Mechanism design for scheduling

George Christodoulou
Max-Planck-Institut für Informatik,
Saarbrücken, Germany.
gchristo@mpi-inf.mpg.de

Elias Koutsoupias*
Department of Informatics
University of Athens, Greece
elias@di.uoa.gr

Abstract

We consider mechanism design issues for scheduling problems and we survey some recent developments on this important problem in Algorithmic Game Theory. We treat both the related and the unrelated version of the problem.

1 The scheduling problem

The problem of scheduling unrelated machines [21, 14] is one of the most fundamental algorithmic problems: There are n machines and m tasks* and machine i can execute task j in time t_{ij}. These times can be totally unrelated (thus the name of the problem). The objective is to allocate the tasks to machines to minimize the makespan (the time needed to finish all tasks). Thus the output is simply a partition of the m tasks into n sets. A convenient way to express it is to use indicator variables $x_{ij} \in \{0, 1\}$: x_{ij} is 1 iff task j is allocated to machine i. Each task j is allocated to exactly one machine, therefore we must have $\sum_{i=1}^{n} x_{ij} = 1$ for every j. With this notation, the computational problem can be expressed more precisely: given $n \times m$ values t_{ij}, find appropriate $x_{ij} \in \{0, 1\}$ which satisfy these constraints and minimize $\max_{i=1}^{m} \sum_j x_{ij}t_{ij}$.

From the traditional algorithmic point of view, the unrelated machines scheduling problem is one of the most important open problems. We know that the problem is NP-hard; it is even NP-hard to approximate it within 3/2 [21]; this lower bound applies also to some special cases [11]. On the positive side, there is a polynomial-time approximation algorithm with approximation ratio 2 [21].

*Partially supported by IST-15964 (AEOLUS) and IST-2008-215270 (FRONTS).

*We opt for the game-theoretic notation here and we denote the number of machines and tasks with n and m respectively. In the scheduling literature, they usually use the opposite notation.

Closing the gap between the lower and upper bounds on the approximation ratio remains a long-standing major algorithmic problem.

There are many interesting variants of the problem. When, for example, the times t_{ij} are inversely proportional to the speed of the machine, that is, when there are speeds s_i and times p_j such that $t_{ij} = p_j/s_i$, we have the special case of the problem called the related machines scheduling problem. Also, when we allow a task to be split across the machines, which is to say that x_{ij} are nonnegative reals instead of integers, we call this the fractional scheduling problem. The computational complexity of these problems is completely settled: There is a polynomial time approximation scheme (PTAS) [13] for the related machines problem and a fully-PTAS (FPTAS) [15] when the number of machines is fixed; the general case is strongly NP-complete, so we don't expect to find an FPTAS unless P=NP. For the fractional version of the problem, there is a polynomial-time algorithm (because it can be expressed as a linear program).

Nisan and Ronen in their seminal work [27, 28] which started the area of Algorithmic Mechanism Design considered the unrelated machines problem from a game-theoretic point of view: suppose that each machine i is a rational agent who is the only one knowing the values of row t_i. Suppose further that the machines want to minimize their execution time. Without any incentive, the machines will lie in order to avoid getting any task. To coerce the machines to cooperate, we pay them to execute the tasks. The payments do not have to be proportional to the execution times, but can be arbitrary functions. The combination of the algorithmic problem of allocating the tasks to machines together with the incentives in the form of payments is called a mechanism. In this article, we survey recent developments in this area of mechanisms for the scheduling problem.

We consider direct revelation mechanisms with dominant truthful strategies. Direct revelation means that the players—who know the mechanism in advance—declare their hidden values to the mechanism which collects the values and computes an allocation of tasks and appropriate payments to the players. In such a mechanism, a player may have an incentive to lie and declare values other than his true values. If the mechanism is such that, independently of the values of the other players, a player has no incentive to lie, we say that the mechanism is truthful (with dominant truthful strategies). These mechanisms are very desirable and easy to be implemented since there is no reason for machines to strategize. There other weaker notions of truthfulness but we don't consider them in this note.

2 Mechanism design

The mechanism design setting for scheduling is a special case of the social choice problem. We define here the more general framework so that we can place the scheduling problem within the general picture. In the social choice setting, there are n players and a set of possible outcomes A which in most cases is considered finite. The players may value the outcomes differently. For each player there is a valuation function $v_i : A \to R$ which gives the value of player i for every outcome. The goal of the mechanism designer is to implement a social choice function f which assigns a desirable outcome to every set of valuation functions. For example, a social choice function selects the outcome for which the median of the valuations of the players is maximum. More generally, for every set of valuation functions there may be a set of desirable outcomes and the mechanism designer wants to implement a social choice function which selects one of these outcomes.

When we consider problems with finitely many possible outcomes, we can recast the above in a more familiar notation. A mechanism design problem with n players and $k = |A|$ outcomes is defined by a subset D of $R^{n\times k}$. We call D the domain of the mechanism design problem. A social choice function f is simply a function from D to $\{1, \dots, k\}$ (or more generally to the collection of subsets of $\{1, \dots, k\}$).

An instance of a mechanism design problem is simply a point of D. A concrete way to represent a point of D is by a real-valued $n \times k$ matrix v. Each player i knows the values of row $\hat{v}_i$; this is private information and it is not known to the mechanism. In a direct-revelation mechanism, each player i declares values v_i of row i. These values may be different than $\hat{v}_i$. The declarations of the players form a matrix v. The mechanism takes as input this matrix v and computes two quantities: an outcome $o = o(v) \in \{1, \dots, k\}$ (the outcome of the mechanism) and payments $p = p(v) \in R^n$ for the players. The payoff of player i is the value of the original row at the outcome minus the payment: $\hat{v}_{i,o} - p_i$.

In summary, a mechanism design problem is defined by a set D of $n \times k$ real matrices (a subset of $R^{n\times k}$) and a function f from D to subsets of $\{1, \dots, k\}$. The mechanism designer must come up with an outcome function

$$o : D \to \{1, \dots, k\},$$

and a payment scheme

$$p : D \to R^n,$$

such that the outcome o is a desirable outcome ($o(v) \in f(\hat{v})$) and the payment scheme induces the players to declare values which produce desirable outcomes.

The mechanism is *truthful* when the outcome and payment functions are such that the players gain nothing by not declaring their true values, i. e., the objective of each player i, for every declared values v_{-i} of the other players, is maximized when player i declares $v_i = \hat{v}_i$. This notion of truthfulness is called dominant-strategy truthfulness since declaring the true values is a dominant strategy for each player.

The *Revelation Principle* states that for every mechanism there is an equivalent truthful mechanism which on the same input $\hat{v}$ has the same outcome and payments. This frees us to consider only truthful mechanisms.

Here are a few typical mechanism design domains:

Example (The unrestricted domain): One of the most natural mechanism design problems for n players and k outcomes is when D is the whole $R^{n\times k}$ space.

Example (The combinatorial auction domain): The outcomes are all allocations of m items to n players; there are $k = n^m$ possible allocations. The domain is defined by values $v_{i,x}$, one for each player i and for each allocation x. An allocation x is defined by a matrix where x_{ij} is a variable indicating whether task j is allocated to player i. The values $v_{i,x}$ satisfy the natural restriction that the valuation of a player depends only on the items allocated to him: $v_{i,x} = v_{i,x'}$ when the two allocations agree on player i (that is, when $x_i = x'_i$). In economic terms this condition says that there are no externalities. Another natural restriction is that the value of a player can only increase when he gets additional items ($v_{i,x} \geq v_{i,x'}$ when $x_{ij} \geq x'_{ij}$ for $j = 1, \ldots, m$). The values are also nonnegative and they are exactly 0 when a player is allocated no item.

For example, for $n = 3$ players and $m = 2$ items the domain D contains the points of $R^{3\times 9}$ which are of the form:

$$\begin{array}{|ccccccccc|} u_{1,12} & u_{1,1} & u_{1,1} & u_{1,2} & u_{1,2} & 0 & 0 & 0 & 0 \\ 0 & u_{2,2} & 0 & u_{2,1} & 0 & u_{2,12} & u_{2,1} & u_{2,2} & 0 \\ 0 & 0 & u_{3,2} & 0 & u_{3,1} & 0 & u_{3,2} & u_{3,1} & u_{3,12} \end{array},$$

where the values are nonnegative and $u_{i,12} \geq u_{i,1}$ and $u_{i,12} \geq u_{i,2}$.

For the special case of the single-item auction, the matrix is a diagonal one where the value $u_{i,i}$ is the valuation of player i for the item.

Example (The unrelated machines domain): This is a special case of the combinatorial auction when the domain is additive. It is also a cost game (as opposed to a payoff one). The outcomes again are all allocations of m tasks to n players ($k = n^m$). The domain is defined by

$v_{i,x} = \sum_{j=1}^{m} x_{ij} t_{ij}$. For example, for $n = 2$ players and $m = 2$ tasks the domain D contains the points of $R^{2\times 4}$ for which are of the form:

$t_{11} + t_{12}$	t_{11}	t_{12}	0
0	t_{22}	t_{21}	$t_{21} + t_{22}$

where t_{ij}'s are nonnegative.

2.1 Known mechanisms

Given a domain, one can ask whether there are any (truthful) mechanisms. If we view the payments as the means to implement a social choice function, we can rephrase the question: For which social choice functions are there payment functions so that the resulting mechanism is truthful? In this way we focus on the social choice function. For example, for the single-item auction domain, are there payment functions for a mechanism to allocate the item to the player with maximum (private) value? with the second maximum value? As we will discuss soon, the answer to the first question is positive and to the second question negative.

There are few mechanisms that are known to be truthful and the best-studied one is the VCG mechanism [31, 8, 12].

Example (The VCG mechanism and affine maximizers): The VCG mechanism implements the social choice function of selecting the outcome (column) with the maximum total value:

$$f(v) = \operatorname*{argmax}_{j\in\{1,\ldots,k\}} \sum_{i=1}^{n} v_{ij}.$$

A generalization of this mechanism is the affine maximizer which weights with positive multipliers λ_i the values of each player (row) i and add a constant γ_j to the value of each outcome (column) j:

$$f(v) = \operatorname*{argmax}_{j\in\{1,\ldots,k\}} \sum_{i=1}^{n} \lambda_i v_{ij} + \gamma_j.$$

The VCG mechanism is truthful for every domain. The payments (for the general domain) align the objective of each player l with the social choice function. This can be achieved when the payments are

$$p_l(v) = -\frac{1}{\lambda_l}\left(\sum_{i\neq l} \lambda_i v_{ij} + \gamma_j\right).$$

Player l wants the mechanism to select an outcome j which maximizes $v_{lj}-p_l(v) = \frac{1}{\lambda_l}\sum_{i=1}^{n}\lambda_i v_{ij} + \gamma_j$. This is the same expression with the argmax expression above and shows that the player's objective is achieved at the social choice function. The VCG has slightly different payments: Because these payments may be negative, the VCG mechanism adds appropriate values to the payment of each player that depend only on the values of the other players (this keeps the player truthful).

Another interesting class of mechanisms for the scheduling problem are the task independent mechanisms: Each task is allocated independently of the remaining tasks. Not all task-independent mechanisms are truthful. Task-independent mechanisms are special cases of threshold mechanisms:

> **Example** (Threshold mechanisms): A threshold mechanism for the scheduling domain is one for which there are threshold functions h_{ij} such that the mechanism allocates item j to player i if and only if $v_{ij} \geq h_{ij}(v_{-i})$. What distinguishes these mechanisms from general mechanisms is that the thresholds depend only on the values of the other players but not on the other values of the player himself. It is not true in general that every set of functions h_{ij} defines a legal mechanism, as they have to be consistent between them. In particular, the threshold functions should be such that every item j is allocated to exactly one player. In other words, exactly one of the constraints $v_{ij} \geq h_{ij}(v_{-i})$, for $i = 1, \ldots, n$, should be satisfied.

3 Truthfulness

One of the central questions in mechanism design is to find a nice characterization of truthful mechanisms. In algorithmic terms, we want to determine which algorithms are implementable, i.e., for which algorithms for the scheduling problem there exist payments that make the players truthful. It should be clear that for rich domains, such as the scheduling domain, not all algorithms are truthful. In fact, it seems that the set of truthful algorithms is very limited, but whether this is the case or not is perhaps the most outstanding open problem in algorithmic mechanism design:

Open Problem. *Characterize the set of truthful mechanisms for scheduling.*

But what kind of characterization we seek? We are going to see that we do have a necessary and sufficient condition, the so-called Monotonicity Property. But we want a characterization which is more than a necessary and sufficient condition. An important result in the area of mechanism design, Roberts' Theorem

[16], shows exactly the type of characterization we seek. Roberts' Theorem applies to the unrestricted domain and states that the only truthful mechanisms for this domain of $k \geq 3$ outcomes are the affine maximizers. In a sense, this is a very disappointing result, because it says that only very simple algorithms can be implemented. The question becomes much more interesting for restricted domains and in particular for the auction and scheduling domain. It is a simple observation that when we restrict the domain the set of available mechanisms can only become richer. More precisely, for domains $D \subset D'$, every truthful mechanism for D' is also a truthful mechanism for D.

We discuss below the Monotonicity Property which a simple necessary and sufficient condition for truthfulness. This is true for every convex domain, but we restrict the discussion to the scheduling domain.

Definition 3.1 (Monotonicity Property)**.** *An allocation algorithm is called monotone if it satisfies the following property: for every two sets of tasks t and t' which differ only on machine i (i.e., on the i-the row) the associated allocations x and x' satisfy*

$$(x_i - x'_i) \cdot (t_i - t'_i) \leq 0,$$

where $\cdot$ denotes the dot product of the vectors, that is, $\sum_{j=1}^{m}(x_{ij} - x'_{ij})(t_{ij} - t'_{ij}) \leq 0$.

The property, which sometimes in the literature is called weak monotonicity, essentially states that when we increase the times of the tasks for machine i, the allocation for the machine can only become smaller. Notice that the Monotonicity Property involves only the allocation of one player (the i-th player).

Theorem 3.2 (Saks and Yu)**.** *A mechanism is truthful if and only if its allocations satisfy the Monotonicity Property.*

To establish that the Monotonicity Property is necessary for truthfulness is easy (it was done for example in [28]) and we show it below. Saks and Yu showed that it is also a sufficient condition. In fact, they showed a much more general result: the property is sufficient for every convex domain; this includes the unrestricted domain and the combinatorial auction domains.

To show that the property is necessary condition for truthful mechanisms, we observe that the payments cannot depend directly on the declaration t_i of player i, but indirectly through the selected outcome $x(t)$ and the declarations t_{-i} of the other players, that is, $p_i(t) = p_i(x_i(t), t_{-i})$. To see this, suppose that there exist t_i, t'_i such that $x_i(t_i, t_{-i}) = x_i(t'_i, t_{-i})$, but $p_i(t_i, t_{-i}) < p_i(t'_i, t_{-i})$. Then the player whose true processing times are t_i has incentive to declare falsely that its processing times are t'_i in order to increase his utility, as we have $p_i(t_i, t_{-i}) - \sum_{j=1}^{m} t_i x_{ij} < p_i(t'_i, t_{-i}) - \sum_{j=1}^{m} t_i x_{ij}$, contradicting the truthfulness of the mechanism.

When player i has valuations t_i, he has no incentive to declare t'_i when

$$t_i x_i - p_i(x_i, t_{-i}) \leq t_i x'_i - p_i(x'_i, t_{-i})$$

Similarly, when we inverse the roles of t and t', we have

$$t'_i x'_i - p_i(x'_i, t'_{-i}) \leq t'_i x_i - p_i(x_i, t'_{-i})$$

Now if we add the above inequalities and take into account that the instances differ only on the i-th player, that is, $t'_{-i} = t_{-i}$, we get the Monotonicity Property.

The implications are that we don't have to consider at all the payment algorithm. This transforms the problem from the realm of Game Theory to the realm of Algorithms. To prove lower bounds or to design good mechanisms, we can completely forget about mechanisms, payments, truthfulness etc, and simply focus on the subclass of monotone allocation algorithms.

The Monotonicity Property has a straightforward geometric form. For simplicity, let us consider 2 tasks and consider the space of possible valuations for a particular machine i. The generalization to more tasks is straightforward. Fix the values t_{-i} of the remaining players. For every (t_{i1}, t_{i2}), let us consider how a mechanism which satisfies the Monotonicity Property allocates the tasks. In particular, let $R_{x_{i1}x_{i2}}$ denote the set of inputs of player i for which the mechanism has allocation (x_{i1}, x_{i2}) for the i-th player. The Monotonicity Property is equivalent to the constraint that the boundary between $R_{x_{i1}x_{i2}}$ and $R_{x'_{i1}x'_{i2}}$ is of the form $(x'_{i1} - x_{i1})t_{i1} + (x'_{i2} - x_{i2})t_{i2} = 0$. Since the allocation variables x_{i1} and x_{i2} are 0-1, the boundaries have very specific slopes. Therefore the allocation of the mechanism should have one the 2 forms of Figure 1.

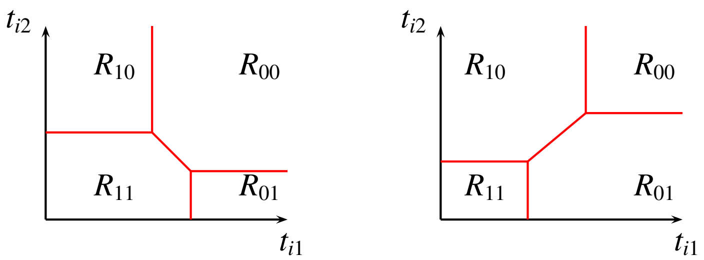

Figure 1: The two possible ways to partition the positive orthant.

In other words, a mechanism is truthful if and only if it partitions the $R^{n\times 2}$ space so that the appropriate lower dimensional cuts have the form of Figure 1. Thus characterizing the truthful mechanisms amounts to characterize the partitions of $R^{n\times m}$ that have specific lower dimensional cuts.

Affine minimizers are the special class of algorithms for which the boundaries in Figure 1 are linear functions of the values of the other players. The diagonal part in the picture exist if and only if the additive constants γ_j are not all equal. On the other hand, threshold mechanisms are exactly those whose diagonal part has 0 length (i.e., the partition is defined by orthogonal hyperplanes).

A recent paper by Dobzinski and Sundararajan [10] gives a simple characterization of mechanisms for 2 machines. They consider only mechanisms which have bounded approximation ratio with respect to makespan and they show that only task-independent mechanisms can be truthful. In [7], a more complete characterization was given which is independent of the approximation ratio: for 2 machines only affine minimizers and threshold algorithms can be truthful[†].

In the next 2 sections we consider positive and impossibility results for the unrelated machines problem. In the last section, we discuss positive results for the related machines version.

4 Upper bounds for the unrelated case

There are only a few positive results which give approximation algorithms for the unrelated machines scheduling problem. We discuss most of them here:

Deterministic mechanisms: Nisan and Ronen [28] gave a mechanism that is n-approximate. The mechanism is essentially the VCG, i.e., it assigns job j to the machine with minimum t_{ij}. It runs independent second-price auctions per item, which is equivalent to the VCG because the valuations in the scheduling domain are additive.

Randomized mechanisms: There are two major notions of truthfulness for randomized mechanisms: *universally truthful* and *truthful in expectation* mechanisms. A universally truthful mechanism is a probability distribution over truthful deterministic mechanisms; this means that even when the players know the outcome of the random choices (coins), they have no incentive to lie. This is in contrast to the truthful in expectation mechanisms where players has no incentive to lie before the random choices but they may have incentive to lie after the random choices.

Nisan and Ronen [28] suggested the following 1.75-approximate randomized mechanism for 2 machines. The mechanism is a universally truthful one and it works as follows: For every task j, with probability $1/2$ the algorithm gives the item to the minimizer of $min\{t_{1j}, \frac{4}{3}t_{2j}\}$, and with $1/2$ to the minimizer of

[†]This holds only for decisive mechanisms, that is mechanisms where all allocations are possible; non-decisive algorithms are not very natural and among their properties is that they have unbounded approximation ratio.

$min\{t_{2j}, \frac{4}{3}t_{1j}\}$. Mu'alem and Schapira [24] extended the mechanism for n machines which gives approximation ratio $1.875n$.

Recently, the result of Nisan and Ronen for 2 machines was improved by Lu and Yu [22] who gave a 1.67-approximation mechanism; they improved this later [23] to 1.59.

Fractional mechanisms: Christodoulou et al. [5] gave an algorithm for the fractional version of the problem which allocates each task independently. The fractions of task j assigned to machines $1, 2, \ldots, n$ are inversely proportional to the squares of the execution times of task j. For example, for 2 machines the allocation of task j is given by

$$x_{1j} = \frac{t_{2j}^2}{t_{1j}^2 + t_{2j}^2} \qquad x_{2j} = \frac{t_{1j}^2}{t_{1j}^2 + t_{2j}^2}.$$

The mechanism has approximation ratio $\frac{n+1}{2}$ and this is optimal for task-independent mechanisms.

Restricted Domain mechanisms: Lavi and Swamy [20] studied two cases where the valuation domain is restricted. Instead of allowing t_{ij} to get any positive real values, they restrict the values to 2: low and high. They show that in such domains there exist algorithms with constant approximation ratio, in contrast to general domains where the current best upper bounds are linear with respect to n.

These domains are not convex and Theorem 3.2 of Saks and Yu [30] does not apply. Instead a more complicated property, the cycle monotonicity property [29], is necessary and sufficient for this domain (and every other domain): The cycle monotonicity property considers closed paths of inputs and requires that the sum of a certain expression is nonnegative over every cycle. The Monotonicity Property is the special case when the cycles have length 2.

When the tasks are allowed to have different low and high values, Lavi and Swamy gave a 3-approximate algorithm which is truthful in expectation. The algorithm computes the optimal fractional solution, it transforms it to a cycle-monotone fractional allocation, and finally rounds it using randomized rounding. When all tasks have the same low and high values, an even better result is possible: they gave a 2-approximate deterministic cycle-monotone algorithm based on maxflow.

5 Lower bounds for the unrelated case

In this section we summarize the main impossibility results and techniques for the unrelated machines problem. It so happens that all the lower bounds for this problem (deterministic, fractional, and randomized) are based on the restrictions of truthfulness and they hold independently of computational complexity considerations. In other words, the lower bounds apply even to exponential-time algorithms.

Nisan and Ronen [28] gave a lower bound of 2 for any truthful deterministic mechanism for 2 machines[‡]. Christodoulou et al. [6] improved the lower bound to $1 + \sqrt{2} = 2.41$ for 3 or more machines, and Koutsoupias and Vidali [17] to $1 + \phi \approx 2.61$ for n machines where n is arbitrarily large. It is a major open problem to close the gap between the lower and the upper bound.

Conjecture (Nisan and Ronen)**.** *No mechanism has approximation ratio better than n.*

Mu'alem and Schapira [24] gave a lower bound of $2 - \frac{1}{n}$ for randomized truthful in expectation mechanisms (which also applies to universally truthful mechanisms). Christodoulou et al. [5] showed that the same bound holds even for fractional domains. Notice that while for deterministic and fractional mechanisms we have tight bounds for 2 machines, for randomized mechanisms there is still a gap between the lower bound of 1.5 and the upper of 1.59. It is an interesting open problem here is to close this gap. Interestingly, even for the restricted domain of two values, Lavi and Swamy [20] showed a lower bound of 11/10.

In the next subsections we discuss the basic ideas behind the lower bounds for deterministic mechanisms. We don't consider randomized and fractional settings, but the main ideas are similar (although sometimes more complicated) [24, 5].

5.1 The case of 2 machines

Recall that every truthful mechanism is monotone. A useful tool that comes out of the Monotonicity Property and is used implicitly or explicitly in most of the lower bound proofs is the following.

Lemma 5.1. *Let t be the input matrix and let $x = x(t)$ be the allocation produced by a truthful mechanism. Suppose that we change only the processing times of machine i in such a way that $t'_{ij} > t_{ij}$ when $x_{ij} = 0$, and $t'_{ij} < t_{ij}$ when $x_{ij} = 1$. A truthful mechanism does not change the allocation to machine i, i.e., $x_i(t') = x_i(t)$.*

Proof. By the Monotonicity Property 3.1, we have that

$$\sum_{j=1}^{m}(t_{ij} - t'_{ij})(x_{ij}(t) - x_{ij}(t')) \leq 0.$$

[‡]It is almost trivial to see that any lower bound for n machines applies to the case of more than n machines.

Observe that all terms of the sum are nonnegative (by the premises of the lemma). The only way to satisfy the inequality is to have all terms equal to 0, that is, $x_{ij}(t) = x_{ij}(t')$. ■

Now we will use this lemma to get easily a lower bound of 2, which first appeared in [28].

Theorem 5.2. *Any truthful mechanism has approximation ratio of at least* 2 *for two or more machines.*

Proof. Suppose that we have an instance with $n = 2$ and $m = 3$ and $t_{ij} = 1$, for all i, j. Any allocation algorithm can either allocate all tasks to a single machine (say the first one), or partition them (say the first two tasks to the first machine and the third task to the second machine). In the former case, we apply Lemma 5.1 to the first player (where the star symbol indicates allocation, and ϵ is an arbitrarily small positive number):

$$t = \begin{pmatrix} 1^\star & 1^\star & 1^\star \\ 1 & 1 & 1 \end{pmatrix} \Rightarrow t' = \begin{pmatrix} 1-\epsilon^\star & 1-\epsilon^\star & 0^\star \\ 1 & 1 & 1 \end{pmatrix}.$$

The resulting assignment on t' has approximation ratio of $\frac{2(1-\epsilon)}{1} \approx 2$. In the latter case, we apply Lemma 5.1 to the second player:

$$t = \begin{pmatrix} 1^\star & 1^\star & 1 \\ 1 & 1 & 1^\star \end{pmatrix} \Rightarrow t' = \begin{pmatrix} 1^\star & 1^\star & 1 \\ 1+\epsilon & 1+\epsilon & 0^\star \end{pmatrix}.$$

The resulting assignment on t' has approximation ratio of $\frac{2}{1+\epsilon} \approx 2$. ■

5.2 The case of 3 or more machines

There is a qualitative difference between the case of 2 machines and the case of 3 or more machines. For 2 machines, the allocation of a player completely determines the allocation of the other player. This is not true for more than 2 players and it complicates the situation.

There are basically two approaches one can follow to prove a lower bound. One approach is to provide a global characterization of all possible mechanisms, such as Roberts' Theorem. This approach however requires the solution of the characterization problem which is potentially a more difficult problem. For example, Christodoulou et al. [7] use this approach to extend the lower bound of 2 [27] to instances with only 2 tasks.

The other approach is to use an appropriately chosen subset of the input instances. The Monotonicity Property implies some relations between the allocations of these instances. We can use them to show that one of the instances has high approximation ratio. A typical application of this approach is forthe lower bound of 2.41 [6] and 2.61 [17]. In [6] as we will see later the set of instances is small. It consists of instances of 2 and 3 machines respectively and no more than 5 tasks. In [17], they use the same principles but apply them in an infinite subset of inputs using a double induction to keep track of how all these allocations depend on each other.

We sketch here the proof of the 2.41 lower bound.

Theorem 5.3. *Any truthful mechanism has approximation ratio of at least* $1 + \sqrt{2}$ *for three or more machines.*

The general idea of the proof is the following: We start with the set of tasks

$$t = \begin{pmatrix} 0 & \infty & \infty & a & a \\ \infty & 0 & \infty & a & a \\ \infty & \infty & 0 & a & a \end{pmatrix},$$

for some parameter $a > 1$. This set of tasks essentially admits two distinct allocations (up to symmetry). This is true because the first three tasks need to be assigned to a single machine by any mechanism with bounded approximation ratio. For each allocation, we increase or decrease some values appropriately. Them is is shown that in order to keep the approximation ratio low (below $1 + \alpha$), the following set of tasks must have the allocation indicated by the stars (in which the first machine gets both tasks 4 and 5):

$$t = \begin{pmatrix} 0^\star & \infty & \infty & 1^\star & 1^\star \\ \infty & 0^\star & \infty & a & a \\ \infty & \infty & 0^\star & a & a \end{pmatrix}.$$

Finally, the input of the first player is modified as the following matrix indicates. By using a lemma that is similar in spirit to Lemma 5.1, but in addition takes into account the fact that there is a unique way to allocate the first three tasks, we get the allocation

$$t = \begin{pmatrix} \alpha^\star & \infty & \infty & 1-\epsilon^\star & 1-\epsilon^\star \\ \infty & 0^\star & \infty & a & a \\ \infty & \infty & 0^\star & a & a \end{pmatrix}.$$

This allocation has an approximation ratio of $\frac{a+2}{\alpha}$, for arbitrarily small value of ϵ. Taking into account that the ratio is at most $1 + \alpha$ we get the theorem.

6 Related machines

In this section, we consider the important special case of the scheduling problem, the *related* machines version. In this setting, the processing times of the tasks are

$p_1 \geq \ldots \geq p_m$, while the machines have speeds $s_1, \ldots, s_n$. Given an assignment of the jobs to the machines, let w_i denote the workload assigned to machine i. The makespan $C(w, s)$ is $\max_i w_i/s_i$. Monotonicity for this special case is very simple. An algorithm is monotone (truthful) when it has the following property: when we decrease the speed of a machine i, keeping all other speeds the same, the new workload on machine i can only decrease.

The mechanism design version of the problem was first studied by Archer and Tardos [3]. It is a very important problem in Algorithmic Mechanism Design, because it's a typical single-parameter problem, which means that each player has only one real private value and his objective is proportional to this value (for a precise definition see Chapters 9 and 12 of [26]). Such problems were studied extensively by Myerson [25]. Furthermore, the optimal allocation is monotone and therefore truthful, but it cannot be computed in polynomial time unless P=NP. It is therefore an appropriate example to explore the interplay between truthfulness and computational complexity. It is a major open problem whether a *deterministic* monotone PTAS exists for this problem[§]. A very recent breakthrough result [9] shows that there exists a randomized truthful-in-expectation PTAS.

In contrast to the scheduling problem of unrelated machines, in this special case there exist truthful mechanisms that output an optimal allocation. A concrete example is the Lex-Opt algorithm which outputs the lexicographically first optimal allocation; the lexicographic order is with respect to the loads $(w_1, \ldots, w_n)$ of the machines.

Theorem 6.1. *[3]* Lex-Opt *is monotone*

Proof. Let $w = (w_1, \ldots, w_n)$ be the workload vector computed by Lex-Opt on input $s = (s_1, \ldots, s_n)$. We consider the case when machine i reports a slower speed $s'_i < s_i$. Let w' be the new schedule for input $s' = (s'_i, s_{-i})$. To show that Lex-Opt is monotone we need to show that $w'_i \leq w_i$.

Clearly the optimal makespan of the new speed vector can only increase, i.e. $C(w', s') \geq C(w, s)$. Let's consider first the case where $C(w, s) = C(w, s')$. The workload vector w is the lexicographically first and therefore Lex-Opt will select this for speeds s'. Clearly in this case, $w'_i = w_i$. In the other case, when the makespan $C(w, s')$ is greater than $C(w, s)$, let's assume that machine i is the bottleneck in schedule w, i.e., $C(w, s') = \frac{w_i}{s'_i} > C(w, s)$. But since w' is the lexicographically-first optimal workload for s', we have that $C(w', s') \leq C(w, s')$, and therefore $\frac{w'_i}{s'_i} \leq C(w', s') \leq C(w, s') = \frac{w_i}{s'_i}$. Again, $w'_i \leq w_i$. ■

We next consider randomized approximation polynomial-time mechanisms and then deterministic ones.

[§]We assume that a mechanism runs in polynomial time when both the allocation algorithm and the payment algorithm run in polynomial time.

6.1 Randomized mechanisms

We will discuss 2 mechanisms in this section. The first mechanism is due to Archer and Tardos [3] and has approximation ratio 3. Later Archer [2] improved the randomized rounding procedure obtaining a 2-approximate mechanism. The second mechanism is due to Dhangwatnotai et al. [9] and is a randomized PTAS. Both mechanisms are truthful-in-expectation and they have similar approach: they create first a monotone fractional solution and then apply a randomized rounding procedure. The randomization is useful only to guarantee truthfulness and has no implication on the approximation ratio.

A 2-approximate truthful in expectation mechanism [3, 2]

Given the speed vector $s = (s_1, \ldots, s_n)$, the algorithm first computes the following threshold

$$T_{LB} = \max_j \min_i \max \left\{ \frac{p_j}{s_i}, \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i} s_l} \right\}, \tag{1}$$

which is a lower bound of the optimal makespan $C(w, s)$.

Given T_{LB}, the algorithm computes a fractional assignment as follows: It assigns the jobs in non-increasing order with respect to their size, i.e. $p_1 \geq \ldots \geq p_m$. It first assigns as many jobs as possible to the fastest machine so that its load becomes equal to T_{LB}. It may have to assign a fraction of some job to achieve this (thus the assignment is fractional). It continues the same procedure for the remaining machines. The threshold is such that all jobs will be assigned to the machines.

We now describe a randomized rounding procedure turn the fractional allocation into an integral one: Pick a random number α uniformly at random in $[0, 1]$. Assume that task j is fractionally assigned to machines i and $i + 1$. If $x_{ij} \geq \alpha$ then assign the task to machine i, otherwise assign it to machine j.

Theorem 6.2. *The above algorithm is monotone.*

Proof. All we need to prove is that the fractional workloads are monotone. This is because the expected workload of every machine is equal to the fractional workload (since α was chosen uniformly).

Assume now that a machine i reports a smaller speed $s_i' < s_i$ and let w and w' be the workload vectors. To show monotonicity we need to show $w_i' \leq w_i$. Let $s_i = \beta \cdot s_i'$, for some $\beta > 1$. The new threshold T_{LB}' can only increase, but it can be bounded by $T_{LB}' \leq \beta \cdot T_{LB}$. If machine i had load T_{LB}, then $w_i' \leq T_{LB}' \cdot s_i' \leq \beta \cdot T_{LB} \cdot s_i' \leq T_{LB} \cdot s_i = w_i$.

If machine i was not full (that is, it had load less than T_{LB}), then it can at most take the load that exceeded the previous machines. Now that the threshold T_{LB}'

has increased, the total workload on those machines with $k < i$ can only increase. Therefore the workload of machine i can only decrease. ∎

A PTAS truthful in expectation mechanism Very recently, Dhangwatnotai et al. [9] suggested the following randomized PTAS, that is truthful in expectation.

The algorithm first groups the jobs of size that differ within a factor of $1 + \epsilon$ from each other, for some small ϵ. Then it smooths the jobs, i.e., it pretends that every job has a size equal to the average of its group. Then the algorithm constructs a set $\mathcal{P}$ of allowable (fractional) partitions of jobs to the machines, giving also a total ordering of these partitions. Then the algorithm optimizes over the partitions in $\mathcal{P}$. From the fractional partition P, we get a fractional schedule $w(P)$, by giving to machine with the i-th slowest speed, the i-th smallest partition set. Then using randomized rounding we get an integral schedule. Finally we replace the smoothed jobs with the real ones. The algorithm does this by random shuffling.

Theorem 6.3. *The above randomized PTAS is monotone.*

Proof. Again it is enough to show that the fractional schedule of the smoothed jobs is monotone.

Assume that machine i reports a smaller speed $s'_i < s_i$ and let $s = (s_i, s_{-i})$ and $s' = (s'_i, s_{-i})$ be the corresponding speed vectors. Let machine i be the k-th slowest in s and the k'-th slowest in s', with $k' \leq k$. Let us denote by $P = P(s)$ and $P' = P(s')$ the corresponding partitions chosen by the algorithm in both cases. Let us also denote by $w(P) = (w_1(P), \ldots, w_n(P))$, the sorted (in increasing order) workload vector with respect to the partition P. We need to prove that $w_{k'}(P') \leq w_k(P)$.

Clearly the makespans satisfy $C(w(P'), s') \geq C(w(P), s)$, since machine i has decreased its speed. Let us first assume that the schedule induced by the partition P, does not increase the makespan for s', i.e. $C(w(P), s') = C(w(P), s)$. Therefore $P' = P$ and since the player i has decreased its position in the sorted speed vector s', it is $w_{k'}(P') = w_{k'}(P) \leq w_k(P)$.

If the schedule induced by P causes an increase of the makespan for s', then the bottleneck is one of the machines in the positions between k' and k, say the machine with index $l \in [k', k]$.

The workload of the machine in this position decreases, i.e. $w_l(P') \leq w_l(P)$ because

$$\frac{w_l(P')}{s'_l} \leq C(w(P'), s') \leq C(w(P), s') = \frac{w_l(P)}{s'_l}.$$

Finally we get $w_{k'}(P') \leq w_l(P') \leq w_l(P) \leq w_k(P)$, as needed. ∎

6.2 Deterministic mechanisms

We now consider deterministic mechanisms for the related machines problem. We distinguish 2 cases: the case of fixed number of machines (for which there is a FPTAS for the non-mechanism-design version) and the case of variable number of machines (for which there is a PTAS but no FPTAS unless P=NP).

6.2.1 Fixed number of machines

Auletta et al. [4] give the first deterministic polynomial-time monotone algorithm for the fixed number of machines problem. Their algorithm is 4-approximate. The algorithm schedules optimally the h largest jobs, for some parameter h and it assigns the rest of the jobs in a greedy fashion. A central point of their approach is that the greedy allocation is monotone for the special case when the speeds are powers of 2. They first round down the originals speeds in the closest power of 2, and then apply their monotone algorithm.

This result was improved by Andelman et al. [1] who gave a PTAS and a different mechanism FPTAS. The PTAS algorithm first modifies the set M of the jobs to a set M' as follows: It partitions the jobs into a set B of big jobs, and a set S of small jobs. A job is in B if its size is above some threshold T. Then, jobs in S are packed into chunks of size in $[T/2, T]$ (the last chunk may have size than $T/2$). Let us call the set of chunks S'. The working set of jobs is the merge of the two sets, $M' = B \cup S'$, for which we find the optimal assignment applying the lexicographically first optimal algorithm Lex-Opt .

The algorithm is trivially monotone, as the construction of the modified job set M' is independent of the speed vector and because the Lex-Opt algorithm is monotone, as we showed in Theorem 6.1.

Andelman et al. [1] gave a different monotone FPTAS for the problem. The algorithm takes any black-box algorithm with approximation ratio c and transforms it to a monotone algorithm with approximation $c(1+\epsilon)$, for every $\epsilon > 0$. They use this on the FPTAS of [15].

The transformation is performed in 3 steps:

1. In the first step the algorithm produces a modified vector of speeds d as follows: First it rounds the speeds down to powers of $(1+\epsilon)$. Then it normalizes the vector such that $d_n = 1$. Finally, it rounds the machines that are very slow, with respect to some threshold L to $(1+\epsilon)^{-L}$.

2. In the second step the algorithm performs an enumeration over all the different vectors d' with speeds $(1+\epsilon)^{-i}$, with $i \in \{0, L\}$. For every such vector d', it applies the black-box algorithm. Finally it sorts the workloads such that the machine with the i-th smallest speed will get the i-th smallest workload.

3. In the final step, it tries all the sorted assignments to d and outputs the assignment that minimizes the makespan (choosing the lexicographically first in case of ties).

6.2.2 Arbitrary number of machines

The following algorithm due to Andelman et al. [1] is based on the ideas of the algorithm of Archer and Tardos (Section 6.1) and has approximation ratio 5. To overcome the problem of derandomizing imposed by monotonicity they modify the speed set.

The currently best deterministic algorithm is due to Kovacs [18]. The algorithm first rounds the speeds down to the closest power of 2, i.e. $d_i = 2^{\lfloor \log s_i \rfloor}$. Then it runs the well-known algorithm Longest Processing Time first (LPT) on the modified speed vector d. Finally, among machines of the same rounded speed, the algorithm reorders the assigned work such that $w_i \leq w_{i+1}$. The algorithm is monotone and attains approximation ratio 2.8 [19]. The proof of its monotonicity is complicated and it is beyond the scope of this article.

7 Conclusions

The scheduling problem with its many facets is one of the driving problems of the area of Algorithmic Mechanism Design. There are many interesting open problems, but we feel that the following are the most important:

- Characterize the set of truthful mechanisms for unrelated machines.
- Close the gap between the lower (2.61) and the upper (n) bound on the approximation ratio for unrelated machines. Also important are the same questions about the fractional and randomized case.
- Give a deterministic PTAS mechanism for the related machines problem or prove that none exists.

References

[1] Nir Andelman, Yossi Azar, and Motti Sorani. Truthful approximation mechanisms for scheduling selfish related machines. *Theory of Computing Systems*, 40(4):423–436, 2007.

[2] Aaron Archer. *Mechanisms for Discrete Optimization with Rational Agents*. PhD thesis, Cornell University, January 2004.

[3] Aaron Archer and Éva Tardos. Truthful mechanisms for one-parameter agents. In *42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 482–491, 2001.

[4] Vincenzo Auletta, Roberto De Prisco, Paolo Penna, and Giuseppe Persiano. Deterministic truthful approximation mechanisms for scheduling related machines. In Volker Diekert and Michel Habib, editors, *STACS*, volume 2996 of *Lecture Notes in Computer Science*, pages 608–619. Springer, 2004.

[5] George Christodoulou, Elias Koutsoupias, and Annamária Kovács. Mechanism design for fractional scheduling on unrelated machines. In *Automata, Languages and Programming: 34th International Colloquium (ICALP)*, pages 40–52, 2007.

[6] George Christodoulou, Elias Koutsoupias, and Angelina Vidali. A lower bound for scheduling mechanisms. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1163–1170, 2007.

[7] George Christodoulou, Elias Koutsoupias, and Angelina Vidali. A characterization of 2-player mechanisms for scheduling. In *Algorithms - ESA 2008, 16th Annual European Symposium*, pages 297–307, 2008.

[8] E. H. Clark. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[9] Peerapong Dhangwatnotai, Shahar Dobzinski, Shaddin Dughmi, and Tim Roughgarden. Truthful approximation schemes for single-parameter agents. In *FOCS*, pages 15–24, 2008.

[10] Shahar Dobzinski and Mukund Sundararajan. On characterizations of truthful mechanisms for combinatorial auctions and scheduling. In *ACM Conference on Electronic Commerce*, pages 38–47, 2008.

[11] Tomás Ebenlendr, Marek Krcal, and Jiri Sgall. Graph balancing: a special case of scheduling unrelated parallel machines. In Shang-Hua Teng, editor, *SODA*, pages 483–490. SIAM, 2008.

[12] T. Groves. Incentives in teams. *Econometrica*, 41:617–663, 1973.

[13] Dorit S. Hochbaum and David B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM J. Comput.*, 17(3):539–551, 1988.

[14] D.S. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co. Boston, MA, USA, 1996.

[15] Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *J. ACM*, 23(2):317–327, 1976.

[16] Roberts Kevin. The characterization of implementable choice rules. *Aggregation and Revelation of Preferences*, pages 321–348, 1979.

[17] E. Koutsoupias and A. Vidali. A lower bound of 1+phi for truthful scheduling mechanisms. In *Mathematical Foundations of Computer Science (MFCS)*, pages 454–464, Krumlov, Czech Republic, 26-31 August 2007.

[18] Annamária Kovács. Fast monotone 3-approximation algorithm for scheduling related machines. In *Algorithms - ESA 2005, 13th Annual European Symposium*, pages 616–627, 2005.

[19] Annamária Kovács. *Fast Algorithms for Two Scheduling Problems.* PhD thesis, Universität des Saarlandes, 2007.

[20] Ron Lavi and Chaitanya Swamy. Truthful mechanism design for multi-dimensional scheduling via cycle monotonicity. In *ACM Conference on Electronic Commerce (EC)*, 2007.

[21] J.K. Lenstra, D.B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(1):259–271, 1990.

[22] Pinyan Lu and Changyuan Yu. An improved randomized truthful mechanism for scheduling unrelated machines. In *STACS*, pages 527–538, 2008.

[23] Pinyan Lu and Changyuan Yu. Randomized truthful mechanisms for scheduling unrelated machines. In *WINE*, pages 402–413, 2008.

[24] Ahuva Mu'alem and Michael Schapira. Setting lower bounds on truthfulness. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1143–1152, 2007.

[25] Roger B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[26] N. Nisan, T. Roughgarden, E. Tardos, and V.V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[27] Noam Nisan and Amir Ronen. Algorithmic mechanism design (extended abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC)*, pages 129–140, 1999.

[28] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[29] Jean-Charles Rochet. A necessary and sufficient condition for rationalizability in a quasilinear context. *Journal of Mathematical Economics*, 16:191–200, 1987.

[30] Michael E. Saks and Lan Yu. Weak monotonicity suffices for truthfulness on convex domains. In *Proceedings 6th ACM Conference on Electronic Commerce (EC)*, pages 286–293, 2005.

[31] W. Vickrey. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, pages 8–37, 1961.