# Mechanism design for fractional scheduling on unrelated machines*

George Christodoulou
Max-Planck-Institut für Informatik

Elias Koutsoupias
University of Athens

Annamária Kovács
Max-Planck-Institut für Informatik

July 28, 2008

**Abstract**

Scheduling on unrelated machines is one of the most general and classical variants of the task scheduling problem. Fractional scheduling is the LP-relaxation of the problem, which is polynomially solvable in the non-strategic setting, and is a useful tool to design deterministic and randomized approximation algorithms.

The mechanism design version of the scheduling problem was introduced by Nisan and Ronen. In this paper, we consider the mechanism design version of the fractional variant of this problem. We give lower bounds for any fractional truthful mechanism. Our lower bounds also hold for any (randomized) mechanism for the integral case. In the positive direction, we propose a truthful mechanism that achieves approximation 3/2 for 2 machines, matching the lower bound. This is the first new tight bound on the approximation ratio of this problem, after the tight bound of 2, for 2 machines, obtained by Nisan and Ronen. For $n$ machines, our mechanism achieves an approximation ratio of $\frac{n+1}{2}$.

Motivated by the fact that all the known deterministic and randomized mechanisms for the problem, assign each task independently from the others, we focus on an interesting subclass of allocation algorithms, the *task-independent* algorithms. We give a lower bound of $\frac{n+1}{2}$, that holds for every (not only monotone) allocation algorithm that takes independent decisions. Under this consideration, our truthful independent mechanism is the best that we can hope from this family of algorithms.

## 1 Introduction

Mechanism design is an important branch of Microeconomics and in particular of Game Theory. The objective of a mechanism designer is to implement a goal, e.g., to sell an object to a set of potential buyers. The problem derives from the fact that the designer may not be informed about some parameters of the input. These values are controlled by selfish agents that may have incentive to misinform the designer, if this can serve their atomic interests. The mechanism design approach concerns the construction of a game, so that the outcome (equilibrium) of the game is the goal of the designer.

Task scheduling is one of the most important and well-studied problems in Computer Science, as it often arises, in numerous forms, as a subproblem in almost every subfield of Computer Science. One of its most classical and general variants is the scheduling on *unrelated* machines. In this setting, there are $n$ machines[1] and $m$ tasks, and the processing time needed by machine $i$ to perform task $j$ is determined by the $t_{ij}$ entry of an $n \times m$ matrix $t$. A common objective is to assign the tasks to the machines in such a way, that the maximum load of the machines (i.e., the makespan) is minimized.

[25] initiated the study of the mechanism design version of scheduling on unrelated machines. In this form of the problem, the processing times that a machine $i$ needs in order to execute the tasks (vector $t_i$),

*A preliminary version of this work appeared in [10]. The second author was partially supported by IST programs IST-2005-15964 (AEOLUS) and IST-2008-215270 (FRONTS).

[1]In game-theoretic settings $n$ is used to denote the number of the players, while in scheduling literature, usually $m$ is used to denote the cardinality of the machines set. In our case, the aforementioned sets coincide. We prefer to use the former notation, in order to be compatible with the original paper [25].

are *private* values that are known only to the corresponding machine. The machines are controlled by selfish agents that aim at satisfying their own interests, and in the particular case they are unwilling to perform any task. In order to motivate them to reveal their actual values, the classical approach adopted by mechanism design is to introduce side payments, i.e., to hire the machines. A mechanism for this problem consists of an allocation algorithm and a payment scheme. We are interested in bounding the approximation ratio of the mechanism's allocation algorithm.

In the classical version of the problem, each task must be assigned to exactly one machine. The LP-relaxation of the problem, also known as fractional scheduling, concerns the version where instead of being assigned to a single machine, each task can be split among the machines. Fractional variations of combinatorial problems have been studied extensively in network optimization, e.g., routing splittable traffic or flow problems.

The fractional scheduling problem can be formulated as a linear program and hence it can be solved in polynomial time. LP-relaxation turns out to be a useful tool in the design of approximation algorithms (both deterministic and randomized)[2]. Furthermore, it turned out to be a powerful technique to provide randomized truthful mechanisms (see e.g. [19, 20, 3]). It is natural to ask how powerful LP-relaxation is in the mechanism design framework.

In this paper we consider the mechanism design version of the fractional scheduling on unrelated machines. An interesting fact is that while the non-strategic version of the problem is polynomially solvable, it turns out that in the mechanism design version of the problem it cannot be solved exactly, even by non-polynomial mechanisms (see Section 3). This means, that the additional properties that the allocation of a mechanism needs to satisfy in contrast to a simple algorithm (cf. Section 2), do not allow us to achieve an exact solution, even in non-polynomial time. Lower bounding fractional mechanisms is a nice approach to lower bound randomized (and deterministic) mechanisms of the integral case. Our lower bound easily extends for those cases (cf. Remark 3).

**Task-Independence** We are especially interested in a family of mechanisms that we call *task-independent*. A task-independent algorithm is any algorithm that in order to allocate task $j$, only considers the processing times $t_{ij}$, that concern the particular task. Such a consideration is motivated by the fact that (to the best of our knowledge) all the known positive results for this problem (e.g., see the mechanisms in [22, 25]), and in addition the mechanism that we propose in this paper, belong to this family of mechanisms. The question that we address here is: *how far can we go with task-independent algorithms?*

## 1.1 Related Work

Scheduling on unrelated machines is a classical NP-hard problem. [21] gave a 2-approximation polynomial time algorithm, while they also proved that the problem cannot be approximated (in polynomial time) within a factor less than 3/2. The mechanism design version of the problem originates in the seminal work of [25]. They gave an $n$-approximation truthful mechanism and a lower bound of 2, while they conjectured the actual bound to be $n$. [11] improved the lower bound to $1+\sqrt{2}$ for 3 or more machines, and [15] to $1+\phi \approx 2.618$ for n machines. Narrowing the gap between the lower and the upper bound still remains a big open question. [20] studied the case where for every task there are two possible running times for every machine. They came up with a 2-approximation truthful mechanism, while they showed a lower bound of 1.14.

Randomization usually reduces the approximation ratio and that is also the case for this problem. [25] proposed a randomized mechanism for 2 machines with approximation ratio 7/4. [22] generalized this to a $\frac{7}{8}n$-approximation randomized truthful mechanism for $n$ machines. In the same work, they also gave a lower bound of $2 - 1/n$ for randomized mechanisms. Notice that all the known lower bounds for this problem (both deterministic and randomized) follow due to the infrastructure of truthful mechanisms, and do not reside in any computational assumption; consequently they hold even for non-polynomial time mechanisms.

From the mechanism design point of view, scheduling on *related* machines, was first studied by [4]. In this variant of the problem, the private parameter for each machine, is a single value (its speed). [4] characterized the class of truthful mechanisms for this setting, in terms of a monotonicity condition of the mechanism's allocation algorithm. A similar characterization for one-parameter mechanism design problems (single item

[2] In fact, it has been used in order to obtain the 2-approximation algorithm in [21].

auction) can also be found in [23]. For this problem, it turns out that the optimal allocation algorithm can be modified to be a truthful mechanism. [4] gave a randomized truthful 3-approximation algorithm, which was later improved to a 2-approximation by [2]. For a fixed number of machines, [5] gave a deterministic truthful 4-approximation algorithm, and [1] improved this by giving an FPTAS. [1] gave the first deterministic polynomial mechanism for the problem, for any number of machines, with an approximation ratio of 5. [16] improved this by giving a 3-approximation deterministic truthful mechanism, while finally the ratio was reduced to 2.8 [17].

In the field of Combinatorial Auctions, a wide variety of combinatorial optimization problems has been considered from the mechanism design point of view (see for example [3, 7, 9, 12, 13, 6] and references within). In this context, [26] characterized the class of truthful mechanisms for combinatorial auctions with convex valuations, generalizing results of [8, 14, 18].

### 1.2 Our results

In this paper, we consider the mechanism design version of fractional scheduling on unrelated machines. We give a $2 - 1/n$ lower bound on the approximation ratio that can be achieved by any truthful mechanism. This result shows that even in the case of such a problem, for which the non-strategic version can be solved exactly in polynomial time, its mechanism design analog may turn out to be impossible to be solved exactly, even by non-polynomial mechanisms. Notice that giving a lower bound for fractional mechanisms is another way to obtain lower bounds for randomized mechanisms for the integral case. Our $2 - 1/n$ lower bound extends the lower bound of [22] to the class of fractional mechanisms. Note that a fractional mechanism is more powerful than a randomized mechanism for the integral case, since it has the flexibility to split a task among many machines, while a randomized mechanism, finally, has to assign the whole task to a machine, and this affects its approximation ratio. Based on the above observation, Remark 3 explains how the lower bound for the fractional mechanisms, can be extended for the randomized mechanisms for the integral case.

In the positive direction, we give a truthful mechanism with approximation ratio 3/2 for 2 machines, which matches our lower bound. This is the first new tight bound that we have for any variant of the problem, after the tight bound of 2 in the integral case, obtained for 2 machines in the original paper of [24]. The generalization of our mechanism for $n$ machines gives us an approximation ratio of $\frac{n+1}{2}$.

Next we turn our attention to a family of mechanisms that we call *task-independent*. This family consists of mechanisms, where the decision for the assignment of a task, depends only on the processing times that concern the particular task (time column that corresponds to the task). Considering task-independence is motivated by the fact that all known 'reasonable' deterministic and randomized mechanisms for this problem are task-independent. Furthermore, this sort of independence has attractive properties: easy to design by applying methods for one-parameter auctions, fits well with on-line settings, where tasks may appear one-by-one. It is natural to ask if there is room for improvement on the approximation ratio by use of such mechanisms. We extend this question for the class of task-independent *algorithms* that need not satisfy the additional properties imposed by truthfulness. We give a lower bound of $\frac{n+1}{2}$ on the approximation ratio of any algorithm that belongs to this class. Our mechanism is also task-independent, and hence is optimal over this family of algorithms.

## 2 Problem definition

In this section we fix the notation that we will use throughout the paper, furthermore we give some preliminary definitions and cite relevant results.

There are $n$ machines and $m$ tasks. Each machine $i \in [n]$ needs $t_{ij}$ units of time to perform task $j \in [m]$. We denote by $t_i$ the row vector corresponding to machine $i$, and by $t^j$ the column vector of the running times of task $j$. We assume that each machine $i \in [n]$ is controlled by a selfish agent that is 'lazy', and therefore reluctant to perform any operation, and vector $t_i$ is private information known only to her. The vector $t_i$ is also called the *type* of agent $i$. In the most general version of the problem, the set $T_i$ of possible types of agent $i$ consists of all vectors $b_i \in \mathbb{R}_+^m$.

Any mechanism defines for each player $i$ a set $A_i$ of available strategies, the player (agent) can choose from. We will consider *direct revelation* mechanisms, i.e., $A_i = T_i$ for all $i$, meaning that the players strategies are to simply report their types to the mechanism. A player may report a false vector $b_i \neq t_i$, if this serves his interests.

A mechanism $M = (x, p)$ consists of two parts:

**An allocation algorithm:** The allocation algorithm $x$, depends on the players' bids $b = (b_1, \ldots, b_n)$, with $0 \leq x_{ij} \leq 1$ denoting the fraction of task $j$ that is assigned to the machine $i$. In the unsplittable case, these variables take only integral values $x_{ij} = \{0, 1\}$. Every task must be completely assigned to the machines' set, so $\sum_{i \in [n]} x_{ij} = 1, \quad \forall j \in [m]$.

**A payment scheme:** The payment scheme $p = (p_1, \ldots, p_n)$, also depends on the bid values $b$. The functions $p_1, \ldots, p_n$ stand for the payments that the mechanism hands to each agent.

The *utility* $u_i$ of a player $i$ is the payment that he gets minus the *actual* time that he needs in order to execute the set of tasks assigned to her, $u_i(b) = p_i(b) - \sum_{j \in [m]} t_{ij} x_{ij}(b)$.

We are interested in *truthful* mechanisms. A mechanism is truthful, if for every player, reporting his true type is a *dominant strategy.* Formally,

$$u_i(t_i, b_{-i}) \geq u_i(t'_i, b_{-i}), \qquad \forall i \in [n], \;\; t_i, t'_i \in T_i, \;\; b_{-i} \in T_{-i},$$

where $T_{-i}$ denotes the possible types of all players disregarding $i$.

We remark here, that once we adopt the solution concept of dominant strategies, focusing on direct revelation and in particular on truthful mechanisms is not at all restrictive, due to the *Revelation Principle.* Roughly, the Revelation Principle states that any problem that can be implemented by a mechanism with dominant strategies, can also be implemented by a truthful mechanism (cf. [23, 25]).

The objective function that we consider in order to evaluate the performance of a mechanism's allocation algorithm, is the maximum load of a machine (makespan). The makespan of the allocation algorithm $x$ with respect to a given input $t$ is

$$Mech(t) \stackrel{\text{def}}{=} \max_{i \in [n]} \sum_{j \in [m]} t_{ij} x_{ij}(t).$$

Since we aim at minimizing the makespan, the optimum is

$$Opt(t) = \min_x \max_{i \in [n]} \sum_{j \in [m]} t_{ij} x_{ij}.$$

We are interested in the approximation ratio of the mechanism's allocation algorithm. A mechanism $M$ is *c-approximate*, if the allocation algorithm is $c$-approximate, that is, if $c \geq \frac{Mech(t)}{Opt(t)}$ for all possible inputs $t$.

Although our mechanism is polynomially computable, we do not aim at minimizing the running time of the algorithm; we are looking for mechanisms with low approximation ratio. Our lower bounds also don't make use of any computational assumptions.

A useful characterization of truthful mechanisms in terms of the following monotonicity condition, helps us to get rid of the payments and focus on the properties of the allocation algorithm.

**Definition 1.** An allocation algorithm is called *monotone*[3] if it satisfies the following property: for every two input matrices $t$ and $t'$ which differ only on machine $i$ (i.e., on the $i$-th 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,

$$\sum_{j \in [m]} (x_{ij} - x'_{ij})(t_{ij} - t'_{ij}) \leq 0.$$

The following theorem states that every truthful mechanism has to satisfy the monotonicity condition. It was used by [25] in order to obtain their lower bounds.

**Theorem 1. [25]** *Every truthful mechanism is monotone.*

[3] Also known as *weakly monotone*.

[26] proved that in the combinatorial auctions setting with convex valuations, monotonicity is also a sufficient condition (i.e., there exist payments that can make a monotone algorithm into a truthful mechanism).

For the one-parameter case, that is when every agent has a single value to declare (e.g., the speed of her machine), [23] (for auction setting) and [4] (for scheduling setting), showed that the monotonicity of the (allocation) algorithm is a necessary and sufficient condition for the existence of a truthful payment scheme. In this case they also provide an explicit formula for the payments. In their theorem cited below, the notion of a *decreasing output function* corresponds to a monotone algorithm in the one-parameter setting.

**Theorem 2. [23, 4]** *The output function admits a truthful payment scheme if and only if it is decreasing. In this case the mechanism is truthful if and only if the payments $p_i(b_i, b_{-i})$ are of the form*

$$h_i(b_{-i}) + b_i x_i(b_i, b_{-i}) - \int_0^{b_i} x_i(u, b_{-i})\, du$$

*where the $h_i$ are arbitrary functions.*

In the original notation of [4], $b_i$ is the declared load (running time) per unit work of agent $i$, and $x_i$ would stand for the work allocated to the agent. Observe that this conforms to our notation: given a single job, $b_i$ is the declared running time of 'one unit of' this job, while the fraction $x_i$ is, indeed, the amount that agent $i$ gets from the job.

# 3 Lower bound for truthful mechanisms

Here we will give a lower bound on the approximation ratio of any fractional truthful mechanism.

**Theorem 3.** *There is no deterministic truthful mechanism that can achieve an approximation ratio better than $2 - \frac{1}{n}$, where $n$ is the number of the machines.*

*Proof.* Let $m = n + 1$, and $t$ be the actual time matrix of the players as below

$$t_{ij} = \begin{cases} 0, & j = i \\ 1, & j = n+1 \\ A, & \text{otherwise.} \end{cases}$$

Let $x = x(t)$ be the corresponding allocation that a truthful mechanism $M = (x, p)$ gives with respect to $t$. Clearly, there is a player $k \in [n]$, with $x_{k,n+1} \geq \frac{1}{n}$. Now, consider the behaviour of the allocation mechanism for the following time matrix as an input

$$t'_{ij} = \begin{cases} \frac{1}{n-1}, & i = j = k \\ 1 - \epsilon, & i = k, j = n+1 \\ t_{ij}, & \text{otherwise.} \end{cases}$$

For significantly large values of $A$, with both inputs $t$ and $t'$, player $k$ gets substantially the whole portion of task $k$, otherwise the approximation ratio is high, e.g., for $A = \frac{2}{\delta}$, both $x_{kk}$ and $x'_{kk}$ are at least $1-(n-1)\delta$, otherwise the approximation ratio is at least 2. Consequently, $|x_{kk} - x'_{kk}| \leq (n-1)\delta$.

The following claim states that due to monotonicity, the mechanism cannot assign to player $k$ a substantially smaller portion of the $n + 1^{\text{st}}$ task than $\frac{1}{n}$.

**Claim 1.** If $x_{k,n+1} \geq \frac{1}{n}$, then for the allocation $x' = x(t')$ on input $t'$ it holds that $x'_{k,n+1} \geq \frac{1}{n} - \epsilon$.

*Proof.* Due to the monotonicity condition (Theorem 1), for every player $i \in [n]$ holds that

$$\sum_{j \in [m]} (t_{ij} - t'_{ij})(x_{ij} - x'_{ij}) \leq 0$$

and by applying this to the $k$-th player we get

$$(0 - \frac{1}{n-1})(x_{kk} - x'_{kk}) + (1 - 1 + \epsilon)(x_{k,n+1} - x'_{k,n+1}) \leq 0,$$

from which we get

$$x'_{k,n+1} \geq x_{k,n+1} + \frac{x'_{kk} - x_{kk}}{\epsilon(n-1)} \geq x_{k,n+1} - \frac{\delta}{\epsilon} \geq \frac{1}{n} - \frac{\delta}{\epsilon}$$

and for $\delta = \epsilon^2$ we finally obtain

$$x'_{k,n+1} \geq \frac{1}{n} - \epsilon.$$

□

On the other hand, an optimal allocation $x^*$ for $t'$ is

$$x^*_{ij} = \begin{cases} 1, & j = i \\ 0, & i = k, j = n+1 \\ \frac{1}{n-1}, & i \neq k, j = n+1 \\ 0, & \text{otherwise} \end{cases}$$

providing optimal makespan $\frac{1}{n-1}$, while the mechanism gives player $k$ a total load of at least

$$(1-(n-1)\delta)\frac{1}{n-1} + \left(\frac{1}{n} - \epsilon\right)(1-\epsilon) > \frac{1}{n-1} + \frac{1}{n} - \delta - \epsilon\left(\frac{n+1}{n}\right).$$

For arbitrary small $\epsilon$, this finally gives an approximation ratio of at least $2 - \frac{1}{n}$. □

*Remark.* Consider a randomized (integral) mechanism. Let $t$ be any input, and $x_{ij}$ denote the probability that machine $i$ receives job $j$ from the mechanism. The expected execution time of $i$ is then given by $\sum_{j \in [m]} t_{ij} x_{ij}$. If the mechanism is truthful in expectation, then formally the monotonicity requirement of Definition 1 has to be fulfilled.

Observe that in a randomized mechanism the expected makespan is at least the maximum expected finish time over the machines (i.e., the makespan of the corresponding fractional mechanism). Still, a lower bound for fractional mechanisms does not automatically imply the same bound for randomized mechanisms, since in the latter case the (integral) optimum makespan may be higher. However, our lower bound can be easily modified so as to hold for any mechanism that is truthful in expectation. The only modification one needs to make is to substitute the $n+1$-st job of the construction with $n \cdot (n-1)$ jobs of the same value $t_{ij} = 1$; our new instance will have $n + n(n-1)$ tasks in total. Following the lines of the previous proof, there exists a player $k$ with $\sum_{j>n} x_{kj} \geq n-1$. Modify the input as follows

$$t'_{ij} = \begin{cases} n, & i = k, j = i \\ 1-\epsilon, & i = k, j > n \\ t_{ij}, & \text{otherwise.} \end{cases}$$

By using a claim analogous to Claim 1 in a straight-forward manner, we can deduce that essentially player $k$ will take the same fraction of jobs $n+1, \ldots, n(n+1)$, while he has to keep also task $k$, and therefore the makespan of the mechanism is at least $2n-1$, while the optimum makespan is $n$. Note that there exists an integral optimum in this instance, and therefore the lower bound holds for randomized mechanisms.

# 4 The truthful mechanism

We describe a truthful mechanism, called SQUARE, for the fractional scheduling problem, with approximation ratio $\frac{n+1}{2}$. On two machines this ratio becomes 3/2, so in this case SQUARE has the best possible worst case ratio w.r.t. truthful mechanisms. Furthermore, in Section 5 we will show that for arbitrary number of machines, our mechanism is optimal among the so called *task-independent* algorithms.

Next, we define the mechanism[4] SQUARE$= (x^{Sq}, p^{Sq})$. Recall that $b_{ij}$ is the reported value for $t_{ij}$, the actual execution time of task $j$ on machine $i$.

[4]In most of the section we will omit the superscripts $^{Sq}$.

**Definition 2** (The mechanism SQUARE$=(x^{Sq}, p^{Sq})$)**.**

**Allocation algorithm:** Let $b^j = (b_{1j}, b_{2j}, \ldots, b_{nj})^T$ be the $j$th column-vector of the input matrix. If $b^j$ has at least one zero coordinate, then SQUARE distributes the $j$th task among machines having zero execution time arbitrarily. If $b_{ij} \neq 0$ $(i \in [n])$, then the fraction of the $j$th task allocated to machine $i$ is

$$x_{ij}^{Sq}(b) = x_{ij}(b) = \frac{\prod_{k\neq i} b_{kj}^2}{\sum_{l=1}^{n} \prod_{k\neq l} b_{kj}^2}. \tag{1}$$

**Payment scheme:** Let the constants $c_{ij}$ be defined as

$$c_{ij} = \frac{\prod_{k\neq i} b_{kj}}{\sqrt{\sum_{l\neq i} \prod_{k\neq l,i} b_{kj}^2}},$$

then the payments $p^{Sq} = (p_1, \ldots, p_n)$ to the agents are

$$p_i(b) = \sum_{j=1}^{m} p_{ij}(b),$$

where

$$p_{ij}(b) = b_{ij} \cdot \frac{c_{ij}^2}{b_{ij}^2 + c_{ij}^2} + c_{ij} \cdot \frac{\pi}{2} - c_{ij} \arctan \frac{b_{ij}}{c_{ij}}.$$

The algorithm $x^{Sq}$ of SQUARE allocates the tasks individually (independently), and so that the fractions of task $j$ assigned to machines $1, 2, \ldots, n$ are inversely proportional to the squares of (declared) execution times of $j$ on the respective machines. For instance, for two machines (1) boils down to

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

For arbitrary $n$ it is obvious that $0 \leq x_{ij} \leq 1$, and $\sum_{i=1}^{n} x_{ij} = 1$. It is easy to see that SQUARE is monotone: Let the input matrix $b$ be changed only on the $i$th row, that is, for any fixed task $j$, just the entry $b_{ij}$ may change. Assume first that in the column-vector $b^j$ all execution times are nonzero. Observe that the variable $b_{ij}$ appears only in the denominator of the expression (1), namely as $b_{ij}^2$, having a positive coefficient. Thus, $x_{ij}$ does not increase when $b_{ij}$ increases, and vice versa. It is easy to see that the same holds if in $b^j$ there are zero entries other than $b_{ij}$, and similarly, if $b_{ij}$ was, or just became the only zero entry. Thus, we obtained that for every single one-parameter problem $b^j$, the assignment is monotone, and this, in turn, implies weak monotonicity (see Definition 1) for $x^{Sq}$.

Now consider $p^{Sq}$. For two machines, the constant $c_{ij}$ is simply the bid of the other machine for this job, that is, $c_{1j} = b_{2j}$ and $c_{2j} = b_{1j}$. In general, for any number of machines it holds that $x_{ij} = c_{ij}^2/(b_{ij}^2 + c_{ij}^2)$; so to speak $c_{ij}$ would be the 'bid' of a single other machine, if we replaced the machines $[n]\backslash\{i\}$ with one machine.

Let us fix a machine $i$. The payment $p_i(b)$ is defined to be the sum of the payments that agent $i$ would get for performing each (fractional) task independently, as determined for truthful mechanisms for one-parameter agents by Theorem 2:

$$p_i(b_i, b_{-i}) = h_i(b_{-i}) + b_i x_i(b_i, b_{-i}) - \int_0^{b_i} x_i(u, b_{-i})\, du.$$

Here the $h_i(b_{-i})$ are arbitrary constants. If we want that the so called *voluntary participation* [4] of the players is ensured (i.e., it is worth taking part in the game), then $h_i$ can be chosen to be $h_i = \int_0^\infty x_i(u, b_{-i})\, du$, so that eventually we get

$$p_i(b_i, b_{-i}) = b_i x_i(b_i, b_{-i}) + \int_{b_i}^{\infty} x_i(u, b_{-i})\, du, \tag{2}$$

for the one-parameter case. We show that applying this formula for each task individually, leads to the payments specified by Definition 2. Assume now that task $j$ is fixed. For this task, the reported execution time $b_i$ becomes $b_{ij}$, whereas the assigned fraction of work $x_i$, becomes $x_{ij}$. Now it is straightforward to check that for task $j$ the formula (2) yields

$$\begin{aligned}
p_{ij}(b) &= b_{ij}x_{ij} + \int_{b_{ij}}^{\infty} x_{ij}(u)\, du \\
&= b_{ij} \cdot \frac{c_{ij}^2}{b_{ij}^2 + c_{ij}^2} + \int_{b_{ij}}^{\infty} \frac{c_{ij}^2}{u^2 + c_{ij}^2}\, du \\
&= b_{ij} \cdot \frac{c_{ij}^2}{b_{ij}^2 + c_{ij}^2} + \left[ c_{ij} \arctan \frac{u}{c_{ij}} \right]_{b_{ij}}^{\infty} \\
&= b_{ij} \cdot \frac{c_{ij}^2}{b_{ij}^2 + c_{ij}^2} + c_{ij} \cdot \frac{\pi}{2} - c_{ij} \arctan \frac{b_{ij}}{c_{ij}}.
\end{aligned}$$

**Theorem 4.** *The mechanism* Square *is truthful.*

*Proof.* To put it short, truthfulness follows from the fact that Square is the sum of $m$ independent truthful mechanisms for the one-parameter problem. Here, we give an elementary proof for *strong* truthfulness. We need to show that for any machine $i$, true time vector $t_i$, and bid vectors of the other machines $b_{-i}$, it holds that

$$u_i(t_i, b_{-i}) \geq u_i(b_i, b_{-i}) \quad \forall b_i \in T_i,$$

and the inequality is strict for $b_i \neq t_i$. Substituting the definition of utility $u_i$, and then considering the payments for each job separately, now our goal is to prove

$$p_i(t_i, b_{-i}) - \sum_{j \in [m]} t_{ij} x_{ij}(t_i, b_{-i}) \geq p_i(b_i, b_{-i}) - \sum_{j \in [m]} t_{ij} x_{ij}(b_i, b_{-i})$$

$$\sum_{j \in [m]} (p_{ij}(t_i, b_{-i}) - t_{ij} x_{ij}(t_i, b_{-i})) \geq \sum_{j \in [m]} (p_{ij}(b_i, b_{-i}) - t_{ij} x_{ij}(b_i, b_{-i})).$$

We claim that the inequality holds for every task $j \in [m]$, that is,

$$p_{ij}(t_i, b_{-i}) - t_{ij} x_{ij}(t_i, b_{-i}) \geq p_{ij}(b_i, b_{-i}) - t_{ij} x_{ij}(b_i, b_{-i}),$$

with strict inequality if $t_{ij} \neq b_{ij}$. Assume that there exist $i$, $j$, $t_i$, $b_{-i}$ and $b_i$ so that

$$p_{ij}(t_i, b_{-i}) - t_{ij} x_{ij}(t_i, b_{-i}) \leq p_{ij}(b_i, b_{-i}) - t_{ij} x_{ij}(b_i, b_{-i}).$$

Plugging in the formulae for the payments $p_{ij}$ and the assigned work $x_{ij}$,

$$\begin{aligned}
t_{ij} &\cdot \frac{c_{ij}^2}{t_{ij}^2 + c_{ij}^2} + c_{ij} \cdot \frac{\pi}{2} - c_{ij} \arctan \frac{t_{ij}}{c_{ij}} - t_{ij} \frac{c_{ij}^2}{t_{ij}^2 + c_{ij}^2} \leq \\
b_{ij} &\cdot \frac{c_{ij}^2}{b_{ij}^2 + c_{ij}^2} + c_{ij} \cdot \frac{\pi}{2} - c_{ij} \arctan \frac{b_{ij}}{c_{ij}} - t_{ij} \frac{c_{ij}^2}{b_{ij}^2 + c_{ij}^2},
\end{aligned}$$

which reduces to

$$\arctan \frac{b_{ij}}{c_{ij}} - \arctan \frac{t_{ij}}{c_{ij}} \leq (b_{ij} - t_{ij}) \cdot \frac{c_{ij}}{b_{ij}^2 + c_{ij}^2}.$$

Equality holds if $b_{ij} = t_{ij}$. Now suppose that $b_{ij} > t_{ij}$. Applying the Mean-Value theorem, we obtain that for some $t_{ij} < \eta < b_{ij}$,

$$\left(\arctan \frac{y}{c_{ij}}\right)'_{y=\eta} = \frac{\arctan \frac{b_{ij}}{c_{ij}} - \arctan \frac{t_{ij}}{c_{ij}}}{(b_{ij} - t_{ij})} \leq \frac{c_{ij}}{b_{ij}^2 + c_{ij}^2}.$$

And this solves to

$$\frac{1}{\frac{\eta^2}{c_{ij}^2} + 1} \cdot \frac{1}{c_{ij}} \leq \frac{c_{ij}}{b_{ij}^2 + c_{ij}^2},$$

$$\frac{1}{\eta^2 + c_{ij}^2} \leq \frac{1}{b_{ij}^2 + c_{ij}^2},$$

a contradiction, since $\eta < b_{ij}$. If $b_{ij} < t_{ij}$, then we obtain $\frac{1}{\eta^2+c_{ij}^2} \geq \frac{1}{b_{ij}^2+c_{ij}^2}$, which contradicts $b_{ij} < \eta < t_{ij}$.

Thus, our mechanism is strongly truthful, since any bid $b \neq t$ leads to strictly less utility, than truth telling. □

## 4.1 Approximation ratio

Let $Squ(t)$ be the makespan of the schedule produced by SQUARE on input $t$, and $Opt(t)$ denote the optimum makespan. In what follows, we show that $\frac{Squ(t)}{Opt(t)} \leq \frac{n+1}{2}$ for any matrix $t$. The next lemma will largely simplify the upper-bound proof:

**Lemma 1.** *If there exists an input instance $t$, such that $Squ(t)/Opt(t) = \alpha$, then there also exists an instance $t^*$, for which $Squ(t^*)/Opt(t^*) = \alpha$, such that there is an optimal allocation of $t^*$ that does not split any job.*

*Proof.* Suppose that $t$ is an input matrix and there is a task (i.e., column-vector) $t^j = \tau = (\tau_1, \tau_2, \dots, \tau_n)^T$ in $t$ that is distributed by some optimal allocation OPT according to $\nu = (\nu_1, \nu_2, \dots, \nu_n)^T$ where $\nu_i < 1\ \forall i$, and $\sum_{i=1}^n \nu_i = 1$. We can assume that $\tau_i > 0$ for every machine $i$, otherwise it is trivial to assign the job to only one machine in an optimal allocation. Now we construct the new instance $t^*$, by introducing $n$ new tasks in place of task $\tau$, namely tasks corresponding to the column-vectors $\nu_1 \cdot \tau,\ \nu_2 \cdot \tau,\ \dots,\ \nu_n \cdot \tau$.

We claim that $Opt(t) = Opt(t^*)$, and this optimum can be obtained without splitting the new jobs. Notice first, that it yields the original optimum makespan $Opt(t)$, if we allocate the first task completely to the first machine, the second one completely to the second machine, and so on. Indeed, the execution times on the machines due to the new jobs are then $(\nu_1 \cdot \tau_1,\ \nu_2 \cdot \tau_2,\ \dots,\ \nu_n \cdot \tau_n)^T$, which is the same as the execution times due to job $\tau$ in the allocation OPT.

On the other hand, suppose that on input $t^*$ some schedule OPT* yields a better makespan than $Opt(t)$, where OPT* splits the new jobs according to the distributions

$$\begin{pmatrix} \xi_{11} \\ \xi_{21} \\ \vdots \\ \xi_{n1} \end{pmatrix}, \quad \begin{pmatrix} \xi_{12} \\ \xi_{22} \\ \vdots \\ \xi_{n2} \end{pmatrix}, \quad \dots \quad , \begin{pmatrix} \xi_{1n} \\ \xi_{2n} \\ \vdots \\ \xi_{nn} \end{pmatrix}.$$

In this case, on input $t$ keeping OPT* for the unchanged jobs, and then splitting $\tau$ according to the distribution $(\sum_{s=1}^n \xi_{1s}\nu_s,\ \sum_{s=1}^n \xi_{2s}\nu_s,\ \dots, \sum_{s=1}^n \xi_{ns}\nu_s)^T$ would yield a lower makespan than $Opt(t)$ as well. Observe that the distribution of $\tau$ is valid, since

$$\sum_{k=1}^n \sum_{s=1}^n \xi_{ks}\nu_s = \sum_{s=1}^n \nu_s \cdot (\sum_{k=1}^n \xi_{ks}) = \sum_{s=1}^n \nu_s \cdot 1 = 1.$$

Moreover, it would result in the same execution times as OPT* for the set of new jobs in $t^*$.

Finally, a straightforward calculation shows that $Squ(t) = Squ(t^*)$ also holds. Given the input $t^*$, let us consider the fraction of the $s$th new job on machine $i$ as determined by the formula (1) for $x^{Sq}$. We get

$$\frac{\prod_{k\neq i}(\nu_s\tau_k)^2}{\sum_{l=1}^n\prod_{k\neq l}(\nu_s\tau_k)^2} = \frac{\prod_{k\neq i}\tau_k^2}{\sum_{l=1}^n\prod_{k\neq l}\tau_k^2}.$$

Therefore, the execution time of this (fractional) task on machine $i$ is

$$\frac{\prod_{k\neq i}\tau_k^2}{\sum_{l=1}^n\prod_{k\neq l}\tau_k^2}\cdot\nu_s\tau_i;$$

and the execution times of all new tasks on this machine total to

$$\sum_{s=1}^n\left(\frac{\prod_{k\neq i}\tau_k^2}{\sum_{l=1}^n\prod_{k\neq l}\tau_k^2}\cdot\nu_s\tau_i\right) = \frac{\prod_{k\neq i}\tau_k^2}{\sum_{l=1}^n\prod_{k\neq l}\tau_k^2}\cdot\tau_i\sum_{s=1}^n\nu_s = \frac{\prod_{k\neq i}\tau_k^2}{\sum_{l=1}^n\prod_{k\neq l}\tau_k^2}\cdot\tau_i\cdot 1.$$

This is the same as the running time of the fraction of task $\tau$ on machine $i$ given the original input $t$. □

**Theorem 5.** *For the approximation ratio of* Square, $\frac{Squ(t)}{Opt(t)} \leq \frac{n+1}{2}$ *holds, where $n$ denotes the number of machines, and $t$ is an arbitrary set of input tasks.*

*Proof.* Consider the input $t$. Due to the previous lemma, we can assume that the (indices of) tasks are partitioned into the sets $J_1,\ J_2,\ \dots, J_n$, so that there is an optimal allocation Opt where job $t^j$ is allocated completely to machine $i$, if and only if $j \in J_i$. We can also assume that $t_{ij} > 0$ for all $i$ and $j$. Otherwise we would have a job that adds zero execution time to the makespan in both the allocation of Square, and of Opt, and removing this job from the input would not affect the approximation ratio. For the optimum makespan it holds that

$$Opt(t) = \max_{i\in[n]}\sum_{j\in J_i} t_{ij}. \tag{3}$$

For the running time of an arbitrary machine $i$ in Square, we have

$$Squ_i(t) = \sum_{r=1}^n\sum_{j\in J_r} x_{ij}(t)t_{ij},$$

where the $x_{ij}(t)$ are defined by (1). We decompose the above expression as follows:

$$Squ_i(t) = \sum_{j\in J_i} x_{ij}t_{ij} + \sum_{r\neq i}\sum_{j\in J_r} x_{ij}t_{ij}.$$

We can upper bound the first sum using (3), and the fact that $x_{ij} \leq 1$ :

$$\sum_{j\in J_i} x_{ij}t_{ij} \leq \sum_{j\in J_i} 1\cdot t_{ij} \leq Opt(t).$$

Next we upper bound every sum of the form $\sum_{j\in J_r} x_{ij}t_{ij} \quad (r\neq i)$, by $\frac{1}{2}\cdot Opt(t)$. Since there are $n-1$ such sums, this will prove that

$$Squ_i(t) \leq Opt(t) + (n-1)\cdot\frac{1}{2}\cdot Opt(t) = (1+\frac{n-1}{2})\cdot Opt(t).$$

Since $i$ was an arbitrary machine, eventually this implies

$$Squ(t) = \max_{i\in[n]} Squ_i(t) \leq (1+\frac{n-1}{2})\cdot Opt(t).$$

The bound $\sum_{j\in J_r} x_{ij}t_{ij} \leq \frac{1}{2}\cdot Opt(t)$ can be proven as follows:

$$
\begin{aligned}
\sum_{j\in J_r} x_{ij}t_{ij} &= \sum_{j\in J_r} \frac{\prod_{k\neq i} t_{kj}^2}{\sum_{l=1}^{n}\prod_{k\neq l} t_{kj}^2}\cdot t_{ij} \\
&= \sum_{j\in J_r} \frac{t_{ij}t_{rj}\prod_{k\neq i,r} t_{kj}^2}{\sum_{l=1}^{n}\prod_{k\neq l} t_{kj}^2}\cdot t_{rj} \\
&= \sum_{j\in J_r} \frac{t_{ij}t_{rj}}{t_{ij}^2+t_{rj}^2+\sum_{l\neq i,r} t_{ij}^2t_{rj}^2/t_{lj}^2}\cdot t_{rj} \\
&\leq \sum_{j\in J_r} \frac{t_{ij}t_{rj}}{t_{ij}^2+t_{rj}^2}\cdot t_{rj} \\
&\leq \sum_{j\in J_r} \frac{1}{2}\cdot t_{rj} \qquad (4) \\
&= \frac{1}{2}\sum_{j\in J_r} t_{rj} \\
&\leq \frac{1}{2}\cdot Opt(t).
\end{aligned}
$$

The inequality (4) follows from $\frac{\alpha\beta}{\alpha^2+\beta^2} \leq \frac{1}{2}$, which holds for any two positive real numbers. The last inequality is implied by (3). □

**Corollary 1.** *For two machines the truthful mechanism* SQUARE *has approximation ratio* $3/2$, *which is the best worst case ratio we can expect from any truthful mechanism for the fractional scheduling problem.*

# 5 Lower bound for independent algorithms

In this section we prove a lower bound of $\frac{n+1}{2}$ for the worst case ratio of independent fractional algorithms. An algorithm is independent, if it allocates the tasks independently of each-other, or formally:

**Definition 3.** An allocation algorithm $x$ is called *task-independent*, or simply *independent*, if the following holds: If $t$ and $t'$ are two $n \times m$ input matrices, such that for the $j$th task $t_{ij} = t'_{ij}$ ($\forall i \in [n]$), then for this task it also holds that $x_{ij} = x'_{ij}$ ($\forall i \in [n]$).

It is remarkable, that the currently known best mechanisms (in fact, any 'reasonable' mechanism we know of) are all independent, in the integral, the randomized, and the fractional case. It is not difficult to come up with independent (suboptimal) algorithms, which are also weakly monotone. However it seems to be an intriguing question, whether there exist non-inependent, and still monotone algorithms having better approximation ratio than the best independent ones. We note that in the integral case it is easy to construct an instance with $n$ machines and $n^2$ tasks, that proves a lower bound of $n$ (i.e., tight bound) for independent algorithms: Consider a task-independent algorithm for the integral problem, and the input matrix where $t_{ij} = 1$ for all $i \in [n]$ and $j \in [n^2]$. By the pigeonhole principle, the algorithm allocates at least $n$ jobs to one of the machines. Assume w.l.o.g. that this is the first machine, and it receives (at least) the first $n$ jobs. Now we set every $t_{ij} = 0$ for all jobs $j > n$, and keep $t_{ij} = 1$ if $j \leq n$. Due to the independence, the first machine still receives the first $n$ tasks, and the makespan becomes $n$, whereas the optimum makespan is obviously 1.

**Theorem 6.** *If $x$ is an independent fractional allocation algorithm for the unrelated machines problem, then it has approximation ratio of at least $\frac{n+1}{2}$, where $n$ denotes the number of machines.*

*Proof.* In order to obtain the lower bound, consider the following input matrix with $n \geq 2$ machines and $m = 1 + \binom{n}{2}$ tasks. The first task has execution time 0 on every machine; furthermore, for all $\binom{n}{2}$ possible pairs of machines $(i_1, i_2)$ there is a task $j$ with $t_{i_1j} = t_{i_2j} = 1$ and $t_{ij} = A$ for $i \notin \{i_1, i_2\}$ :

$$t = \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 & A & \cdots & A \\ 0 & 1 & A & \cdots & A & 1 & \cdots & A \\ 0 & A & 1 & \cdots & A & 1 & \cdots & A \\ \vdots & \vdots & & & & & & \vdots \\ 0 & A & A & \cdots & A & A & \cdots & 1 \\ 0 & A & A & \cdots & 1 & A & \cdots & 1 \end{pmatrix}.$$

By setting $A$ to a large enough number, we can ensure – similar to the proof of Theorem 3 – that the corresponding share of a player of a certain task is arbitrarily small, otherwise the approximation ratio gets too large. That is, we can assume that the bulk of any job is allocated to the machines having execution time 1 for this job.

Let us consider an arbitrary independent algorithm $x$. Observe that no matter how $x$ allocates the above tasks, the total running time of all the jobs cannot be less than $\binom{n}{2}$. Thus, there exists a machine, say the first one, with running time at least $\binom{n}{2}/n = \frac{n-1}{2}$. Now we modify the instance $t$ to $t'$ as follows: we keep the original execution times of tasks that had running time 1 on the first machine, and zero out all other $t_{ij}$; furthermore, the very first task will now have execution time 1 on the first machine, and $A$ on other machines.

$$t' = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ A & 1 & A & \cdots & A & 0 & 0 & \cdots & 0 \\ A & A & 1 & \cdots & A & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & & \vdots & \vdots & \vdots & & \vdots \\ A & A & A & \cdots & A & 0 & 0 & \cdots & 0 \\ A & A & A & \cdots & 1 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

As noted above, on instance $t$ at least $\frac{n-1}{2} - \epsilon$ running time on the first machine was due to jobs that have execution time 1 on this machine, i.e., to the jobs $2, \dots, n$. Since the algorithm $x$ is task-independent, on input instance $t'$ the first machine gets the same allocation over jobs $2, \dots, n$, and also gets a $(1-\epsilon)$ fraction of job 1, achieving a running time of at least $1 + (n-1)/2 - 2\epsilon$, for any $\epsilon > 0$. On the other hand, it is clear that the optimal allocation has makespan 1. □

**Corollary 2.** *The mechanism* Square *has optimal approximation ratio among all independent mechanisms.*

It can be shown that among all allocations where the distribution of task $j$ is proportional to $(t_{1j}^{-\alpha}, t_{2j}^{-\alpha}, \dots, t_{nj}^{-\alpha})$ for some $\alpha > 0$, the above optimal approximation ratio is obtained if and only if $\alpha = 2$. We sketch the proof of this statement next. In the general case

$$x_{ij}(t) = \frac{\prod_{k \neq i} t_{kj}^{\alpha}}{\sum_{l=1}^{n} \prod_{k \neq l} t_{kj}^{\alpha}}.$$

Consider the proof of Theorem 5, and denote by $c$ the quotient $t_{ij}/t_{rj}$ in the inequality (4). Now the same argument implies a worst case ratio of $1 + (n-1) \cdot B$, where $B$ is an upper bound on $c/(c^{\alpha}+1)$ for all $c > 0$. The function $f(c) = c/(c^{\alpha}+1)$ is unbounded for $\alpha < 1$ ($B = \infty$), resp. is bounded by $B = 1$ for $\alpha = 1$; for $\alpha > 1$ it has its only maximum in $c = (\alpha-1)^{-1/\alpha}$ (that is, $B = f((\alpha-1)^{-1/\alpha})$.

The worst case ratio obtained this way is 'tight' as shown by the following input to the mechanism:

$$t = \begin{pmatrix} 1 & C & C & \cdots & C \\ \infty & 1 & \infty & \cdots & \infty \\ \infty & \infty & 1 & \cdots & \infty \\ \vdots & \vdots & & & \vdots \\ \infty & \infty & \infty & \cdots & \infty \\ \infty & \infty & \infty & \cdots & 1 \end{pmatrix}$$

The optimal allocation of this input has makespan 1, while our mechanism assigns a running time of $1 + (n-1) \cdot \frac{1}{C^\alpha+1} \cdot C$ to the first machine. Setting $C \longrightarrow \infty$ proves that the mechanism has unbounded worst case ratio for $\alpha < 1$, respectively a worst case ratio of $1 + (n-1) \cdot 1 = n$ for $\alpha = 1$. If $\alpha > 1$, then let $C = (\alpha - 1)^{-1/\alpha}$. This implies a makespan of $1 + (n-1) \cdot [(\alpha-1)^{\frac{\alpha-1}{\alpha}}/\alpha]$. The term $(\alpha-1)^{\frac{\alpha-1}{\alpha}}/\alpha$ has its minimum (of value $\frac{1}{2}$) at $\alpha = 2$. For any other $\alpha$, the approximation ratio on the given input is larger than $1 + (n-1) \cdot \frac{1}{2}$.

# 6 Conclusion

In this paper, we discuss the application of mechanism design for the fractional scheduling problem on unrelated machines. We give a lower bound on the approximation ratio of truthful mechanisms, and we come up with a matching upper bound for 2 machines. The generalization of our mechanism gives us an upper bound that is linear in the number of the machines. After that, we focus on an interesting class of mechanisms with appealing properties, i.e. task-independent mechanisms. We obtain a lower bound on the approximation ratio of any algorithm in this class. This bound shows that our mechanism is optimal w.r.t. this class.

In all the versions of the scheduling on unrelated machines (i.e. fractional, randomized, integral), we have a constant lower bound and an upper bound that is linear in the number of the machines. [25], conjectured that for the integral case, there is no deterministic mechanism that can achieve a better approximation ratio. For special cases, we know that fractional and randomized mechanisms can attain a better preformance. But is this asymptotically true? Can we hope to construct fractional and randomized mechanisms with sublinear approximation factor, even in exponential running time? Our lower bound for task-independent algorithms, shows that in order to improve the performance, we need to consider more sophisticated mechanisms, that exploit the input information in a more global way. Thus, we also need to come up with new techniques that overcome the monotonicity constraints imposed by truthfulness.

# References


[1] Nir Andelman, Yossi Azar, and Motti Sorani. Truthful approximation mechanisms for scheduling selfish related machines. In *22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 69–82, 2005.

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

[3] Aaron Archer, Christos H. Papadimitriou, Kunal Talwar, and Éva Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 205–214, 2003.

[4] 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.

[5] Vincenzo Auletta, Roberto De Prisco, Paolo Penna, and Giuseppe Persiano. Deterministic truthful approximation mechanisms for scheduling related machines. In *21st Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 608–619, 2004.

[6] Moshe Babaioff, Ron Lavi, and Elan Pavlov. Mechanism design for single-value domains. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference (AAAI)*, pages 241–247, 2005.

[7] Yair Bartal, Rica Gonen, and Noam Nisan. Incentive compatible multi unit combinatorial auctions. In *Proceedings of the 9th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 72–87, 2003.

[8] Sushil Bikhchandani, Shurojit Chatterji, Ron Lavi, Ahuva Mu'alem, Noam Nisan, and Arunava Sen. Weak monotonicity characterizes deterministic dominant strategy implementation. *Econometrica*, 74(4):1109–1132, 2006.

[9] Patrick Briest, Piotr Krysta, and Berthold Vöcking. Approximation techniques for utilitarian mechanism design. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 39–48, 2005.

[10] 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.

[11] 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.

[12] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 610–618, 2005.

[13] Shahar Dobzinski, Noam Nisan, and Michael Schapira. Truthful randomized mechanisms for combinatorial auctions. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 644–652, 2006.

[14] Hongwei Gui, Rudolf Müller, and Rakesh V. Vohra. Dominant strategy mechanisms with multidimensional types. In *Computing and Markets*, 2005.

[15] Elias Koutsoupias and Angelina Vidali. A lower bound of 1+*phi* for truthful scheduling mechanisms. In *Mathematical Foundations of Computer Science, 32nd International Symposium (MFCS)*, pages 454–464, 2007.

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

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

[18] Ron Lavi, Ahuva Mu'alem, and Noam Nisan. Towards a characterization of truthful combinatorial auctions. In *44th Symposium on Foundations of Computer Science (FOCS)*, pages 574–583, 2003.

[19] Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 595–604, 2005.

[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] 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.

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

[24] 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.

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

[26] 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.