Mechanisms for Scheduling with Single-Bit Private Values[⋆]

Vincenzo Auletta[1], George Christodoulou[2] [⋆⋆], and Paolo Penna[1]

[1] Dipartimento di Informatica, Università di Salerno, Italy
{auletta,penna}@dia.unisa.it
[2] Computer Science Department, University of Liverpool, United Kingdom,
gchristo@liv.ac.uk

Abstract. We consider randomized mechanisms for multi-dimensional scheduling. Following Lavi and Swamy [10], we study a setting with restrictions on the domain, while still preserving multi-dimensionality. In a sense, our setting is the simplest multi-dimensional setting, where each machine holds privately only a single-bit of information.
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.

1 Introduction

Designing truthful mechanisms for scheduling problems was first suggested in the seminal paper by Nisan and Ronen [15], 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 is a *private* value held by a selfish agent that controls the machine. Therefore this value might be misreported to the designer if this is advantageous to the agent. Mechanism design suggests using monetary compensation to incentivize agents to report truthfully. *Truthfulness* is desired, because it facilitates the prediction of the outcome and at the same time simplifies the agents' way of reasoning.

[⋆] Research partially supported by the PRIN 2008 research project COGENT (COmputational and GamE-theoretic aspects of uncoordinated NeTworks), funded by the Italian Ministry of University and Research.

[⋆⋆] Part of this work was done while the author was at the Max-Planck Institute for Informatics, Saarbrücken, and while visiting University of Salerno.

The challenge is to design truthful mechanisms that optimize/approximate the makespan. When the entries of the matrix t are unrelated, the domain of input 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 [15], 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 [5] for 3 machines, and to 2.618 [9] for many machines. In [15], it was also shown that applying the VCG mechanism [17, 3, 8] 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. [2], 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 by a probability distribution of truthful mechanisms, while it is *truthful in expectation*, if in expectation no player can benefit by lying. Already in [15], a universally truthful mechanism was suggested, for two machines. The mechanism was extended for the case of m machines by Mu'alem and Schapira [14] with an approximation guarantee of $0.875m$, and this was further improved in [12] to $0.837m$. Lu and Yu [13] showed a truthful-in-expectation mechanism with a guarantee of $(m + 5)/2$. In [14] it was also shown a lower bound of $2 - 1/m$, for both randomized versions, while in [4] the lower bound was extended for fractional mechanisms, and an upper bound of $(m + 1)/2$ was provided. Surprisingly, even for the special case of two machines a tight answer has not been given for randomized mechanisms. Currently the lower bound is 1.5 [14], while the best upper bound is 1.5963 due to [13].

Setting restrictions to the input domain can make the problem easier. The single-dimensional counterpart of the problem is the scheduling on *related* machines. In that case it is assumed that machine i has speed s_i, then $t_{ik} = w_k/s_i$, where the weights w_k of the jobs are known to the designer. Notice that the only missing information is the speed of the machines. In that case, the constraints imposed by truthfulness seem harmless; the optimal allocation is truthfully implementable [1], although it takes exponential running time, while the best possible approximation guarantee, a PTAS, can be achieved by polynomial time truthful mechanisms [7, 6]. 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 [10], is to restrict the input domain, but still keep the multi-dimensional flavour. They assumed that each entry 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. Yu [18] extended this study, for more than two values, but where the inputs are restricted in balls around two values.

1.1 Our contribution

The focus of this work is, following [10] to restrict even further the domain. We still allow only two possible values L, H, that are publicly known, but we even restrict the way these values are placed in a player i's input vector. We assume that for each machine some known partition of the tasks into two parts is given to the designer. The only missing information is in which part player assigns low values, and in which part it assigns the high ones. Therefore, the only missing information is a *single bit* for each player[3]. The lower bound given in [10] 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:

(1) Power of truthful-in-expectation mechanisms. There is a class of two-values scheduling problems for which every algorithm (thus including optimal ones) can be turned into a truthful-in-expectation mechanism with the same approximation guarantee (Theorem 3). On the contrary, randomized universally truthful mechanisms cannot achieve an approximation better than 31/30 (Theorem 17), and the 11/10 lower bound for deterministic mechanisms in [10] also applies.

Notice that such a separation was not known for the general problem since, although Lu [11], 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.

(2) Two-values vs three-values domains. For two machines, three-values domains are as difficult as the general unrelated machines: the lower bound of 2 for deterministic mechanisms still hold (Theorem 18). We give a *partial* evidence of the fact that two-values domains are easier by giving a deterministic truthful 3/2-approximation mechanisms for the subcase of given partitions (the most general domain we consider in this work – see Section 1.2 – still a restriction of the two-values domains).

Due to space limitations some of the proofs are omitted. We refer the reader to the full version of this work.

1.2 Preliminaries

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

[3] 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 [16] for the precise definition of a single-dimensional domain.

vector of processing times or *type* $t_i = (t_{ih})_h$, where $t_{ih} \in \Re_{\geq 0}$ is i's processing time for job h. In the **two-values** domains by Lavi and Swamy [10], 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). 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

$$t_{ih} = L_S^h := \begin{cases} L & \text{if } h \in S \\ H & \text{otherwise} \end{cases} \tag{1}$$

and similarly for H_S-machines.

In this work, we consider the following special case of the two-values domains [10]. In the case with **given partitions**, 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 must belong to a simple domain of two elements only (i.e., $t_i \in \{L_{S_i}, H_{S_i}\}$). 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 $\bar{S}_i = [n] \setminus S_i$). We shall further distinguish between three restrictions (of increasing difficulty) of the given partitions domain: (i) **Identical partitions**, where all subsets S_i are identical; (ii) **Uniform partitions**, where all subsets S_i have size s for some $s \geq 0$; and (iii) **(Unrestricted) Given 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 machine i. The *makespan* of x with respect to t is $\max_i C_i(x, t)$. An *exact* or *optimal* allocation x 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 randomized algorithms, $A(t)$ is a probability distribution over all possible allocations; we call $A(t)$ a randomized allocation.

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 $(t_1, \ldots, t_{i-1}, \hat{t}_i, t_{i+1}, \ldots, t_m)$. 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: $n_L^i(x, t) := |\{h : t_{ih} = L \text{ and } x_{ih} = 1\}|$ and $n_H^i(x, t) := |\{h : t_{ih} = H \text{ and } x_{ih} = 1\}|$.

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

$$n_L^i - n_H^i + \hat{n}_L^i - \hat{n}_H^i \geq 0 \tag{2}$$

where $n^i_L = n^i_L(A(t), t)$, $n^i_H = n^i_H(A(t), t)$, $\hat{n}^i_L = n^i_L(A(\hat{t}), \hat{t})$, *and* $\hat{n}^i_L = n^i_L(A(\hat{t}), \hat{t})$.

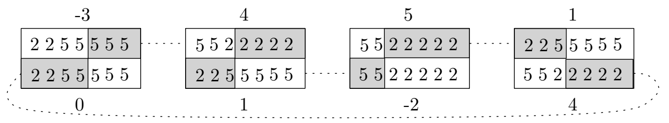
By applying [10, Proposition 5.7] to our given partitions domains, we obtain that truthfulness is equivalent to the monotonicity condition above:

Theorem 2. *For the case of identical/uniform/unrestricted given partitions, there exist prices P such that the mechanism (A, P) is truthful (in expectation) iff A is monotone (in expectation).*

Throughout the paper, we refer to the quantity $n^i_L(x,t) - n^i_H(x,t)$ as the *unbalance* of machine i. We also refer to the quantity in (2) 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 α on machine i and β on machine j as $J^{ij}_{\alpha\beta}(t) := \{h : t_{ih} = \alpha \text{ and } t_{jh} = \beta\}$.

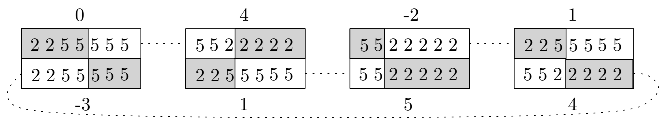
1.3 An illustrative example

We begin with an example and show how to use randomization. Consider the following instances along with their optimal allocation (gray box), and the quantities $n^i_L - n^i_H$ for each of the two machines (numbers outside the box):



(3)

Let machine 1 and machine 2 correspond to top and bottom machine, respectively. Observe that the monotonicity condition is violated for machine $i = 2$ by looking at the two middle instances (they differ in the machine connected by dotted line). Indeed the quantity in (2) is $1 - 2 = -1$. Alternatively, we can swap the allocation in the first and in the third input:



(4)

Now, however, the monotonicity condition is violated by machine $i = 1$ for the last two instances. These instances are used by Lavi and Swamy [10] to prove a lower bound for deterministic mechanisms. However, if we choose *randomly* between the allocation in (3) and the one in (4) with the same probability, the corresponding optimal algorithm satisfies monotonicity *in expectation* (for example, in the leftmost instance the unbalance becomes $-3/2$ for both machines, while in the second instance it remains unchanged).

Our simple example shows that for two machines with identical partitions as above, and values $L = 2$ and $H = 5$, there exists an exact truthful-in-expectation mechanism but no truthful mechanism can achieve an approximation factor better than 1.1 [10]. 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 algorithms into mechanisms that are truthful-in-expectation. The main result of this section is summarized by the following theorem.

Theorem 3. *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 not worse 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 $S = \{1, 2, \ldots, s\}$ and $\bar{S} = \{s+1, s+2, \ldots, n\}$. 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: $L_S = (L \cdots LH \cdots\cdots H)$ 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 numbers of L_S-machines and H_S-machines in t, respectively. Clearly, $m_S(t) + m_{\bar{S}}(t) = m$. It is convenient to count, for each side of the partition, the number of jobs that x allocates as L-jobs:

$$\ell_S(x, t) = |h \in S : x_{ih} = 1 \text{ and } t_{ih} = L| \tag{5}$$

The quantity $\ell_{\bar{S}}(x, t)$ is defined similarly, and $\ell(x, t) := \ell_S(x, t) + \ell_{\bar{S}}(x, t)$ is the overall number of allocated 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 4. *For any deterministic allocation x we denote by $x^{(rand)}$ the randomized allocation obtained as follows:*

- *Pick an integer $r \in \{0, \ldots, m_S - 1\}$ uniformly at random, and set $x_i^{(rand)} := x_{i+r \mod m_S}$ for each L_S-machine i;*
- *Pick an integer $\bar{r} \in \{0, \ldots, m_{\bar{S}} - 1\}$ uniformly at random, and set $x_i^{(rand)} := x_{i+\bar{r} \mod m_{\bar{S}}}$ 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 $\ell_S(x^{(rand)}, t) = \ell_S(x, t)$ and $\ell_{\bar{S}}(x^{(rand)}, t) = \ell_{\bar{S}}(x, t)$. In the following discussion we fix x and t and simply write ℓ_S and $\ell_{\bar{S}}$.

For any L_S-machine i, its expected load consists of $n_L^i = \ell_S / m_S$ L-jobs and $n_H^i = (n - s - \ell_{\bar{S}})/m_S$ H jobs. Thus the expected unbalance of an L_S-machine is $n_L^i - n_H^i = \frac{1}{m_S}[\ell_S - (n - s - \ell_{\bar{S}})] = \frac{1}{m_S}[\ell - (n - s)]$. Similarly, the expected load of an H_S-machine i consists of $n_L^i = \ell_{\bar{S}}/m_{\bar{S}}$ L-jobs and $n_H^j = (s - \ell_S)/m_{\bar{S}}$ H-jobs and its expected unbalance is equal to $n_L^i - n_H^i = \frac{1}{m_{\bar{S}}}[\ell_{\bar{S}} - (s - \ell_S)] = \frac{1}{m_{\bar{S}}}(\ell - s)$.

Lemma 5. *Algorithm $A^{(rand)}$ is monotone-in-expectation if the deterministic algorithm A satisfies the following condition. For any $t = (L_S, t_{-i})$ and $\hat{t} = (H_S, t_{-i})$, it holds that*

$$\frac{\ell - (n-s)}{m_S} + \frac{\hat{\ell} - s}{m_{\bar{S}} + 1} \geq 0 \tag{6}$$

where $\ell = \ell(A(t), t)$ and $\hat{\ell} = \ell(A(\hat{t}), \hat{t})$ denote the number of L-jobs allocated 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 the total unbalance of machine i is $n^i_L - n^i_H + \hat{n}^i_L - \hat{n}^i_H = \frac{\ell-(n-s)}{m_S} + \frac{\hat{\ell}-s}{\hat{m}_{\bar{S}}}$, 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 (6) is equivalent to (2) and thus $A^{(rand)}$ is monotone-in-expectation.

2.1 Canonical allocations

Lemma 5 says that in order to design an exact truthful-in-expectation mechanism it is enough to design a deterministic exact algorithm A which obeys the condition in (6). We show that this is always possible by transforming the allocation of the algorithm into a "canonical" allocation (specified below).

Definition 6. *Given an allocation x and an instance t, for any $\alpha, \beta \in \{L, H\}$ and for any two machines i and j, we let $n^{ij}_{\alpha\beta}(x,t)$ be the number of α-jobs that are allocated to machine i and that are β-jobs for machine j: $n^{ij}_{\alpha\beta}(x,t) := |\{k : x_{ik} = 1,\ t_{ik} = \alpha, \quad \text{and } t_{jk} = \beta\}|$.*

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

Definition 7 (canonical allocation). *A canonical allocation (for the instance t) is an allocation obtained by modifying a deterministic allocation x as follows:*

1. *Apply the following **Rule R1** until 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 $t_{jh} < t_{jk}$ (no machine gets worse and at least one gets better if we swap the jobs), 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** until possible: If $n^{ij}_{HL}(x,t) > n^{ji}_{LH}(x,t)$ and j gets only jobs from $J^{ji}_{LH}(t)$, then move all n^{ij}_{HL} jobs in $J^{ij}_{HL}(t)$ from i to j, and move all jobs from $J^{ji}_{LH}(t)$ from j to i (see Figure 1.).*

Remark 8. Both Rule 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 (following the two steps in Definition 7) whose cost is not larger than the cost of x.

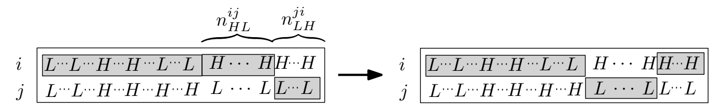

n_{HL}^{ij}
n_{LH}^{ji}
i
j
$L \cdots L \cdots H \cdots H \cdots L \cdots L$
$H \cdots H$
$H \cdots H$
$L \cdots L \cdots H \cdots H \cdots H \cdots H$
$L \cdots L$
$L \cdots L$

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

2.2 A black-box construction (proof of Theorem 3)

Lemma 9. *Every deterministic algorithm A can be turned into a randomized algorithm $A^{(rand)}$ which is monotone in expectation and whose output allocation has makespan not worse than the one returned by A.*

Proof (Proof Idea). The proof is based on the following structural properties of canonical allocations. Either $\ell_S(x,t) = |S|$ or $\ell_{\bar{S}}(x,t) = |\bar{S}|$. Moreover these bounds also hold:

$$\ell_S(x,t) = |S| \text{ implies } \ell_{\bar{S}}(x,t) \geq \frac{m_{\bar{S}}}{m} \cdot |\bar{S}|$$
$$\ell_{\bar{S}}(x,t) = |\bar{S}| \text{ implies } \ell_S(x,t) \geq \frac{m_S}{m} \cdot |S|$$

To apply Lemma 5 we need to prove that, for any $t = (L_S, t_{-i})$ and $\hat{t} = (H_S, t_{-i})$

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

We distinguish these four possible cases: (1) $\ell_S = |S|$ and $\hat{\ell}_{\bar{S}} = |\bar{S}|$, (2) $\ell_{\bar{S}} = |\bar{S}|$ and $\hat{\ell}_S = |S|$, (3) $\ell_S = |S|$ and $\hat{\ell}_S = |S|$, (4) $\ell_{\bar{S}} = |\bar{S}|$ and $\hat{\ell}_{\bar{S}} = |\bar{S}|$. Cases (1)-(2) use the fact that $\ell + \hat{\ell} \geq n$. Cases (3)-(4) use the bounds above and $\hat{m}_{\bar{S}} = m_{\bar{S}} + 1$.

3 Mechanisms for two machines

In this section we restrict our attention to the case of $m = 2$ machines. We give an exact truthful-in-expectation mechanism for the case of uniform partitions (Sect. 3.1) and a deterministic 3/2-approximation mechanism for the case of unrestricted given partitions (Sect. 3.2).

3.1 Exact randomized mechanisms for uniform partitions

We can always assume that any deterministic algorithm can produce canonical allocations (see Remark 8). We show that every exact algorithm can be turned into a monotone-in-expectation algorithm (thus an exact truthful-in-expectation mechanism). Unlike the case of identical partitions, we apply a randomization step only in some cases. Specifically, we will randomize if the two machines have types $t = (L_{S_1}, L_{S_2})$ or $t' = (H_{S_1}, H_{S_2})$, and give in output directly the allocation of the deterministic algorithm in the other two cases, inputs (L_{S_1}, H_{S_2}) and (H_{S_1}, L_{S_2}).

We start by characterizing exact canonical allocations.

Definition 10 (allocation classes). *An allocation x is classified with respect to the instance t as $J^{ij}_{LH}(t)$ if machine i gets only a proper subset of the jobs in $J^{ij}_{LH}(t)$; x is symmetric if each machine i gets all jobs in $J^{ij}_{LH}(t)$, no job in $J^{ij}_{HL}(t)$, and a (possibly empty) subset of the jobs in $J^{ij}_{LL}(t) \cup J^{ij}_{HH}(t)$.*

Lemma 11. *Every instance t admits an exact canonical allocation that is either symmetric or of class $J^{ij}_{LH}(t)$, with $i \in \{1, 2\}$.*

We next show that *certain* symmetric allocations can be converted into randomized allocations having a "good" unbalance, without increasing their cost.

Definition 12 (randomizable allocation). *A symmetric allocation x is* randomizable *if the following holds. Let i be the machine such that $|J^{ij}_{LH}(t)| \geq |J^{ij}_{HL}(t)|$.*[4] *Then, one of the following two conditions holds: (1) $n^{ji}_{LL}(x, t) \geq |J^{ij}_{LH}(t)| - |J^{ij}_{HL}(t)|$, or (2) $n^{ji}_{HH}(x, t) \leq n^{ij}_{HH}(x, t)$.*

Lemma 13. *For every deterministic allocation x that is randomizable with respect to the instance t, there exists a randomized allocation $x^{(rand)}$ which gives an expected unbalance of $\frac{n}{2} - |J_{HH}(t)|$ to both machines and has the same makespan as the makespan of x.*

Proof (Sketch). Let x be an allocation that is randomizable with respect to t. We build a new allocation y as follows. Let i be a machine such that $|J^{ij}_{LH}(t)| \geq |J^{ij}_{HL}(t)|$ and let $\delta := |J^{ij}_{LH}(t)| - |J^{ij}_{HL}(t)|$. The new allocation is obtained by swapping all jobs in $J^{ij}_{HH}(t)$ and *some* of the jobs in $J^{ij}_{LL}(t)$ according to the following two cases:

1. ($n^{ji}_{LL}(x, t) \geq \delta$) In this case we move $n^{ji}_{LL}(x, t) - \delta$ jobs from machine j to machine i, and $n^{ij}_{LL}(x, t)$ jobs from machine i to machine j.
2. ($n^{ji}_{LL}(x, t) < \delta$) In this case we move $n^{ij}_{LL}(x, t) + \delta - n^{ji}_{LL}(x, t)$ jobs from machine i to machine j (this quantity is nonnegative by Definition 12).

Finally, build the randomized allocation $x^{(rand)}$ by picking at random x or y.

The above lemma and its randomization procedure guarantees that the monotonicity condition can be satisfied whatever are the canonical allocations that we *do not randomize*, inputs (L_{S_1}, H_{S_2}) and (H_{S_1}, L_{S_2}). Moreover, for the case of uniform partitions, it turns out that the exact allocations of instances $t = (L_{S_1}, L_{S_2})$ and $t' = (H_{S_1}, H_{S_2})$ are both randomizable. Thus the following holds:

Theorem 14. *There exists an exact truthful-in-expectation mechanism for the case of two machines with uniform partitions.*

3.2 A deterministic 3/2-approximation mechanism

In this section we prove the following result:

Theorem 15. *There exists a 3/2-approximation deterministic truthful mechanism for two machines with (unrestricted) given partitions.*

In order to prove this result we exhibit a monotone 3/2-approximation algorithm.

[4] Recall that by definition $J^{ij}_{LH}(t) = J^{ji}_{HL}(t)$ and thus such machine must exist.

The algorithm. On input t, the algorithm partitions the jobs into three subsets: $J_{LL}(t)$, $J_{HH}(t)$ and $J_{LHHL}(t) := J_{LH}^{12}(t) \cup J_{HL}^{12}(t)$.

First, 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" must 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 for *all* jobs:
   (a) **(class J_{LH}^{12} or J_{LH}^{21}).** 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 $C_i^{(LHHL)}$ be the corresponding cost.
2. **Step 2 (allocate jobs in J_{LL} and J_{HH})** For a set of jobs S and a nonnegative integer q, we denote by $\lfloor S/q \rfloor$ and $\lceil S/q \rceil$ an arbitrary subset of S of cardinality $\lfloor |S|/q \rfloor$ and $\lceil |S|/q \rceil$, 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 that is equal to $J_{LHHL}^{i} \cup \left\lfloor \frac{J_{HH}}{2} \right\rfloor \cup \left\lceil \frac{J_{LL}}{2} \right\rceil$, if both $|J_{LL}|$ and $|J_{HH}|$ are odd, and $J_{LHHL}^{i} \cup \left\lfloor \frac{J_{HH}}{2} \right\rfloor \cup \left\lfloor \frac{J_{LL}}{2} \right\rfloor$ otherwise. Machine j gets all the other jobs $J^j := \bar{J}^i = [n] \setminus J^i$.

Approximation guarantee. The proof of the approximation guarantee is based on the following lemma that proves that the two steps of the algorithm keep a small difference between the completion time of the two machines:

Lemma 16. *After each step, the difference between the two completion times is at most the optimum, that is* $\max_i C_i^{(LHHL)} \leq OPT$ *and* $\max_i C_i \leq \min_i C_i + OPT$.

To prove the 3/2-approximation we have to distinguish two cases, depending on which class of canonical allocation has been used in Step 1 to allocate jobs in J_{LHHL}. We observe that in the first case (class J_{LH}^{12} or J_{LH}^{21}) at least one of these jobs is allocated as an H-job; in the second case, instead, all jobs in J_{LHHL} are allocated as L-jobs and the remaining jobs are distributed among the two machines in order to minimize the makespan. In both the cases, using Lemma 16 we can prove that $APX \leq (3/2) \cdot OPT$.

It is easy to verify that the mechanism can indeed reach a 3/2 approximation guarantee.

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,

$n_{LH}^{ij} \geq \left\lceil \frac{J_{LH}^{ij}}{2} \right\rceil$, $n_{LL}^{ij} \geq \left\lfloor \frac{J_{LL}^{ij}}{2} \right\rfloor$ 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 7 can be applied. Therefore we obtain

$$n_L^i \geq \left\lceil \frac{J_{LH}^{ij}}{2} \right\rceil + \left\lfloor \frac{J_{LL}^{ij}}{2} \right\rfloor, \qquad n_H^i \leq \left\lceil \frac{J_{HH}^{ij}}{2} \right\rceil + \left\lfloor \frac{J_{HL}^{ij}}{2} \right\rfloor. \tag{7}$$

Second, observe that if the type of machine i flips from t_i to $\hat{t}_i$, then $|\hat{J}_{\bar{\alpha}\beta}^{ij}| = |J_{\alpha\beta}^{ij}|$ (here $\bar{L} = H$ and $\bar{H} = L$). Applying (7) for both $t_i, \hat{t}_i$ and using the last identity, we finally obtain (2).

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 3)

Theorem 17. *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.*

The proof combines the idea of the lower bound by Lavi and Swamy [10] with the use of Yao's Min-Max Principle suggested by Mu'alem and Schapira [14]. The above bounds can be strengthen by considering three-values domains (which are no-longer "single-bit").

Theorem 18. *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. *Moreover, no randomized universally truthful mechanism can achieve and approximation factor better than* $9/8$.

The first part is an alternative proof for the lower bound of 2 for two machines, first showed by Nisan and Ronen in [15]. The proof in [15] 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 only 3 different values. The second part adapts this proof via Yao's Min-Max Principle.

5 Conclusion

This work leaves several open questions. First of all, we are able to derive mechanisms for an arbitrary number of machines only in one case (identical partitions). Second, exact truthful-in-expectation mechanisms for two machines are given only for uniform partitions. Is it 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 given partitions.

References

1. Archer. A., Tardos, E.: Truthful Mechanisms for One-Parameter Agents, in Proc. 42nd FOCS, 2001, 482 –491
2. Ashlagi, I., Dobzinski, S., Lavi, R.: An Optimal Lower Bound for Anonymous Scheduling Mechanisms, in Proc. 10th EC, 2009, 169 –176
3. Clarke, E.H.: Multipart pricing of public goods, Public Choice, vol. 11 n. 1, 1971, 17 –33
4. Christodoulou, G., Koutsoupias, E., Kovács A.: Mechanism design for fractional scheduling on unrelated machines, ACM Trans. on Algorithms, vol. 6 n. 2, 2010
5. Christodoulou G., Koutsoupias E., Vidali, A.: A Lower Bound for Scheduling Mechanisms, Algorithmica, vol. 55 n. 4, 2009, 729 –740
6. Christodoulou, G., Kovács, A.: A deterministic truthful PTAS for scheduling related machines, in Proc. of 21st SODA, 2010, 1005 –1016
7. Dhangwatnotai, P., Dobzinski, S., Dughmi, S., Roughgarden, T.: Truthful Approximation Schemes for Single-Parameter Agents, SIAM J. on Computing, vol. 40 n. 3, 2011, 915 –933
8. Groves, T.: Incentives in teams, Econometrica, vol. 41 n. 4, 1973, 617 –631
9. Koutsoupias, E., Vidali, A.: A Lower Bound of $1+\phi$ for Truthful Scheduling Mechanisms, in Proc. MFCS, 2007, 454 –464
10. Lavi, R., Swamy, C.: Truthful mechanism design for multidimensional scheduling via cycle monotonicity, Games and Economic Behavior, vol. 67 n. 1, 2009, 99 –124
11. Lu, P.: On 2-Player Randomized Mechanisms for Scheduling, in Proc. 5th WINE, 2009, 30 –41
12. Lu, P., Yu, C.: An Improved Randomized Truthful Mechanism for Scheduling Unrelated Machines, in Proc. of 25th STACS, 2008, LIPIcs vol. 1, 527 –538
13. Lu, P., Yu, C.: Randomized Truthful Mechanisms for Scheduling Unrelated Machines, in Proc. of 4th WINE, 2008, 402 –413
14. Mu'alem, A., Schapira, M.: Setting Lower Bounds on Truthfulness, in Proc. of 18th SODA, 2007, 1143 –1152
15. Nisan, N., Ronen, A.: Algorithmic Mechanism Design, Games and Economic Behavior, vol. 35, 2001, 166 –196
16. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.: Algorithmic Game Theory, Cambridge University Press, 2007
17. Vickrey, V.: Counterspeculations, auctions and competitive sealed tenders, Journal of Finance, vol. 16, 1961, 8 –37
18. Yu, C.: Truthful mechanisms for two-range-values variant of unrelated scheduling, Theoretical Computer Science, vol. 410 n. 21 –23, 2009, 2196 –2206