with  $\hat{t}_i$ .

Given types  $t$  and a job allocation  $x$ , we count the number of  $L$ -jobs and the number of  $H$ -jobs allocated to machine  $i$  in  $x$ :

$$\begin{aligned} n_L^i(x, t) &:= |\{h : t_{ih} = L \text{ and } x_{ih} = 1\}|, \\ n_H^i(x, t) &:= |\{h : t_{ih} = H \text{ and } x_{ih} = 1\}|. \end{aligned}$$

We next introduce a monotonicity condition which characterizes truthful mechanisms for our domains. This condition (and the characterization) is a special case of a known characterization by Rochet [19] for general domains, as well as the corresponding characterization that Lavi and Swamy [12] derive for two-values domains. Rochet [19] proved that a mechanism is truthful if and only if its algorithm satisfies a so-called cycle monotonicity condition. For our domains, cycle monotonicity boils down to the following condition.

**Definition 1** (monotone algorithm) An algorithm  $A$  is monotone (in expectation) if, for any machine  $i$  and for any two inputs that differ only in the type of machine  $i$ ,  $t = (t_i, t_{-i})$  and  $\hat{t} = (\hat{t}_i, t_{-i})$ , the following inequality holds (in expectation):

$$n_L^i - n_H^i + \hat{n}_L^i - \hat{n}_H^i \geq 0 \tag{1}$$

where  $n_L^i = n_L^i(A(t), t)$ ,  $n_H^i = n_H^i(A(t), t)$ ,  $\hat{n}_L^i = n_L^i(A(\hat{t}), \hat{t})$ , and  $\hat{n}_H^i = n_H^i(A(\hat{t}), \hat{t})$ .

The results by Rochet [19] and by Lavi and Swamy [12] imply that truthfulness for our domains is equivalent to the monotonicity condition above. For completeness, we give a direct proof (using essentially the same arguments) since this will also allow us to provide a simple closed formula for the payments.

**Theorem 2** *For the case of two-values domains with publicly known partitions, there exist prices  $P$  such that the mechanism  $(A, P)$  is truthful (in expectation) iff  $A$  is monotone (in expectation).*

*Proof* Truthfulness is equivalent to the condition that, for every  $i$  and  $t = (t_i, t_{-i})$  and  $\hat{t} = (\hat{t}_i, t_{-i})$ , the following holds

$$\begin{aligned} P_i(t) - L \cdot n_L^i(x, t) - H \cdot n_H^i(x, t) &\geq P_i(\hat{t}) - L \cdot n_L^i(\hat{x}, t) - H \cdot n_H^i(\hat{x}, t) \\ P_i(\hat{t}) - L \cdot n_L^i(\hat{x}, \hat{t}) - H \cdot n_H^i(\hat{x}, \hat{t}) &\geq P_i(t) - L \cdot n_L^i(x, \hat{t}) - H \cdot n_H^i(x, \hat{t}) \end{aligned}$$

where  $x = A(t)$  and  $\hat{x} = A(\hat{t})$ . We can rewrite some of these terms in a more convenient form by observing that, by definition of our domain, the types  $t_i$  and  $\hat{t}_i$  are “complementary” meaning that a job has cost  $L$  (respectively,  $H$ ) in  $t_i$  if and only if in  $\hat{t}_i$  it has cost  $H$  (respectively,  $L$ ). Therefore

$$n_L^i(\hat{x}, t) = n_H^i(\hat{x}, \hat{t}), \quad n_H^i(\hat{x}, t) = n_L^i(\hat{x}, \hat{t}), \quad n_L^i(x, \hat{t}) = n_H^i(x, t), \quad n_H^i(x, \hat{t}) = n_L^i(x, t)$$

and truthfulness is equivalent to

$$P_i(t) - Ln_L^i - Hn_H^i \geq P_i(\hat{t}) - L\hat{n}_L^i - H\hat{n}_H^i \tag{2}$$

$$P_i(\hat{t}) - L\hat{n}_L^i - H\hat{n}_H^i \geq P_i(t) - Ln_L^i - Hn_H^i \tag{3}$$

where  $n_L^i = n_L^i(A(t), t)$ ,  $n_H^i = n_H^i(A(t), t)$ ,  $\hat{n}_L^i = n_L^i(A(\hat{t}), \hat{t})$ , and  $\hat{n}_H^i = n_H^i(A(\hat{t}), \hat{t})$ .

These two inequalities imply the monotonicity condition (1). That is, a mechanism  $(A, P)$  is truthful only if  $A$  is monotone.

We next prove that, if  $A$  is monotone, then there are payments  $P$  such that  $(A, P)$  is truthful. We set the payments as

$$P_i(t) = (n_L^i + n_H^i) \left( \frac{L + H}{2} \right)$$

$$P_i(\hat{t}) = (\hat{n}_L^i + \hat{n}_H^i) \left( \frac{L + H}{2} \right)$$

and observe that the truthfulness conditions (2–3) are satisfied since

$$\begin{aligned}
 (2) &\Leftrightarrow n_L^i \left( \frac{L+H}{2} - L \right) + n_H^i \left( \frac{L+H}{2} - H \right) \geq \hat{n}_L^i \left( \frac{L+H}{2} - H \right) + n_H^i \left( \frac{L+H}{2} - L \right) \\
 &\Leftrightarrow n_L^i \left( \frac{H-L}{2} \right) + n_H^i \left( \frac{L-H}{2} \right) \geq \hat{n}_L^i \left( \frac{L-H}{2} \right) + n_H^i \left( \frac{H-L}{2} \right) \\
 &\Leftrightarrow \left( \frac{H-L}{2} \right) (n_L^i - n_H^i + \hat{n}_L^i - \hat{n}_H^i) \geq 0 \Leftrightarrow (1)
 \end{aligned}$$

and a symmetric argument shows that (3)  $\Leftrightarrow$  (1). □

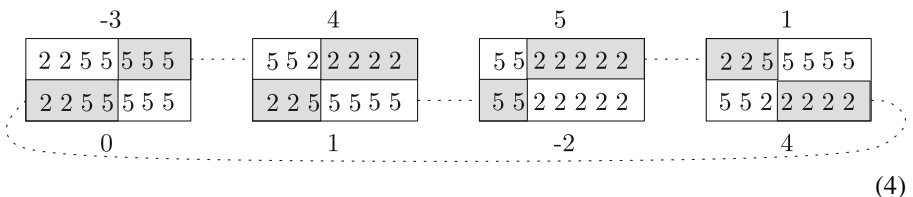
Throughout the paper, we refer to the quantity  $n_L^i(x, t) - n_H^i(x, t)$  as the *unbalance* of machine  $i$  in the allocation  $x$  with respect to type  $t$ . We also refer to the quantity in (1) as the *overall unbalance* of machine  $i$ . For any instance  $t$ , for any two machines  $i$  and  $j$ , and for any  $\alpha, \beta \in \{L, H\}$ , we consider the subset of jobs whose execution time is  $\alpha$  on machine  $i$  and  $\beta$  on machine  $j$ :

$$J_{\alpha\beta}^{ij}(t) := \{h : t_{ih} = \alpha \text{ and } t_{jh} = \beta\}.$$

### 1.3 An Illustrative Example

We begin with an example and show how we can use randomization to design optimal monotone-in-expectation algorithms from deterministic algorithms in the case of two-values domains with identical partitions.

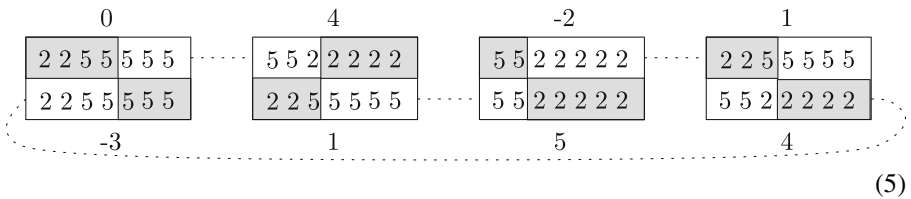
Consider the following instances along with their optimal allocation (gray box), and the unbalance  $n_L^i - n_H^i$  for each of the two machines (numbers outside the box):



Let machine 1 and machine 2 correspond to top and bottom machine, respectively. Note that we connect two instances by a dotted line if they differ in exactly one machine’s type, and the line indicates which one is that machine. Therefore the monotonicity condition (1) requires that, for any dotted line joining two instances  $t = (t_i, t_{-i})$  and  $\hat{t} = (\hat{t}_i, t_{-i})$ , the corresponding machine  $i$  has nonnegative overall unbalance: For example, the first two instances differ by machine 1 and the overall unbalance is  $-3 + 4 = 1$ . Observe that monotonicity condition (1) is violated by the allocations of the two middle instances which differ in the type of machine  $i = 2$  and the corresponding overall unbalance is  $1 - 2 = -1$ .

One may try to recover monotonicity by observing that the allocations in (4) are not the only ones that minimize the makespan. In the first and in the third input we

could swap the jobs between the two machines (leaving the allocations of the second and fourth instance unchanged):



Unfortunately, now the monotonicity condition is violated due to the allocation of the first machine in the last two instances.

The above instances have been used by Lavi and Swamy [12] to prove their lower bound for deterministic mechanisms. However, if we choose *randomly* between the allocation in (4) and the one in (5) with the same probability, the corresponding optimal randomized algorithm satisfies monotonicity *in expectation* (for example, in the first instance the unbalance becomes  $-3/2$  for both machines, while in the second instance it remains unchanged).

**Fact 3** *For two machines and two-values domains with identical partitions as above, with values  $L = 2$  and  $H = 5$ , no truthful mechanism can achieve an approximation factor better than 1.1 [12]. On the contrary, for the same problem there exists an exact truthful-in-expectation mechanism.*

In the sequel we show that this positive result holds in general for some of our domains, and not only in the very special instance where deterministic mechanisms cannot be optimal.

## 2 Identical Partitions

In this section we consider the case of machines with identical partitions and give a general (“black box”) method to convert scheduling deterministic algorithms into mechanisms that are truthful-in-expectation. The main result of this section is summarized by the following theorem.

**Theorem 4** *Every deterministic algorithm A for scheduling jobs on machines with identical partitions can be turned into a randomized mechanism M which is truthful-in-expectation and such that the allocation returned by M has makespan no greater than the one returned by A.*

Let  $(S, \bar{S})$  be a partition of the jobs, with  $|S| = s$  and  $|\bar{S}| = n - s$ , such that for each machine  $i$  we have  $S_i = S$ . Without loss of generality we can reorder the jobs in such a way that  $S = \{1, 2, \dots, s\}$  and  $\bar{S} = \{s + 1, s + 2, \dots, n\}$ . Since the partition of the jobs is public, the only information that is private to each machine is which side



of the partition contains its  $L$ -jobs. Thus, the type’s domain of each machine contains only two elements:

$$L_S = (L \cdots LH \cdots \cdots H) \text{ and } H_S = (H \cdots HL \cdots \cdots L)$$

For any instance  $t$ , we denote by  $m_S(t)$  and  $m_{\bar{S}}(t)$  the number of  $L_S$ -machines and  $H_S$ -machines in  $t$ , respectively. Clearly,  $m_S(t) + m_{\bar{S}}(t) = m$ . Given an allocation  $x$ , it is convenient to count, for each side of the partition, the number of jobs that are scheduled in  $x$  as  $L$ -jobs:

$$n_{L_S}(x, t) = |\{h \in S : \exists i \text{ such that } x_{ih} = 1 \text{ and } t_{ih} = L\}| \tag{6}$$

$$n_{L_{\bar{S}}}(x, t) = |\{h \in \bar{S} : \exists i \text{ such that } x_{ih} = 1 \text{ and } t_{ih} = L\}| \tag{7}$$

and  $n_L(x, t) := n_{L_S}(x, t) + n_{L_{\bar{S}}}(x, t) = \sum_i n_L^i(x, t)$  is the overall number of jobs allocated in  $x$  as  $L$ -jobs.

Following the idea described in Section 1.3, we show now how to obtain a randomized allocation from a deterministic one by randomly “shuffling” machines of the same type:

**Definition 5** For any deterministic allocation  $x$ , we denote by  $x^{(rand)}$  the randomized allocation obtained from  $x$  as follows:

- Pick an integer  $r \in \{0, \dots, m_S - 1\}$  uniformly at random, and set  $x_i^{(rand)} := x_{i+r \bmod m_S}$  for each  $L_S$ -machine  $i$ ;
- Pick an integer  $\bar{r} \in \{0, \dots, m_{\bar{S}} - 1\}$  uniformly at random, and set  $x_i^{(rand)} := x_{i+\bar{r} \bmod m_{\bar{S}}}$  for each  $H_S$ -machine  $i$ .

For any deterministic algorithm  $A$ , we let  $A^{(rand)}$  be the randomized algorithm that, on input  $t$ , returns the randomized allocation  $x^{(rand)}$  where  $x = A(t)$ .

Notice that  $n_{L_S}(x^{(rand)}, t) = n_{L_S}(x, t)$  and  $n_{L_{\bar{S}}}(x^{(rand)}, t) = n_{L_{\bar{S}}}(x, t)$ . In the following discussion we fix  $x$  and  $t$  and simply write  $n_{L_S}$  and  $n_{L_{\bar{S}}}$ .

For any  $L_S$ -machine  $i$ , its expected load consists of  $n_L^i = n_{L_S}/m_S$   $L$ -jobs and  $n_H^i = (n - s - n_{L_{\bar{S}}})/m_S$   $H$ -jobs. Thus the expected unbalance of an  $L_S$ -machine is

$$n_L^i - n_H^i = \frac{n_{L_S} - (n - s - n_{L_{\bar{S}}})}{m_S} = \frac{n_L - (n - s)}{m_S}.$$

Similarly, the expected load of an  $H_S$ -machine  $i$  consists of  $n_L^i = n_{L_{\bar{S}}}/m_{\bar{S}}$   $L$ -jobs and  $n_H^i = (s - n_{L_S})/m_{\bar{S}}$   $H$ -jobs and its expected unbalance is equal to

$$n_L^i - n_H^i = \frac{n_{L_{\bar{S}}} - (s - n_{L_S})}{m_{\bar{S}}} = \frac{n_L - s}{m_{\bar{S}}}.$$

**Lemma 6** Algorithm  $A^{(rand)}$  is monotone-in-expectation if the deterministic algorithm  $A$  is such that for any  $t = (L_S, t_{-i})$  and  $\hat{t} = (H_S, t_{-i})$ , it holds that

$$\frac{n_L - (n - s)}{m_S} + \frac{\hat{n}_L - s}{m_{\bar{S}} + 1} \geq 0 \tag{8}$$

where  $n_L = n_L(A(t), t)$  and  $\hat{n}_L = n_L(A(\hat{t}), \hat{t})$  denote the number of  $L$ -jobs allocated by  $A$  on input  $t$  and  $\hat{t}$ , respectively.

*Proof* Consider two instances  $t = (L_S, t_{-i})$  and  $\hat{t} = (H_S, t_{-i})$ . By the previous discussion we have that, for any  $i$ , the total unbalance of machine  $i$  is

$$n_L^i - n_H^i + \hat{n}_L^i - \hat{n}_H^i = \frac{n_L - (n - s)}{m_S} + \frac{\hat{n}_L - s}{\hat{m}_{\bar{S}}}$$

where  $\hat{m}_{\bar{S}}$  is the number of  $H_S$ -machines in  $\hat{t}$ . Since  $\hat{m}_{\bar{S}} = m_{\bar{S}} + 1$ , then (8) is equivalent to (1) and, by Definition 1,  $A^{(rand)}$  is monotone-in-expectation.  $\square$

### 2.1 Canonical Allocations

Lemma 6 says that in order to design an exact truthful-in-expectation mechanism for two-values domains with identical partitions it is sufficient to design a deterministic exact algorithm which satisfies the condition in (8). We now show that this is always possible by taking any exact algorithm  $A$  and transforming the allocations computed by  $A$  into “canonical” allocations. Intuitively, canonical allocations are allocations where it is not possible to swap jobs between two machines and increase the number of allocated  $L$ -jobs without increasing the makespan of the allocation. These allocations can be obtained from any allocation by repeatedly swapping jobs according to local rules (specified below). We will show that each of these swap operations increases the number of allocated  $L$ -jobs, without increasing the makespan.

To this end, we extend our previous notation.

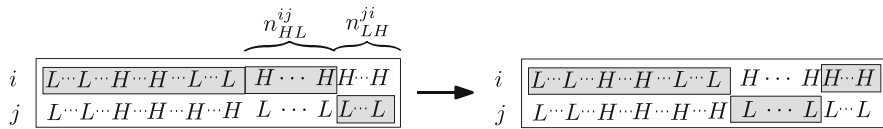
**Definition 7** Given an instance  $t$  and an allocation  $x$ , for any  $\alpha, \beta \in \{L, H\}$  and for any two machines  $i$  and  $j$ , we let  $n_{\alpha\beta}^{ij}(x, t)$  be the number of  $\alpha$ -jobs that are allocated to machine  $i$  and that are  $\beta$ -jobs for machine  $j$ :

$$n_{\alpha\beta}^{ij}(x, t) := |\{k : x_{ik} = 1, t_{ik} = \alpha, \text{ and } t_{jk} = \beta\}|.$$

Notice that  $n_{\alpha\beta}^{ij}$  is different from  $n_{\beta\alpha}^{ji}$  because the first index denotes the machine that has the jobs. We can now define canonical allocations.

**Definition 8** (canonical allocation) A canonical allocation (for instance  $t$ ) is an allocation obtained by modifying a deterministic allocation  $x$  as follows:

1. Apply the following **Rule R1** as long as possible: Suppose jobs  $h$  and  $k$  are allocated to machines  $i$  and  $j$ , respectively ( $x_{ih} = 1 = x_{jk}$ ). If  $t_{ik} \leq t_{ih}$  and  $t_{jh} < t_{jk}$  (i.e., no machine gets worse and at least one gets better if we swap the two jobs among the machines  $i$  and  $j$ ), then move job  $h$  to machine  $j$  and job  $k$  to machine  $i$  (set  $x_{ik} = x_{jh} = 1$  and  $x_{ih} = x_{jk} = 0$ ).
2. Apply the following **Rule R2** as long as possible: If  $n_{HL}^{ij}(x, t) > n_{LH}^{ji}(x, t)$  and  $j$  gets only jobs from  $J_{LH}^{ji}(t)$ , then move  $n_{HL}^{ij}$  jobs in  $J_{HL}^{ij}(t)$  from  $i$  to  $j$ , and move  $n_{LH}^{ji}$  jobs in  $J_{LH}^{ji}(t)$  from  $j$  to  $i$  (see Fig. 1).



**Fig. 1** The swapping Rule R2 in the definition of canonical allocation

**Fact 9** Both Rules R1 and R2 decrease the overall number of H-jobs by at least one. Thus, given  $x$  and  $t$ , it is possible to compute in polynomial time a canonical allocation  $x'$  (following the two steps in Definition 8) whose cost is not larger than the cost of  $x$ .

2.2 A Black-box Construction (Proof of Theorem 4)

We can assume without loss of generality that our deterministic algorithm always returns canonical allocations (Fact 9). The following two lemmas provide important structural properties of such canonical allocations.

**Lemma 10** Every allocation  $x$  that is canonical with respect to the instance  $t$  is such that either  $n_{L_S}(x, t) = s$  or  $n_{L_{\bar{S}}}(x, t) = n - s$ .

*Proof* Suppose for contradiction that  $n_{L_S}(x, t) < s$  and  $n_{L_{\bar{S}}}(x, t) < n - s$ . Then, in the allocation  $x$  there exists a job  $h \in S$  that is assigned to an  $H_S$ -machine  $i$  and a job  $l \in \bar{S}$  that is assigned to an  $L_S$ -machine  $j$ . Since  $t_{ih} = H$  and  $t_{jl} = H$  while  $t_{jh} = L$  and  $t_{il} = L$  then it is possible to apply Rule R1 to jobs  $j$  and  $l$ . But this contradicts the hypothesis that  $x$  is canonical with respect to  $t$ . □

**Lemma 11** If the allocation  $x$  is canonical with respect to the instance  $t$  then we have that:

1. If  $n_{L_S}(x, t) = s$  then  $n_{L_{\bar{S}}}(x, t) \geq \frac{m_{\bar{S}}}{m} \cdot (n - s)$ ;
2. If  $n_{L_{\bar{S}}}(x, t) = n - s$  then  $n_{L_S}(x, t) \geq \frac{m_S}{m} \cdot s$ .

*Proof* We prove only point 1 (proof of point 2 being similar).

Let  $x$  be an allocation that is canonical with respect to  $t$  and such that  $n_{L_S}(x, t) = s$ . Since  $n_{L_{\bar{S}}}(x, t)$  is equal to the number of jobs in  $\bar{S}$  that are allocated to  $H_S$ -machines then there are  $n - s - n_{L_{\bar{S}}}(x, t)$  jobs of  $\bar{S}$  that are allocated to  $L_S$ -machines (i.e. they are allocated as H-jobs). Therefore, there must be an  $L_S$ -machine  $i$  and an  $H_S$ -machine  $j$  such that  $i$  gets at least  $(n - s - n_{L_{\bar{S}}}(x, t))/m_S$  jobs from  $\bar{S}$  (and all these jobs are bad for  $i$  but good for  $j$ ) and  $j$  gets at most  $n_{L_{\bar{S}}}(x, t)/m_{\bar{S}}$  jobs from  $\bar{S}$  (they are good for  $j$  but bad for  $i$ ). Then, we have

$$n_{HL}^{ij} \geq \frac{n - s - n_{L_{\bar{S}}}(x, t)}{m_S}. \tag{9}$$

and

$$n_{LH}^{ji} \leq \frac{n_{L\bar{S}}(x, t)}{m_{\bar{S}}}. \tag{10}$$

Observe now that since by hypothesis  $n_{L_S}(x, t) = s$ , we have that  $j$  gets only jobs from  $\bar{S}$  and, since  $x$  is canonical then, by point 2 of Definition 8, it must be  $n_{HL}^{ij} \leq n_{LH}^{ji}$ . Thus, the following inequality holds:

$$\frac{n_{L\bar{S}}(x, t)}{m_{\bar{S}}} \geq \frac{n - s - n_{L\bar{S}}(x, t)}{m_S} \tag{11}$$

from which we obtain that

$$n_{L\bar{S}}(x, t) \cdot (1/m_S + 1/m_{\bar{S}}) = n_{L\bar{S}}(x, t) \cdot m / (m_S \cdot m_{\bar{S}}) \geq (n - s) / m_S,$$

and thus

$$n_{L\bar{S}}(x, t) \geq (n - s) \cdot \frac{m_{\bar{S}}}{m}.$$

□

We can now give a black-box construction to transform any deterministic algorithm into a randomized and monotone-in-expectation one that has no worse makespan.

*Proof Theorem 4* Let  $A$  be a deterministic scheduling algorithm and let  $A'$  be the algorithm that first calls  $A$  to compute an allocation and then transforms this allocation into a canonical one. We observe that the canonization process described in Definition 8 does not increase the makespan of the allocation. Thus, by Theorem 2 and Lemma 6, to prove the theorem it is sufficient to show that for any  $t = (L_S, t_{-i})$  and  $\hat{t} = (H_S, t_{-i})$  such that  $x = A'(t)$  and  $\hat{x} = A'(\hat{t})$

$$\frac{n_L - (n - s)}{m_S} + \frac{\hat{n}_L - s}{m - m_S + 1} \geq 0$$

where  $n_L$  is the number of  $L$ -jobs allocated in  $x$  and  $\hat{n}_L$  is the number of  $L$ -jobs allocated in  $\hat{x}$ .

Because of Lemma 10 there are only four cases to consider:

1. ( $n_{L_S} = s$  and  $n_{L\bar{S}} = n - s$ .) Observe that  $n_L + \hat{n}_L \geq n$ . Therefore

$$\begin{aligned} \frac{n_L - (n - s)}{m_S} + \frac{\hat{n}_L - s}{m - m_S + 1} &\geq \frac{n_L - (n - s) + \hat{n}_L - s}{\max\{m_S, m - m_S + 1\}} \\ &= \frac{n_L + \hat{n}_L - n}{\max\{m_S, m - m_S + 1\}} \\ &\geq 0. \end{aligned}$$

2. ( $n_{L\bar{S}} = n - s$  and  $n_{L_S} = s$ .) Also in this case  $n_L + \hat{n}_L \geq n$ . The rest of the proof goes like the previous case.

3. ( $n_{L_S} = s$  and  $n_{\hat{L}_S} = s$ .) Since  $m_{\bar{S}} = m - m_S$  and  $\hat{m}_{\bar{S}} = m_{\bar{S}} + 1$ , by Lemma 11, we have that

$$n_{L_{\bar{S}}} \geq \frac{m - m_S}{m} \cdot (n - s) \text{ and } \hat{n}_{L_{\bar{S}}} \geq \frac{m_{\bar{S}} + 1}{m} \cdot (n - s).$$

Hence

$$\begin{aligned} \frac{n_L - (n - s)}{m_S} + \frac{\hat{n}_L - s}{m - m_S + 1} &= \frac{s + n_{L_{\bar{S}}} - (n - s)}{m_S} + \frac{s + \hat{n}_{L_{\bar{S}}} - s}{m_{\bar{S}} + 1} \\ &\geq \frac{s + \frac{m - m_S}{m} \cdot (n - s) - (n - s)}{m_S} + \frac{\frac{m_{\bar{S}} + 1}{m} (n - s)}{m_{\bar{S}} + 1} \\ &\geq \frac{s}{m_S} - \frac{n - s}{m} + \frac{n - s}{m} \\ &\geq 0. \end{aligned}$$

4. ( $n_{L_{\bar{S}}} = n - s$  and  $n_{\hat{L}_{\bar{S}}} = n - s$ .) This case follows by symmetry with the previous case (simply exchange  $S$  with  $\bar{S}$ ).

□

### 2.3 Complexity Considerations

We stress that our black-box construction of truthful-in-expectation mechanisms (Theorem 4) can be carried out efficiently, i.e., in polynomial time. Indeed, any polynomial-time  $c$ -approximation algorithm can be transformed into a polynomial-time monotone algorithm with the same approximation guarantee. Moreover, the payments used in the proof of Theorem 4 can be computed directly from the allocation. In a recent paper [3] Bonifaci and Wiese gave a new PTAS for scheduling jobs on an arbitrary number of unrelated machines when these machines are of a limited number of types. Since, in our setting, machines are of only two types, we can use this result to show the following.

**Corollary 12** *For any number of machines with identical partitions and for any  $\epsilon > 0$ , there exists a polynomial-time truthful-in-expectation mechanism which returns a  $(1 + \epsilon)$ -approximate allocation.*

We shall prove in Section 4 that the same result cannot be achieved by (universally) truthful mechanisms, even if we allow them to run in exponential time.

### 3 Mechanisms for Two Machines

In this section we consider scheduling on two-values domains with publicly known partitions in the case of  $m = 2$  machines. In particular, we give an exact truthful-in-expectation mechanism for the case with uniform partitions (Section 3.1) and a deterministic  $3/2$ -approximation mechanism for the case of (restricted) publicly known partitions (Section 3.2).

#### 3.1 An Exact Truthful-in-Expectation Mechanism

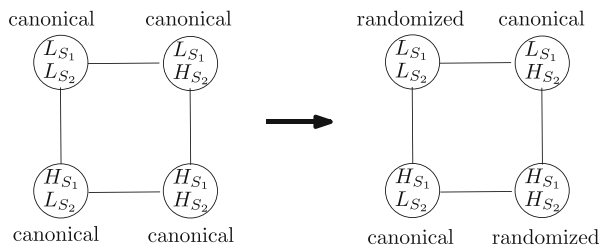
As in the case of identical partitions, we show that every exact algorithm can be turned into an exact monotone-in-expectation algorithm (by Theorem 2 this implies an exact truthful-in-expectation mechanism). Also in this case we restrict ourselves to scheduling algorithms that give in output only canonical allocations (this is without loss of generality by Fact 9) and obtain monotonicity-in-expectation by randomizing among several exact allocations. However, since the partitions are not identical, the randomization procedure will be applied only to *some* of the canonical allocations.

Consider the four possible inputs and let us represent them as the simple graph in Fig. 2 where two inputs are adjacent if and only if they differ in exactly the type of one machine:

Note that the monotonicity condition (1) involves only inputs that are adjacent in this graph. Our strategy is to prove that monotonicity is satisfied at every edge. To this aim, we define two classes of allocations and prove that all canonical allocations are in one of these two classes. We then show that the randomization procedure guarantees monotonicity for canonical allocations in each of the two possible classes.

**Definition 13** (allocation classes) An allocation  $x$  is classified with respect to the instance  $t$  as

- **$i$ -restricted** if machine  $i$  gets only a proper subset of the jobs in  $J_{LH}^{ij}(t)$ , where  $j \neq i$  denotes the other machine;



**Fig. 2** The strategy to obtain randomized monotone allocations for two machines: Each node corresponds to a possible input and two nodes are adjacent if and only if they differ in exactly one machine’s type; The monotonicity condition (1) involves only pairs of adjacent inputs, and our strategy is to turn two out of the four canonical allocations into randomized ones

- **symmetric** if, for any  $i \in \{1, 2\}$  and  $j \neq i$ , machine  $i$  gets all jobs in  $J_{LH}^{ij}(t)$ , no job in  $J_{HL}^{ij}(t)$ , and a (possibly empty) subset of the jobs in  $J_{LL}^{ij}(t) \cup J_{HH}^{ij}(t)$ .

We say that  $x$  is **restricted** if it is 1-restricted or 2-restricted. Restricted and symmetric allocations are shown in Fig. 3.

The following lemma proves that all exact canonical allocations belong to one of the classes defined in Definition 13.

**Lemma 14** *Every exact canonical allocation  $x$  with respect to an instance  $t$  is either symmetric or restricted.*

*Proof* Let  $x$  be an exact canonical allocation with respect to instance  $t$ . We remark that, by definition,  $J_{LH}^{12}(t) = J_{HL}^{21}(t)$  and  $J_{LH}^{21}(t) = J_{HL}^{12}(t)$ . This fact follows directly from Definition 8.

**Fact 15** *If the swapping Rule R1 cannot be applied to an allocation  $x$  with respect to  $t$ , then there is at most one machine  $i$  such that  $n_{HL}^{ij}(x, t) > 0$ , with  $j \neq i$ . In this case, machine  $j$  gets only jobs from the set  $J_{LH}^{ji}(t)$ .  $\square$*

Therefore it is not possible that in the allocation  $x$  machine 1 gets some jobs from  $J_{HL}^{12}(t)$  and machine 2 gets some jobs from  $J_{HL}^{21}(t)$ . Observe that if neither machine gets such jobs then the allocation  $x$  is of class symmetric (see Definition 13). Thus, we have only to prove that in all the other cases in the allocation  $x$  there is a machine  $i$  that gets only a subset of jobs in  $J_{LH}^{ij}(t)$ .

Suppose first that machine 1 gets some jobs from  $J_{HL}^{12}(t)$  and machine 2 gets no jobs from  $J_{HL}^{21}(t)$ . This means that all jobs in  $J_{HL}^{21}(t) = J_{HL}^{12}(t)$  are allocated to machine 1. To conclude that the solution is of class 2-restricted, it is sufficient to show that all jobs in  $J_{LL}(t) \cup J_{HH}(t)$  are allocated to machine 1. But, this is true since if machine 2 would get a job  $\bar{h} \in J_{LL}(t) \cup J_{HH}(t)$  then it would be possible to apply Rule-R1 (see Definition 8) to swap  $\bar{h}$  with a job  $h \in J_{HL}^{12}(t)$ , thus contradicting the hypothesis that the allocation is canonical.

Similarly, we have that if machine 2 gets some jobs from  $J_{HL}^{21}(t)$  and machine 1 gets no jobs from  $J_{HL}^{12}(t)$ , then the solution is of class 1-restricted.



**Fig. 3** The two allocation classes of Definition 13

The following two technical lemmas give bounds on the number of jobs allocated to machines in exact allocations.

**Lemma 16** *In any exact allocation for an instance  $t$ , if there is at least one job that can be assigned as  $L$ -job to one of the two machines (i.e., there is at least a machine  $i$  and a job  $k$  such that  $t_{ik} = L$ ), then no machine gets more than  $\lfloor n/2 \rfloor$   $H$ -jobs.*

*Proof* Consider the allocation in which machine  $i$  gets job  $k$  and any subset of  $\lfloor n/2 \rfloor$  other jobs, and the other machine  $j$  gets all the remaining  $n - (\lfloor n/2 \rfloor + 1) = \lfloor n/2 \rfloor$  jobs. The makespan of this allocation is at most  $L + \lfloor n/2 \rfloor H < H + \lfloor n/2 \rfloor H = \lceil n/2 \rceil H$ . That is, any allocation that assigns more than  $\lfloor n/2 \rfloor$   $H$ -jobs cannot be an exact allocation because it has larger makespan.  $\square$

**Lemma 17** *If an instance  $t$  admits an exact allocation of class  $i$ -restricted then machine  $i$  gets at least  $n/2$  jobs in  $J_{LH}^{ij}(t)$ , with  $i \neq j$ . This implies that  $|J_{LH}^{ij}(t)| > n/2$ .*

*Proof* Let  $x$  be an exact  $i$ -restricted allocation, for some  $i \in \{1, 2\}$ . By Definition 13, machine  $i$  gets in  $x$  only a proper subset of the jobs in  $J_{LH}^{ij}(t)$  and machine  $j$  gets at least one  $H$ -job. Then,  $n_L^i(x, t) = n_{LH}^{ij}(x, t)$  and  $n_H^i(x, t) = 0$ . Since  $x$  is an exact allocation,  $i$  must get more jobs than  $j$ , otherwise a new allocation with smaller makespan could be obtained by moving one job in  $J_{LH}^{ij}(t)$  from  $j$  to  $i$ . Thus,  $n_L^i(x, t) > n/2$ . To conclude the proof observe that  $n_L^i(x, t) = n_{LH}^{ij}(x, t) < |J_{LH}^{ij}(t)|$ , where the inequality is strict because  $i$  gets in  $x$  only a *proper* subset of the jobs in  $J_{LH}^{ij}(t)$ .  $\square$

We remark that in order to prove the monotonicity condition in (1) we would like to have allocations that have large unbalance. The following lemma says that, for two of the four possible inputs, exact canonical allocations can be converted into exact randomized allocations having a “good” expected unbalance on both machines.

**Lemma 18** *For every deterministic exact canonical allocation  $x$  for an instance*

$$t \in \{(L_{S_1}, L_{S_2}), (H_{S_1}, H_{S_2})\}$$

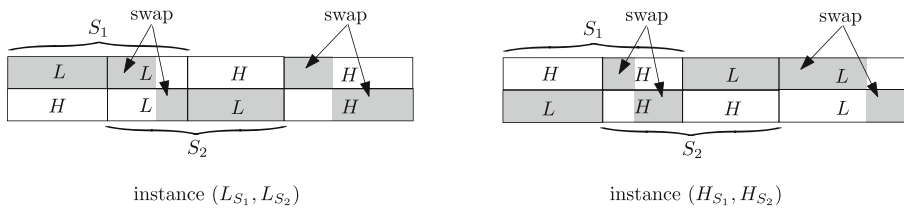
*with uniform partitions (i.e.  $|S_1| = |S_2|$ ), there exists a randomized allocation  $x^{(rand)}$  which gives an expected unbalance of  $\frac{n}{2} - |J_{HH}(t)|$  to both machines and has makespan equal to that of  $x$ .*

*Proof* Since the partitions are uniform, the subsets  $J_{LH}^{12}(t)$  and  $J_{HL}^{12}(t)$  have the same size

$$s - |S_1 \cap S_2|$$

in both instances (see Fig. 4). This implies that allocation  $x$  must be symmetric, for otherwise we would have one of the two subsets  $J_{LH}^{12}(t)$  or  $J_{LH}^{21}(t)$  with more than





**Fig. 4** Swapping operation to obtain allocation  $y$  from the symmetric allocation  $x$

$n/2$  elements (Lemma 14 and Lemma 17), which cannot be because they have the same size and they are disjoint.

Since allocation  $x$  is symmetric, we can build a new allocation  $y$  by swapping jobs in  $J_{LL}^{12}(t)$  and in  $J_{HH}^{12}(t)$  between the two machines (see Fig. 4). Clearly, the number of  $H$ -jobs is now exchanged between the two machines, that is

$$n_H^1(y, t) = n_H^2(x, t) \text{ and } n_H^2(y, t) = n_H^1(x, t).$$

Since  $|J_{LH}^{12}(t)| = |J_{HL}^{12}(t)|$ , the same happens with respect to the  $L$ -jobs, that is

$$n_L^1(y, t) = n_L^2(x, t) \text{ and } n_L^2(y, t) = n_L^1(x, t).$$

This means that  $y$  has the same makespan as  $x$ . Moreover, if we pick at random with probability  $1/2$  between allocation  $x$  and allocation  $y$ , in the resulting randomized allocation  $x^{(rand)}$ , the expected number of  $L$ -jobs assigned to any of the two machines is equal to

$$\frac{n_L^1(x, t) + n_L^2(x, t)}{2} = \frac{n - |J_{HH}(t)|}{2},$$

while the expected number of  $H$ -jobs is equal to

$$\frac{n_H^i(x, t) + n_H^j(x, t)}{2} = \frac{|J_{HH}(t)|}{2}.$$

The expected unbalance of each machine is thus  $n/2 - |J_{HH}(t)|$ . Clearly, since  $y$  and  $x$  have the same makespan, the makespan of  $x^{(rand)}$  is the same as the same makespan of  $x$ . □

We are now in a position to prove the main result of this section:

**Theorem 19** *For  $m = 2$  machines there exists an exact truthful-in-expectation mechanism for scheduling on two-values domains with uniform partitions.*

*Proof* Let  $A$  be any exact deterministic algorithm for scheduling on two machines that gives in output only canonical allocations. We can define a randomized exact algorithm which first computes  $x = A(t)$ , and then replaces  $x$  by the randomized allocation  $x^{(rand)}$  of Lemma 18 in two out of the four possible input instances (see Fig. 2):

$$A^{(rand)}(t) := \begin{cases} x^{(rand)} & \text{if } t \in \{(L_{S_1}, L_{S_2}), (H_{S_1}, H_{S_2})\} \\ x & \text{if } t \in \{(H_{S_1}, L_{S_2}), (L_{S_1}, H_{S_2})\} \end{cases}$$

By Lemma 18,  $A^{(rand)}$  is also an exact algorithm. So, we only need to prove that the monotonicity condition in (1) holds in expectation (Theorem 2).

Observe that any two instances  $t = (t_i, t_{-i})$  and  $\hat{t} = (\hat{t}_i, \hat{t}_{-i})$  that differ only on the type of one machine  $i$  correspond to a pair of adjacent nodes in the graph in Fig. 2. We can thus assume without loss of generality that  $t$  denotes the instance for which  $A^{(rand)}$  returns a randomized allocation  $x^{(rand)}$  and  $\hat{t}$  denotes the instance for which it return a deterministic exact canonical allocation  $\hat{x}$ ,

$$t \in \{(L_{S_1}, L_{S_2}), (H_{S_1}, H_{S_2})\} \quad \text{and} \quad \hat{t} \in \{(L_{S_1}, H_{S_2}), (H_{S_1}, L_{S_2})\}.$$

Consider the overall expected unbalance of machine  $i$ ,

$$UN^i := n_L^i - n_H^i + \hat{n}_L^i - \hat{n}_H^i \tag{12}$$

where  $n_L^i = n_L^i(x^{(rand)}, t)$ ,  $n_H^i = n_H^i(x^{(rand)}, t)$ ,  $\hat{n}_L^i = n_L^i(\hat{x}, \hat{t})$  and  $\hat{n}_H^i = n_H^i(\hat{x}, \hat{t})$ . By Lemma 18, we know that  $n_L^i - n_H^i = n/2 - |J_{HH}(t)|$  and thus

$$UN^i = n/2 - |J_{HH}(t)| + \hat{n}_L^i - \hat{n}_H^i.$$

To show that this quantity is positive we distinguish two cases:

**( $\hat{x}$  is  $i$ -restricted)** Machine  $i$  gets no  $H$ -jobs (Definition 13) and at least  $n/2$   $L$ -jobs (Lemma 17), which means that

$$\hat{n}_L^i - \hat{n}_H^i \geq n/2$$

thus implying

$$\begin{aligned} UN^i &\geq n/2 - |J_{HH}(t)| + n/2 \\ &= n - |J_{HH}(t)| \\ &\geq 0. \end{aligned}$$

**( $\hat{x}$  is not  $i$ -restricted)** By Lemma 14, allocation  $\hat{x}$  is either symmetric or  $j$ -restricted with  $j \neq i$ . In both cases, machine  $i$  gets all jobs in  $J_{LH}^{ij}(\hat{t})$ . Since  $\hat{t} \in \{(L_{S_1}, H_{S_2}), (H_{S_1}, L_{S_2})\}$  and because the partitions have uniform size, in instance  $\hat{t}$  there is at least one job that can be assigned as an  $L$ -job to one of the two machines. Thus, by Lemma 16, in allocation  $\hat{x}$  machine  $i$  gets at most  $n/2$   $H$ -jobs. Therefore we have

$$\hat{n}_L^i - \hat{n}_H^i \geq |J_{LH}^{ij}(\hat{t})| - n/2.$$

Observe that  $J_{LH}^{ij}(\hat{t}) = J_{HH}^{ij}(t)$ , which implies

$$\begin{aligned} UN^i &\geq n/2 - |J_{HH}(t)| + |J_{HH}(t)| - n/2 \\ &= 0. \end{aligned}$$

We have thus shown that  $A^{(rand)}$  is monotone in expectation. □

### 3.2 A Deterministic 3/2-Approximation Truthful Mechanism

Now we switch to deterministic mechanisms and we prove the following result:

**Theorem 20** For  $m = 2$  machines there exists a polynomial-time  $3/2$ -approximation deterministic truthful mechanism for scheduling for two-values domains with (unrestricted) publicly known partitions.

In order to prove this result we exhibit a monotone  $3/2$ -approximation algorithm. We were unable to extend the mechanism for the case of three or more machines so this is left as a natural open problem.

*The algorithm* On input  $t$ , the algorithm partitions the jobs into three subsets:

$$J_{LL}(t), J_{HH}(t) \text{ and } J_{LHHL}(t) := J_{LH}^{12}(t) \cup J_{HL}^{12}(t)$$

First, the algorithm allocates jobs in  $J_{LHHL}(t)$ , and then completes the allocation by dividing “evenly” the other jobs in  $J_{LL}(t)$  and in  $J_{HH}(t)$ . Some careful tie breaking rule will be used here to deal with the case in which some of these subsets of jobs have odd cardinality.

The algorithm consists of the following two steps (in the sequel we do not specify the input “ $t$ ”):

1. **Step 1 (allocate jobs in  $J_{LHHL}$ )** We allocate these jobs depending on the class of the canonical exact allocation (see Definition 13) for all jobs:
  - (a) **(class  $i$ -restricted.** Compute a canonical exact allocation for  $J_{LHHL}$ .
  - (b) **(class symmetric).** Assign all jobs in  $J_{LHHL}$  as  $L$ -jobs.

We denote by  $J_{LHHL}^i$  the set of jobs that are assigned to machine  $i$  in this first step, and let  $C_i^{(LHHL)}$  be the corresponding cost of machine  $i$ .

2. **Step 2 (allocate jobs in  $J_{LL}$  and  $J_{HH}$ )** For a set of jobs  $S$ , we denote by  $\lfloor S/2 \rfloor$  and  $\lceil S/2 \rceil$  an arbitrary subset of  $S$  of cardinality  $\lfloor |S|/2 \rfloor$  and  $\lceil |S|/2 \rceil$ , respectively. We define the set  $J^i$  of all jobs that are assigned to machine  $i$  at the end of this step (which includes the jobs  $J_{LHHL}^i$  assigned in the previous step) as follows: For  $i$  and  $j$  satisfying  $C_i^{(LHHL)} \geq C_j^{(LHHL)}$ , we allocate to  $i$  the set

$$J^i := \begin{cases} J_{LHHL}^i \cup \left\lfloor \frac{J_{HH}}{2} \right\rfloor \cup \left\lceil \frac{J_{LL}}{2} \right\rceil & \text{if both } |J_{LL}| \text{ and } |J_{HH}| \text{ are odd;} \\ J_{LHHL}^i \cup \left\lfloor \frac{J_{HH}}{2} \right\rfloor \cup \left\lfloor \frac{J_{LL}}{2} \right\rfloor & \text{otherwise,} \end{cases}$$

and thus machine  $j$  gets the rest of the jobs

$$J^j := [n] \setminus J^i.$$

*Approximation guarantee* The two steps of the algorithm keep a small difference between the completion times of the two machines as is formally demonstrated in the following:

**Lemma 21** *After each step, the difference between the two completion times is at most the optimum, that is*

$$\max_i C_i^{(LHHL)} \leq OPT \quad \text{and} \quad \max_i C_i \leq \min_i C_i + OPT.$$

*Proof* As for the first part, let  $J_{LHHL}$  be not empty (otherwise the statement trivially holds). Recall that there are two cases for the allocation of jobs in  $J_{LHHL}$ . In the first case (canonical exact is not symmetric) the algorithm allocates  $J_{LHHL}$  identically to the exact allocation on this set, and therefore the statement holds. In the second case (canonical exact is symmetric), the algorithm allocates all jobs as  $L$ -jobs. This is exactly what the exact allocation does, so the statement follows.

For the second statement, if the machine with higher  $C_i^{(LHHL)}$  determines the final makespan, and both  $J_{LL}, J_{HH}$  have odd size, then

$$\begin{aligned} \max_i C_i &= \max_i C_i^{(LHHL)} + \left\lfloor \frac{J_{HH}}{2} \right\rfloor H + \left\lceil \frac{J_{LL}}{2} \right\rceil L \\ &\leq \min_i C_i^{(LHHL)} + OPT + \left\lfloor \frac{J_{HH}}{2} \right\rfloor H + \left\lceil \frac{J_{LL}}{2} \right\rceil L \\ &\leq \min_i C_i^{(LHHL)} + \left\lceil \frac{J_{HH}}{2} \right\rceil H + \left\lfloor \frac{J_{LL}}{2} \right\rfloor L + OPT \\ &= \min_i C_i + OPT, \end{aligned}$$

while if not both  $J_{LL}, J_{HH}$  are odd,

$$\begin{aligned} \max_i C_i &= \max_i C_i^{(LHHL)} + \left\lfloor \frac{J_{HH}}{2} \right\rfloor H + \left\lfloor \frac{J_{LL}}{2} \right\rfloor L \\ &\leq \min_i C_i^{(LHHL)} + OPT + \left\lfloor \frac{J_{HH}}{2} \right\rfloor H + \left\lfloor \frac{J_{LL}}{2} \right\rfloor L \\ &\leq \min_i C_i + OPT. \end{aligned}$$

Finally, if the machine with lower  $C_i^{(LHHL)}$  determines the makespan, then

$$\begin{aligned} \max_i C_i &\leq \min_i C_i^{(LHHL)} + \left\lfloor \frac{J_{HH}}{2} \right\rfloor H + \left\lfloor \frac{J_{LL}}{2} \right\rfloor L + H \\ &\leq \max_i C_i^{(LHHL)} + \left\lfloor \frac{J_{HH}}{2} \right\rfloor H + \left\lfloor \frac{J_{LL}}{2} \right\rfloor L + H \\ &\leq \min_i C_i + OPT. \end{aligned}$$

□

*Proof of approximation guarantee* Then we prove the  $3/2$ -approximation guarantee by distinguishing the two cases in Step 1 of the algorithm (allocation of jobs in  $J_{LHHL}$ ) which correspond to the class of the canonical exact allocation for the whole instance.

(class 1–restricted or 2–restricted). Recall that an allocation of class 2–restricted assigns a *proper* subset of these jobs to machine 2, thus implying that machine 1 must get at least one  $H$ -job from this set. (The case of the solution for the class 1–restricted is similar). Therefore

$$OPT \geq OPT_1 \geq |J_{LL}|L + |J_{HH}|H + H.$$

On the other hand, by using Lemma 21, we obtain

$$2APX = 2 \max_i C_i \leq C_1 + C_2 + H = |J_{LL}|L + |J_{HH}|H + C_1^{(LHHH)} + C_2^{(LHHH)} + H \leq 3OPT.$$

(class symmetric). Consider the number of jobs from respectively  $J_{LL}$  and  $J_{HH}$  that the optimum  $x^{opt}$  assigns to machine 1:

$$n_{LL}^{12} := n_{LL}^{12}(x^{opt}, t) \text{ and } n_{HH}^{12} := n_{HH}^{12}(x^{opt}, t).$$

This gives the following lower bound on the optimum (the costs of the two machines):

$$\begin{aligned} OPT &\geq n_{LL}^{12}L + n_{HH}^{12}H + |J_{LH}^{12}|L \\ OPT &\geq (|J_{LL}| - n_{LL}^{12})L + (|J_{HH}| - n_{HH}^{12})H + |J_{LH}^{21}|L \end{aligned}$$

and thus (by summing up)

$$2OPT \geq |J_{LL}|L + |J_{HH}|H + (|J_{LH}^{12}| + |J_{LH}^{21}|)L.$$

In this scenario, our algorithm assigns all jobs in  $J_{LHHH} = J_{LH}^{12} \cup J_{LH}^{21}$  as  $L$ -jobs and thus the two costs satisfy

$$C_1^{(LHHH)} + C_2^{(LHHH)} = |J_{LH}^{12}|L + |J_{LH}^{21}|L.$$

Therefore

$$\begin{aligned} 2APX = 2 \max_i C_i &\leq C_1 + C_2 + H = |J_{LL}|L + |J_{HH}|H + C_1^{(LHHH)} + C_2^{(LHHH)} + H \\ &\leq |J_{LL}|L + |J_{HH}|H + H + 2OPT - |J_{LL}|L - |J_{HH}|H \leq 3OPT, \end{aligned}$$

Hence the approximation guarantee has been shown for both cases. □

Observe that the performance guarantee of Theorem 20 is tight. In fact, for the instance  $|J_{LL}| = 2, |J_{HH}| = 1, |J_{LHHH}| = 0$ , with  $H = 2L$ , we have that the optimal makespan is  $2L$  while the makespan of the algorithm is  $3L$ .

*Monotonicity* First observe that the algorithm assigns to machine  $i$  at least half (rounded up or down) of its  $L$ -jobs in  $t$ , and at most half (rounded up or down) of its  $H$ -jobs in  $t$ . In particular,

$$n_{LH}^{ij} \geq \left\lceil \frac{J_{LH}^{ij}}{2} \right\rceil, n_{LL}^{ij} \geq \left\lfloor \frac{J_{LL}^{ij}}{2} \right\rfloor \text{ and } n_{HL}^{ij} \leq \left\lfloor \frac{J_{HL}^{ij}}{2} \right\rfloor, n_{HH}^{ij} \leq \left\lceil \frac{J_{HH}^{ij}}{2} \right\rceil.$$

For the jobsets  $J_{LL}$  and  $J_{HH}$  this is immediate. Now, let us consider the jobset  $J_{LH}^{ij}$  (the other case is symmetric). If we are in the symmetric case, the algorithm assigns

all these jobs to  $i$ . Otherwise, the algorithm computes the canonical optimum on  $J_{LHHL}$ . Suppose that  $n_{LH}^{ij} < \lceil \frac{J_{LH}^{ij}}{2} \rceil$ . Optimality implies that machine  $i$  gets at least one  $H$  job from the set  $J_{HL}^{ij}$ . But then the allocation is not canonical (nor optimal) since Rule 2 of Definition 8 can be applied. Therefore we obtain

$$n_L^i \geq \left\lceil \frac{J_{LH}^{ij}}{2} \right\rceil + \left\lfloor \frac{J_{LL}^{ij}}{2} \right\rfloor, \quad n_H^i \leq \left\lceil \frac{J_{HH}^{ij}}{2} \right\rceil + \left\lfloor \frac{J_{HL}^{ij}}{2} \right\rfloor. \quad (13)$$

Second, observe that if the type of machine  $i$  flips from  $t_i$  to  $\hat{t}_i$ , then  $|\hat{J}_{\alpha\beta}^{ij}| = |J_{\alpha\beta}^{ij}|$  (here  $\bar{L} = H$  and  $\bar{H} = L$ ). Applying (13) for both  $t_i, \hat{t}_i$  and using the last identity, we finally obtain (1).

*Complexity* To prove that our mechanism is polynomial we have to show that both the allocation and the prices can be computed in polynomial time. Observe that the allocation algorithm has first to compute an exact canonical allocation to decide how to perform his first step. Since, the cost of the allocation depends only on how many  $L$  jobs and  $H$  jobs are assigned to each of the two machines we can compute an exact allocation in polynomial-time by a brute-force algorithm. Then, by Fact 9, this allocation can be transformed into an exact canonical allocation in polynomial time. Given an exact canonical allocation, the first step of the algorithm takes polynomial time to compute an allocation of the jobs in  $J_{LHHL}$  while the second step takes constant time to complete this allocation with the jobs in  $J_{LL}$  and  $J_{HH}$ . Finally, by Theorem 2, we can compute prices of our mechanism directly from the allocation in constant time.

#### 4 Lower Bounds and Separation Results

The next theorem says that truthful-in-expectation mechanisms are provably *more powerful* than universally truthful mechanisms. Indeed, for this problem version, *exact* truthful-in-expectation mechanism exist for any number of machines (see Theorem 4 and Corollary 12). The proof combines the idea of the lower bound by Lavi and Swamy [12] with a technique used by Mu'alem and Schapira [16]: They use Yao's Min-Max Principle to obtain lower bounds on universally truthful mechanisms.

**Theorem 22** *No universally truthful mechanism can achieve an approximation factor better than 31/30 for scheduling on two machines, even for the case of identical partitions.*

*Proof* A universally truthful mechanism is simply a probability distribution over deterministic truthful mechanisms [17]. A lower bound on the approximation ratio of *any* universally truthful mechanism can be obtained via Yao's Min-Max Principle as illustrated in [16]: find a probability distribution over all possible inputs such that

every deterministic mechanism (monotone algorithm) has an expected approximation guarantee of  $c$  or worse.

*Example 23* (two-machines identical partitions) We have the four inputs in the example in Section 1.3. We know that any deterministic algorithm must err on at least one of these four inputs and this implies that (because of the values) on this input the approximation is  $10/9 \approx 1.111$  (the deterministic lower bound). Taking the uniform distribution over these four inputs, Yao’s principle tells us that every universally truthful mechanism must have an (expected) approximation which is not better than  $(1/4) \cdot (10/9) + 3/4 = 37/36 \approx 1.0277$ .  $\square$

The previous bound can be improved by optimizing the probability distribution. In particular, we assign positive probabilities only to *three* out of the four inputs shown here:

$$\begin{array}{r}
 \text{probability} \\
 \text{input} \\
 \text{APX of alternative allocation}
 \end{array}
 \left|
 \begin{array}{|c|c|c|}
 \hline
 p & 1 - 2p & p \\
 \hline
 \boxed{5522222} & \boxed{5522222} & \boxed{2255555} \\
 \boxed{2255555} & \boxed{5522222} & \boxed{5522222} \\
 \hline
 10/9 & 11/10 & 10/9 \\
 \hline
 \end{array}
 \right.
 \tag{14}$$

Every deterministic monotone algorithm must change the output in at least one of these three inputs (because monotonicity is not satisfied between the first two inputs and between the last two). Deterministic algorithms that use an alternative allocation in the first (or the third) input have an expected approximation at least

$$p \cdot (10/9) + (1 - p) = 1 + p/10$$

while for those algorithms that use an alternative allocation in the second input, the approximation is at least

$$(1 - 2p)11/10 + 2p = 1 + 1/10 - 2p/10$$

Taking  $p = 1/3$  we equate these two quantities and thus obtain that every deterministic algorithm must have an approximation of at least  $31/30 \approx 1.0333$ . Therefore Yao’s Min-Max Principle yields Theorem 22.

*Lower bounds for three-values domains* The above bounds can be strengthened by considering three-values domains (which are no-longer “single-bit”). First, we give an alternative proof for the lower bound of 2 for deterministic mechanisms on two machines, first showed by Nisan and Ronen in [17]. The proof in [17] requires that the input domain consists of at least 4 different values. Here, we extend the proof in order to hold even when the domain consists of 3 different values.

**Theorem 24** *For two machines and the case in which the processing times can take three values, no (deterministic) truthful mechanism can achieve an approximation factor better than 2.*

*Proof* Assume that we have only two machines and an *odd* number of jobs, that is  $n = 2k + 1$ . For every machine  $i$  and job  $h$ , the processing time can only take the following three values  $t_{ih} \in \{0, 1, 1 + \epsilon\}$ .

Let us assume that the input is  $t_{ih} = 1$ , for all machines  $i$ , and for all jobs  $h$ . Let  $x = A(t)$  be the allocation matrix determined by the mechanism  $M = (A, P)$ . If one of the machines gets all the jobs then the algorithm has makespan  $2k + 1$ , while the opt has makespan  $k + 1$ , and so the approximation ratio can be arbitrarily close to 2 for large arbitrarily large values of  $k$ . If the jobs are allocated to both machines, let  $j$  be the machine that gets a subset  $S_j$  of jobs with odd cardinality. Now produce the following input  $\hat{t}$  by changing the type of  $j$  only:

$$\hat{t}_{ih} = \begin{cases} 0, & i = j \text{ and } h \in S_j \\ 1 + \epsilon, & i = j \text{ and } h \notin S_j \\ t_{ih}, & \text{otherwise} \end{cases}$$

Let  $\hat{x} = A(\hat{t})$ , be the allocation for the input  $\hat{t}$ . Because the mechanism is truthful, it should also be monotone (see e.g. [6]):

$$\sum_{h \in S_j} (t_{jh} - \hat{t}_{jh}) \cdot (x_{jh} - \hat{x}_{jh}) \leq 0.$$

From this we get that the subset of jobs that machine  $j$  gets for the input  $\hat{t}$  is also  $S_j$ . The makespan of the mechanism for the input  $\hat{t}$  is  $|\bar{S}_j|$ , where  $\bar{S}_j = [n] \setminus S_j$ , while the optimal makespan is  $|S_j|/2$ . □

We next strengthen the lower bound on universally truthful mechanisms of Theorem 22 by considering domains of three values. The proof consists of a suitable probability distribution over the instances used for the deterministic lower bound.

**Theorem 25** *No universally truthful mechanism can achieve an approximation factor better than 9/8 for scheduling on two machines and three jobs that take three values.*

*Proof* There are three jobs and  $t$  is the matrix with all processing times equal 1. Besides the two allocations which assign all jobs to one machine, there are 6 non-trivial allocations:

$$\begin{bmatrix} \underline{111} \\ \underline{111} \end{bmatrix} \dots \begin{bmatrix} \underline{111} \\ \underline{111} \end{bmatrix} \tag{15}$$

We next show that every deterministic monotone algorithm which gives one of these allocations cannot have an approximation better than 2. Consider a modified instance for each of these allocations:

$$\begin{bmatrix} \underline{\epsilon\delta\delta} \\ \underline{111} \end{bmatrix} \dots \begin{bmatrix} \underline{111} \\ \underline{\delta\delta\epsilon} \end{bmatrix} \tag{16}$$

where  $\delta = 1 + \epsilon$  and  $\epsilon$  is some tiny number. Every monotone deterministic algorithm which chooses one of the allocations in (16) is forced to output the corresponding allocation in (16) (this is the same as in the proof of Theorem 24 when considering three jobs). The approximation is then  $2/(1 + \epsilon)$  which, for tiny  $\epsilon$ , gets arbitrarily close to 2.

Assign a probability  $p$  to each of the instances in (16) and probability  $q = (1 - 6p)$  to the instance of all 1's. Every deterministic algorithm which allocates all jobs to the



same machine has expected approximation at least

$$q(3/2) + 1 - q = 1 + q/2 = 1 + 1/2 - 3p$$

while every deterministic monotone algorithm using on of the allocations in (15) has expected approximation at least

$$2p + 1 - p = 1 + p$$

Taking  $p = 1/8$  these two quantities are equal and thus every deterministic monotone algorithm has an expected approximation at least  $9/8$ . Yao's Min-Max Principle implies that this lower bound applies to every universally truthful mechanism.  $\square$

## 5 Conclusion

In this paper we studied scheduling on selfish machines for two-values input domains with publicly known partitions. We considered three different models of publicly known partitions and presented both exact randomized truthful-in-expectation and approximation deterministic mechanisms and proved some lower bounds.

Our work leaves several open questions. First of all, we are able to derive exact truthful-in-expectation mechanisms for an arbitrary number of machines only in the case where partitions are identical. Moreover, exact truthful-in-expectation mechanisms for two machines are given only for uniform partitions. A natural open question is whether it is possible to extend the result to more general cases like (1) any number of machines with uniform partitions, or (2) two machines with unrestricted given partitions? Finally we note that the lower bound of 2 for three-values domains does not consider jobs' partitions (as our upper bounds do) and thus it would be natural/interesting to prove a lower bound for three-values domains with publicly known partitions.

**Acknowledgments** Part of this work was done while the second author was at the Max-Planck Institute for Informatics, Saarbrücken, while visiting the Università di Salerno, and was also supported by EPSRC grant EP/K01000X/1. Research of the first author partially supported by the PRIN 2011 research project ARS TechnoMedia – Algorithmics for Social Technological Networks, funded by the Italian Ministry of University and Research.

## References

1. Ashlagi, I., Dobzinski, S., Lavi, R.: An optimal lower bound for anonymous scheduling mechanisms. In Proc. of the 10th ACM Conference on Electronic Commerce (EC) (2009)
2. Archer, A., Tardos, É.: Truthful mechanisms for one-parameter agents. In Proc. of the 42nd Annual Symposium on Foundations of Computer Science (FOCS) (2001)
3. Bonifaci, V., Wiese, A.: Scheduling Unrelated Machines of Few Different Types. arXiv:1205.0974 (2012)
4. Christodoulou, G., Kovács, A.: A deterministic truthful PTAS for scheduling related machines. In Proc. of the 21st SIAM Symposium on Discrete Algorithms (SODA) SIAM (2010)
5. Christodoulou, G., Koutsoupias, E., Kovács, A.: Mechanism design for fractional scheduling on unrelated machines. ACM Trans. Algorithm **6**(2) (2010)

6. Christodoulou, G., Koutsoupias, E., Vidali, A.: A lower bound for scheduling mechanisms. *Algorithmica* **55**(4), 729–740 (2009)
7. Clarke, E.H.: Multipart pricing of public goods. *Public Choice* **8** (1971)
8. Dhangwatnotai, P., Dobzinski, S., Dughmi, S., Roughgarden, T.: Truthful approximation schemes for single-parameter agents. *SIAM J. Comput.* **40**(3), 915–933 (2011)
9. Epstein, L., Levin, A., van Stee, R.: A unified approach to truthful scheduling on related machines. In *Proc. of the 24-th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA)* (2013)
10. Groves, T.: Incentives in teams. *Econometrica* **41**(4), 617–631 (1973)
11. Koutsoupias, E., Vidali, A.: A lower bound of  $1+\phi$  for truthful scheduling mechanisms. In *Proc. of the 32-nd International Symposium on Mathematical Foundations of Computer Science (MFCS)* (2007)
12. Lavi, R., Swamy, C.: Truthful mechanism design for multidimensional scheduling via cycle monotonicity. *Games and Econ. Behav.* **67**(1), 99–124 (2009)
13. Lu, P.: On 2-player randomized mechanisms for scheduling. In *Proc. of the 5th International Workshop on Internet and Network Economics (WINE)* (2009)
14. Lu, P., Yu, C.: An improved randomized truthful mechanism for scheduling unrelated machines. In *Proc. of the 25th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1 of LIPIcs (2008)
15. Lu, P., Yu, C.: Randomized truthful mechanisms for scheduling unrelated machines. In *Proc. of the 4th International Workshop on Internet and Network Economics (WINE)* (2008)
16. Mu'alem, A., Schapira, M.: Setting lower bounds on truthfulness. In *Proc. of the 18-th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2007)
17. Nisan, N., Ronen, A.: Algorithmic mechanism design. *Games and Econ. Behav.* **35**, 166–196 (2001)
18. Nisan, N., Roughgarden, T., Tardos, É., Vazirani, V.V.: *Algorithmic Game Theory* Cambridge University Press (2007)
19. Rochet, J.-C.: A necessary and sufficient condition for rationalizability in a quasi-linear context. *J. Math. Econ.* **16**(2), 191–200 (1987)
20. Vickrey, W.: Counterspeculations, auctions and competitive sealed tenders. *J. Financ* **16**, 8–37 (1961)