A lower bound for scheduling mechanisms*

George Christodoulou† Elias Koutsoupias‡ Angelina Vidali §

Abstract

We study the mechanism design problem of scheduling tasks on n unrelated machines in which the machines are the players of the mechanism. The problem was proposed and studied in the seminal paper of Nisan and Ronen on algorithmic mechanism design, where it was shown that the approximation ratio of mechanisms is between 2 and n. We improve the lower bound to $1 + \sqrt{2}$ for 3 or more machines.

1 Introduction

The study of mechanisms in game-theoretic settings is an important area at the intersection of Computer Science and Game Theory. A particular type of mechanisms, for which auctions is a typical example, is the mechanism design problem. Mechanisms are a special class of algorithms and the study of their computational properties was initiated by Nisan and Ronen in their seminal paper [23]. The focus of their paper was on the task allocation problem on unrelated machines. They showed that no mechanism can have approximation ratio better than 2. They conjectured that this lower bound is not tight. In this paper[1], we confirm this and improve the lower bound of the approximation ratio to $1 + \sqrt{2}$.

The scheduling problem we consider here is one of the most fundamental scheduling problems [13, 20]. There are n machines and m tasks and each task may have different execution times on the machines. Let t_{ij} be execution time of task j on machine i. The objective is to schedule the tasks on the machines to minimize the makespan, that is, the time by which all tasks are finished. In the mechanism design setting, each machine i knows its own times (the t_{ij}'s), but the algorithm does not know them. We consider *direct revelation* mechanisms which first ask the machines to declare their times t_{ij} and then proceed to allocate the tasks according to a policy known to machines in advance. The machines are selfish players who are lazy and don't want to execute the tasks, so they may lie. To deal with this problem, the mechanism pays the machines according to their declarations. Thus the mechanism design problem consists of two algorithms: an allocation

*Supported in part by IST-15964 (AEOLUS) and the Greek GSRT.
†Max-Planck-Institut für Informatik, Saarbrücken, Germany Email: gchristo@mpi-inf.mpg.de
‡Department of Informatics, University of Athens Email: elias@di.uoa.gr
§Department of Informatics, University of Athens Email: avidali@di.uoa.gr
[1]A preliminary version of this paper appeared in [9].

algorithm and a payment algorithm. They both take as input the declaration of times by the machines and produce an allocation and a set of payments, one for each machine.

The objective of each machine is to minimize the load of tasks allocated to it, minus its payment. On the other hand, the objective of the mechanism is to minimize the makespan of the allocation. Notice that the mechanism does not care how much it pays the machines. The payments are given to machines as an incentive to tell the truth. A mechanism is called *truthful* when telling the truth is a dominant strategy for each player, independently of the declarations of the other players. A classical result in mechanism design, the Revelation Principle, states that for every mechanism, in which each player has a dominant strategy, there is a truthful mechanism which achieves the same objective. The reason is that given a non-truthful mechanism, we can transform it to a truthful one by promising the players that the mechanism itself will simulate their (lying) strategy. With the Revelation Principle, we are free to concentrate on truthful mechanisms (at least for the class of centralized mechanisms).

There are two major classes of problems in algorithmic mechanism design. For every problem of the first class, there exists an optimal truthful mechanism but the problem is NP-hard (i.e., the problem of computing the optimal allocation is NP-hard). For this kind of problems, we are interested in truthful *polynomial-time approximation* algorithms. Two typical problems in this class are the problem of combinatorial auctions and the problem of scheduling related machines. The second class contains problems that need not be NP-hard, but for which no optimal mechanism is truthful. The quintessential problem in this class is the scheduling unrelated machines problem. For this kind of problems, we can ask either about the optimal approximation ratio of *all* algorithms, or the optimal approximation ratio of *polynomial-time* algorithms. In this paper, we deal with the approximation ratio of all algorithms, not necessarily polynomial-time ones. In other words, *the lower bound of* $1+\sqrt{2}$ *is based on the restrictions imposed only by truthfulness*, not by the computational hardness of the problem.

2 Related Work

The scheduling problem on unrelated machines is one of the most fundamental scheduling problems [13]. Lenstra, Shmoys, and Tardos [20] gave a 2-approximation polynomial-time algorithm for the classical version of the problem. They also showed that the problem cannot be approximated in polynomial time within a factor less than 3/2.

Here we study its mechanism design version and we improve the results of Nisan and Ronen [23, 24], who introduced the problem and initiated the algorithmic theory of Mechanism Design. They gave a truthful n-approximate (polynomial-time) algorithm; they also showed that no mechanism (polynomial-time or not) can achieve approximation ratio better than 2. They conjectured that there is no deterministic mechanism with approximation ratio less than n. On the other hand, they gave a randomized truthful mechanism for two players, that achieves an approximation ratio of 7/4.

Recently, Mu'alem and Schapira [21] proved a lower bound of $2 - \frac{1}{n}$ for any randomized truthful mechanism for n machines and generalized the mechanism in [23] to give a $7n/8$ upper bound. The lower bound result was generalized in [8], by showing that no fractional truthful mechanism can achieve an approximation ratio better than $2 - 1/n$. They also showed that fractional algorithms that treat each task independently cannot do better than $(n + 1)/2$ and this bound is tight.

Lavi and Swamy [19] aim at improving the upper bound instead of producing improved lower bounds. They consider a special case of the same problem—namely when the processing times have only two possible values low or high—and devise a deterministic 2-approximation truthful mechanism.

A simpler variant of the scheduling problem is the problem on related machines. In this case, for each machine there is a single value (instead of a vector), its speed. Myerson [22] gave a characterization of truthful algorithms for this kind of problems (one-parameter problems), in terms of a monotonicity condition. Archer and Tardos [4] found a similar characterization and using it obtained a variant of the optimal algorithm which is truthful (albeit exponential-time). They also gave a polynomial-time randomized 3-approximation mechanism, which was later improved to a 2-approximation, in [2]. This mechanism is truthful in expectation. Andelman, Azar, and Sorani [1] gave a 5-approximation deterministic truthful mechanism, in the same framework. Kovács improved the approximation ratio to 3 [16] and to 2.8 [17].

Much more work has been done in the context of combinatorial auctions (see for example [3, 6, 7, 10, 5, 11] and the references within).

Saks and Yu [25] proved that, for mechanism design problems with convex domains which includes the scheduling problem, a simple necessary monotonicity property is also sufficient for truthful mechanisms, generalizing results of [12, 18].

3 Problem definition

Definition 1 (The scheduling problem)**.** The input to the scheduling problem is a nonnegative matrix t of n rows, one for each machine-player, and m columns, one for each task. The entry t_{ij} (of the i-th row and j-th column) is the time it takes for machine i to execute task j. Let t_i denote the times for machine i, which is the vector of the i-th row. The output is an allocation $x = x(t)$, which partitions the tasks into the n machines and a payment rule $p(t)$.

Here we follow the usual notation of game theory literature where n denotes the number of players/machines. This is different from the standard notation of the scheduling literature where n is the number of tasks and m the number of machines. We will also use the standard game-theoretic convention a_{-i} to denote what remains from a vector a when we drop its i-th element; similarly, (a'_i, a_{-i}) denotes the vector that we get when we replace a_i by a'_i.

We describe the partition of the allocation rule using indicator values $x_{ij} \in \{0,1\}$: $x_{ij} = 1$ iff task j is allocated to machine i. Of course, we should allocate each task to exactly one machine, or more formally $\sum_{j=1}^{m} x_{ij} = 1$.

Let also $p_i(t)$ denote the payment which the mechanism pays to player i when the players declare times t.

For truthful mechanisms, the payments do not depend directly on the declaration t_i of player i, but indirectly through the allocation $x_i = x_i(t)$ and the times of the other players as the following lemma [18] states. For completeness, we prove the lemma here.

Lemma 1 ([18])**.** *The price $p_i(t)$ of a truthful mechanism does not depend on the declaration t_i of player i, but only on its allocation $x_i(t)$ and the declarations of the other players, that is $p_i(t) = p_i(x_i(t), t_{-i})$.*

Proof. Suppose towards a contradiction 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 $p_i(t_i, t_{-i}) - \sum_{j=1}^{m} t_i x_{ij} < p_i(t'_i, t_{-i}) - \sum_{j=1}^{m} t_i x_{ij}$; this contradicts the assumption that the mechanism is truthful. □

A mechanism consists of two algorithms, an allocation mechanism and a payment algorithm. However, we are interested only in the approximation ratio of the allocation algorithm. We can then ask which allocation algorithms admit some payment algorithm so that the resulting mechanism is truthful. It turns out that there is a very simple and appealing property that these allocation mechanisms satisfy, monotonicity. More precisely:

Definition 2 (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). The following proposition was shown in [24].

Proposition 1. *Every truthful mechanism satisfies the Monotonicity Property.*

Proof. When player i gets 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

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

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 lemma. $\square$

The Monotonicity Property states that $(x_i - x'_i) \cdot (t_i - t'_i) \le 0$ is a necessary condition for truthfulness. It turns out that it is also sufficient condition [25], but we will not use this fact here. 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 design a good mechanism, we can completely forget about mechanisms, payments, truthfulness etc, and simply focus on the subclass of monotone allocation algorithms.

Monotonicity, which is not specific to the scheduling task problem but it has much wider applicability [25], poses a new challenging framework for designing algorithms. In the traditional theory of algorithms, the algorithm designer could concentrate on how to solve every instance of the problem by itself. With monotone algorithms, this is no longer the case. The solutions for one instance must be consistent with the solutions of the remaining instances—they must satisfy the Monotonicity Property. Putting it in another way, monotone algorithms are holistic algorithms: they must consider the whole space of inputs together.

4 The tools for the proof

In our proof of the lower bound, we will exploit the Monotonicity Property of truthful mechanisms. In this section, we present three important lemmas that follow from the Monotonicity Property and will be the tools for our proof.

The first lemma will be used repeatedly and is due to Nisan and Ronen [24]. They have used it to obtain their lower bounds in their original paper. It is a specific and direct way to take advantage of the Monotonicity Property. It states that if a machine gets a set of tasks when it declares t_i, it will get exactly the same set of tasks if we lower the execution time of the tasks allocated to the machine and increase the execution time of the remaining tasks.

It is convenient to allow instances with times $t_{ij} = \infty$. When only finite times are allowed, all the statements are still true; in this case ∞ will simply denote an appropriate arbitrarily high value.

Lemma 2. *a) Let t be a matrix of processing times and let $x = x(t)$ be the allocation produced by a truthful mechanism. Suppose that we change only the processing times of machine i and in such a way that $t'_{ij} > t_{ij}$ when $x_{ij} = 0$, and $t'_{ij} < t_{ij}$ when $x_{ij} = 1$. The mechanism does not change the allocation to machine i, i.e., $x_i(t') = x_i(t)$. (However, it may change the allocation of other machines).*

b) We can strengthen the lemma for mechanisms of bounded approximation ratio when all times $t_{i'j}$ of some task j are ∞ except of the value $t_{ij} = 0$. When we change the values as in the first part of the lemma and we now set $t'_{ij} = 1$ (or any other bounded value), the mechanism again does not change the allocation to machine i.

Proof. By the Monotonicity Property, we have

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

For the first property, 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, $x_{ij}(t) = x_{ij}(t')$.

The second property has very similar proof. Simply observe that task j can only be processed by machine i for mechanisms of bounded approximation ratio. Therefore $x_{ij}(t) = x_{ij}(t') = 1$. The remaining terms of the Monotonicity Property sum must be nonnegative and the lemma follows. □

To simplify the presentation, when we apply Lemma 2, we will increase or decrease only some values of a machine, not all its values. The understanding will be that *the rest of the values increase or decrease appropriately by a tiny amount which we omit* to keep the expressions simple.

The second lemma is a useful 2-dimensional property of truthful mechanisms. Fix all values of tasks t except for the values t_{ij} and t_{ik}. A truthful mechanism partitions the two dimensional orthant of $(t_{ij}, t_{ik}) \in \mathbb{R}^2_+$ into 4 regions

$$R_{ab} = \{(t_{ij}, t_{ik}) \,:\, \text{the mechanism allocation has } x_{1j}(t) = a \text{ and } x_{1k}(t) = b\}.$$

The following lemma says that the regions have a particular shape:

Lemma 3. *Every region R_{ab} is bounded by a convex polygon and is separated from region $R_{a'b'}$ by the line*

$$(a - a')t_{ij} + (b - b')t_{ik} = k_{ab:a'b'},$$

where $k_{ab:a'b'}$ is constant (it may however depend on the other values of t except t_{1j} and t_{1k}).

Proof. By the Monotonicity Property, for every $(t_{ij}, t_{ik}) \in R_{ab}$ and $(t'_{ij}, t'_{ik}) \in R_{a'b'}$ we must have $(a-a')(t_{ij}-t'_{ij})+(b-b')(t_{ik}-t'_{ik}) \leq 0$. Equivalently we can write $(a-a')t_{ij}+(b-b')t_{ik} \leq (a - a')t'_{ij} + (b - b')t'_{ik}$. Let now

$$k_{ab:a'b'} = \inf_{(t'_{ij}, t'_{ik}) \in R_{a'b'}} \{(a - a')t'_{ij} + (b - b')t'_{ik}\}$$

and the lemma follows. □

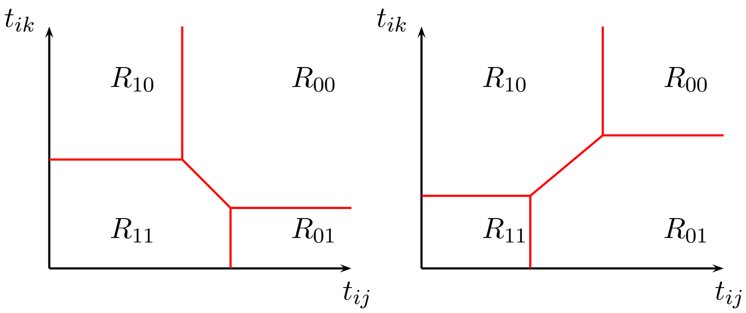

Figure 1: The two possible ways to partition the positive orthant.

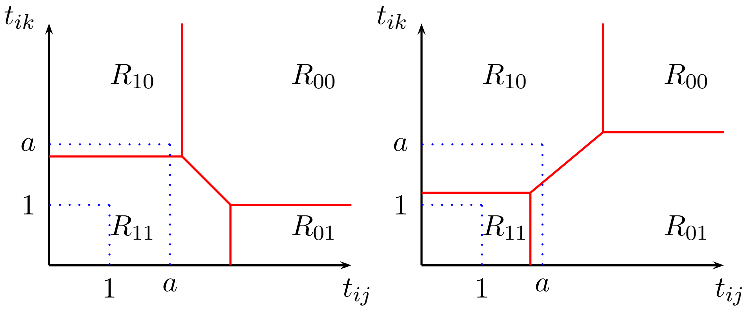

Figure 2: Lemma 4.

Figure 1 depicts the two possibilities of how the positive orthant is partitioned into the four regions (the slope of the inclined parts is $\pm 45\%$). Notice that the lemma does not specify what happens exactly at the boundaries between regions, but this is inconsequential. A useful property that we can extract from the above lemma, and which is going to play an important role in the proof of our main result, is the following:

Lemma 4. *Fix all values of m tasks except of the values t_{ij} and t_{ik}. Assume that a truthful mechanism assigns both tasks to machine i when $(t_{ij}, t_{ik}) = (1, 0)$ and when $(t_{ij}, t_{ik}) = (0, 1)$. Assume also that the mechanism assigns exactly one of the 2 tasks to machine i when $(t_{ij}, t_{ik}) = (a, a)$ for some $a > 1$. Then the mechanism assigns both tasks to machine i when $(t_{ij}, t_{ik}) = (1, 1)$.*

The proof is a simple case analysis and it is essentially shown in Figure 2.

5 The proof of the main result

We will employ instances with 3 machines and 5 tasks. We will assume throughout that the allocation algorithm does not allocate these values (otherwise the mechanism has arbitrarily high approximation ratio).

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

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

where $a > 1$ is a parameter which will be fixed later. This set of tasks has enough symmetries so that it essentially admits two distinct allocations (up to symmetry). For each allocation, we increase or decrease some values appropriately. With the help of the lemmas of the previous section, we show (in Lemma 6 below) that in order to keep the approximation ratio low, the following set of tasks must have the allocation indicated by the stars (in which the first machine gets both tasks 4 and 5):

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

This is sufficient to obtain the lower bound as we will see later.

Lemma 5. *For the instance*

$$\begin{pmatrix} 0 & \infty & \infty & 0 & 1 \\ \infty & 0 & \infty & a & a \\ \infty & \infty & 0 & a & a \end{pmatrix}$$

if the first machine does not get both tasks 4 and 5, then the approximation ratio of the algorithm is at least $1 + a$.

Proof. Suppose that the premises of the lemma hold. As a result, one of machines 2 and 3 will get one of tasks 4 and 5. Suppose without loss of generality that machine 2 gets one of tasks 4 and 5. We raise the 0 of the second player and make it 1 and by Lemma 2 its allocation does not change.

That is, if machine 2 gets task 5, we have

$$\begin{pmatrix} 0^* & \infty & \infty & 0 & 1 \\ \infty & 0^* & \infty & a & a^* \\ \infty & \infty & 0^* & a & a \end{pmatrix} \to \begin{pmatrix} 0^* & \infty & \infty & 0 & 1 \\ \infty & 1^* & \infty & a & a^* \\ \infty & \infty & 0^* & a & a \end{pmatrix},$$

whichever the allocation of the 4th task is (that's what is meant by the absence of a star in the 4th column). Similarly, if machine 2 gets task 4, we have

$$\begin{pmatrix} 0^* & \infty & \infty & 0 & 1 \\ \infty & 0^* & \infty & a^* & a \\ \infty & \infty & 0^* & a & a \end{pmatrix} \to \begin{pmatrix} 0^* & \infty & \infty & 0 & 1 \\ \infty & 1^* & \infty & a^* & a \\ \infty & \infty & 0^* & a & a \end{pmatrix}$$

whichever the allocation of the 5th task is. In either case the cost is at least $1 + a$, while the optimal cost is 1 and is achieved by the allocation

$$\begin{pmatrix} 0^* & \infty & \infty & 0^* & 1^* \\ \infty & 1^* & \infty & a & a \\ \infty & \infty & 0^* & a & a \end{pmatrix}.$$

□

By symmetry, the previous lemma holds also for the case when the processing times of the first player is $(0, \infty, \infty, 1, 0)$ instead of $(0, \infty, \infty, 0, 1)$.

Lemma 6. *If a truthful mechanism has approximation ratio less than $1 + a$ then the first machine should get both tasks 4 and 5 of the matrix of processing times*

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

Proof. Consider the matrix of processing times

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

Without loss of generality, the third machine gets none of the tasks 4 and 5. We essentially have two cases.

Case 1: One of machines 1 and 2, suppose without loss of generaloty that this is machine 1, gets both tasks 4 and 5.

$$\begin{pmatrix} 0^* & \infty & \infty & a^* & a^* \\ \infty & 0^* & \infty & a & a \\ \infty & \infty & 0^* & a & a \end{pmatrix}$$

Using Lemma 2, we can lower the values of t_{14} and t_{15} to 1 without changing the allocation. So we have the indicated allocation for the instance

$$\begin{pmatrix} 0^* & \infty & \infty & 1^* & 1^* \\ \infty & 0^* & \infty & a & a \\ \infty & \infty & 0^* & a & a \end{pmatrix}.$$

Case 2: Tasks 4 and 5 are allocated to different machines. Without loss of generality, machine 1 gets task 4 (as shown in the first of the three sets of tasks and allocations below). Recall that in the previous lemma (Lemma 5) we showed that the middle matrix of processing times below must have the allocation shown in order to keep the approximation ratio lower than $1+a$. By symmetry, the same is true for the third matrix of processing times below

$$\begin{pmatrix} 0^* & \infty & \infty & a^* & a \\ \infty & 0^* & \infty & a & a^* \\ \infty & \infty & 0^* & a & a \end{pmatrix}, \begin{pmatrix} 0^* & \infty & \infty & 0^* & 1^* \\ \infty & 0^* & \infty & a & a \\ \infty & \infty & 0^* & a & a \end{pmatrix}, \begin{pmatrix} 0^* & \infty & \infty & 1^* & 0^* \\ \infty & 0^* & \infty & a & a \\ \infty & \infty & 0^* & a & a \end{pmatrix}.$$

This is the point in our proof where we consider the geometry of the mechanism. We use Lemma 4 for $i=1$, $j=4$, and $k=5$. The lemma implies that the following set of tasks have the indicated allocation

$$\begin{pmatrix} 0^* & \infty & \infty & 1^* & 1^* \\ \infty & 0^* & \infty & a & a \\ \infty & \infty & 0^* & a & a \end{pmatrix},$$

which proves the lemma. □

We now have all the necessary ingredients to prove our main theorem.

Theorem 1. *There is no deterministic mechanism for the scheduling problem with 3 or more machines with approximation ratio less than* $1+\sqrt{2}$*.*

Proof. We will prove that the approximation ratio of any truthful algorithm is at least $\min\{1+a, 1+2/a\}$; for $a=\sqrt{2}$, we have $1+a = 1+2/a$ and the approximation ratio is at least $\min\{1+a, 1+2/a\} = 1+\sqrt{2}$.

By Lemma 6, in order to have approximation ratio lower than $1+a$, the allocation of the following matrix of processing times should be as indicated by the stars

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

We can now increase t_{11} to a. By Lemma 2, this does not change the allocation of the first machine. But then for the matrix of processing times and the indicated allocation below

$$\begin{pmatrix} a^* & \infty & \infty & 1^* & 1^* \\ \infty & 0^* & \infty & a & a \\ \infty & \infty & 0^* & a & a \end{pmatrix}$$

the cost is $2+a$, while the optimum cost is a. The approximation ratio is $1+2/a$. Consequently any truthful algorithm has approximation ratio at least $\min\{1+a, 1+2/a\}$.

Of course, if the number of machines is more than 3, the approximation ratio cannot be lower (by setting, for example, all times of the additional machines to ∞). □

6 Conclusions

Our result improves the existing lower bound of a fundamental problem in the area of mechanisms. The improvement from 2 to $1+\sqrt{2}$ may not be large if one takes into account that the upper bound is still n, but the real importance of our result lies in the fact that it is the only improvement on this important question since the original paper of Nisan and Ronen.

The question is whether the approach of the paper can give better results. In our opinion, it is possible to improve the lower bound to a better constant. But in order to close the huge gap between $1+\sqrt{2}$ and n in a substantial way, we need to find more structural properties of truthful mechanisms, or even better, to obtain a useful characterization. The Monotonicity Property may define the class of truthful mechanisms but it is not a very useful characterization. Our work, and especially Lemma 3, may be a starting point in this direction.

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] 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.

[6] 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.

[7] 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.

[8] George Christodoulou, Elias Koutsoupias, and Annamária Kovács. Mechanism design for fractional scheduling on unrelated machines. In *ICALP*, page (to appear), 2007.

[9] George Christodoulou, Elias Koutsoupias, and Angelina Vidali. A lower bound for scheduling mechanisms. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1163–1169, 2007.

[10] 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.

[11] 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.

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

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

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

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

[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 mechanism design for multi-dimensional scheduling via cycle-monotonicity. In *Proceedings 8th ACM Conference on Electronic Commerce (EC)*, 2007.

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

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

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

[23] 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.

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

[25] 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.