

# A lower bound for scheduling mechanisms\*

George Christodoulou<sup>†</sup>    Elias Koutsoupias<sup>‡</sup>    Angelina Vidali<sup>§</sup>

## 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, 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 [16]. 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, we confirm this and improve the lower bound of the approximation ratio to  $1 + \sqrt{2}$ .

The problem is one of the most fundamental scheduling problems [11, 15]. 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. In the mechanism design setting, each machine  $i$  knows its own times (the  $t_{ij}$ 's), but the algorithm does not know them. The algorithm first asks the machines to declare their times  $t_{ij}$  and then proceeds 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 algorithm and a payment algorithm. They 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 he pays to 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

---

\*Supported in part by IST-15964 (AEOLUS) and the Greek GSRT.

<sup>†</sup>Max-Planck-Institut für Informatik, Saarbrücken, Germany. Work was done while at the University of Athens. Email: {gchristo}@mpi-inf.mpg.de

<sup>‡</sup>Department of Informatics, University of Athens Email: {elias}@di.uoa.gr

<sup>§</sup>Department of Informatics, University of Athens Email: {avidali}@di.uoa.gr

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 on truthful *polynomial-time approximation* algorithms. Two typical problems in this class are the problem of combinatorial auction 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 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 [11, 15]. Here we study its mechanism design version and we improve the results of Nisan and Ronen [16, 17], 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  $7/4$ .

Archer and Tardos [4] considered the variant of the problem for related machines. In this

case, for each machine there is a single value (instead of a vector), its speed. They provided a characterization of all truthful algorithms in this class, in terms of a monotonicity condition. Using this characterization, they gave 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, which was then improved by Kovacs [13] to 3-approximation deterministic truthful mechanism.

Much more work has been done in the context of combinatorial auctions (see for example [3, 6, 7, 8, 5, 9] and the references within).

Saks and Yu [18] proved that for convex domains the Monotonicity Property characterizes the class of social choice functions implementable by truthful mechanisms, generalizing results of [10, 14].

## 3 Problem definition

**DEFINITION 3.1.** (THE SCHEDULING PROBLEM) *The input to the scheduling problem is a non-negative 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. We describe the partition 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  $\sum_{j=1}^m x_{ij} = 1$ .*

A mechanism consists of two algorithms, an allocation mechanism and a payment algorithm. However, we are interested only on 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 3.2. (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  $i$ -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,  $\sum_{j=1}^m (x_{ij} - x'_{ij})(t_{ij} - t'_{ij}) \leq 0$ .

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 [17]. Its proof is based on the fact that, with a truthful mechanism, the payment of a player does not depend on his declared values, but only on the allocation and the values declared by the other players.

**PROPOSITION 3.1.** *Every truthful mechanism satisfies the Monotonicity Property.*

The Monotonicity Property states that  $(x_i - x'_i) \cdot (t_i - t'_i) \leq 0$  is a necessary condition for truthfulness. It turns out that it is also sufficient condition [18], 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 [18], 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 main theorem, 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 two lemmas are about any number of machines  $n$ , while the last one is specific for the case of  $n = 3$  machines.

The first lemma will be used repeatedly and is due to Nisan and Ronen [17]. 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 he 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.

**LEMMA 4.1.** *Let  $t$  be a set of tasks and let  $x = x(t)$  be the allocation produced by a truthful mechanism. Suppose that we change only the tasks 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).*

*Proof.* By the Monotonicity Property, we have

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

Every term of this sum is nonnegative (by the premises of the lemma). The only way to satisfy the inequality (with equality) is to set  $x_{ij}(t) = x_{ij}(t')$  for all  $j$ .

To simplify the presentation, when we apply Lemma 4.1, 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 of the values  $t_{1j}$  and  $t_{1k}$ . A truthful mechanism partitions the two dimensional orthant of  $(t_{1j}, t_{1k}) \in \mathbb{R}_+^2$  into 4 regions

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

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

LEMMA 4.2. *Every region  $R_{ab}$  is bounded by a convex polygon and is separated from region  $R_{a'b'}$  by the line*

$$(a - a')t_{1j} + (b - b')t_{1k} = k_{ab:a'b'},$$

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  $(t_{1j}, t_{1k}) \in R_{ab}$  and  $(t'_{1j}, t'_{1k}) \in R_{a'b'}$  we must have  $(a - a')(t_{1j} - t'_{1j}) + (b - b')(t_{1k} - t'_{1k}) \leq 0$ , equivalently we can write  $(a - a')t_{1j} + (b - b')t_{1k} \leq (a - a')t'_{1j} + (b - b')t'_{1k}$ . Let now

$$k_{ab:a'b'} = \min_{(t'_{1j}, t'_{1k}) \in R_{a'b'}} \{(a - a')t'_{1j} + (b - b')t'_{1k}\}$$

and the lemma follows.

Figure 1 depicts the two possibilities of how the positive orthant is partitioned into the four regions. Notice that the lemma does not specify what happens at the boundaries between regions,

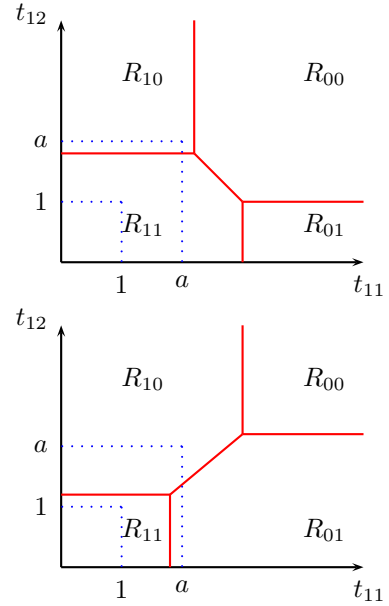


Figure 1: The two possible ways to partition the positive orthant.

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

The proof is a simple case analysis and it is essentially shown in Figure 1.

The last important ingredient in the proof of our main result is also the following lemma, which unlike the above two lemmas is specific for 3 machines.

LEMMA 4.4. *Let  $t = (t_1, t_2, t_3)$  be a set of tasks for 3 machines and let  $x = x(t)$  be the allocation*

of a truthful mechanism. Suppose that the allocation remains the same for all three machines when we change the tasks of machine 2 from  $t_2$  to  $t'_2$  in such a way that  $t'_2 > t_2$  when  $x_{2j} = 0$ , and  $t'_2 < t_2$  when  $x_{2j} = 1$ . Suppose also that the allocation does not change when we change the tasks of machine 3 from  $t_3$  to  $t'_3$  in such a way that  $t'_3 > t_3$  when  $x_{3j} = 0$ , and  $t'_3 < t_3$  when  $x_{3j} = 1$ . Then the allocation does not change when we perform the changes to both machines 2 and 3.

*Proof.* We have  $x = x(t_1, t_2, t_3) = x(t_1, t'_2, t_3)$ . We take  $(t_1, t'_2, t_3)$  and change  $t_3$  to  $t'_3$ . By Lemma 4.1, the allocation of machine 3 remains the same. So we have that  $x_3 = x_3(t_1, t'_2, t'_3)$ . Similarly, we have  $x_2 = x_2(t_1, t'_2, t'_3)$ . But then the allocation of machine 1, which is completely determined by the allocation to the other two machines, remains also the same:  $x_1 = x_1(t_1, t'_2, t'_3)$ .

## 5 The proof of the main result

We will employ instances with 3 machines and 5 tasks. Some values of the tasks are arbitrarily high and we denote them by  $\infty$ . 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

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

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 three lemmas of the previous section, we show (in Lemma 5.3 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):

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

This is sufficient to obtain the lower bound as the next lemma shows.

LEMMA 5.1. *Let  $a > 1$  be a constant. A truthful mechanism which assigns the last two tasks to the first machine of the following instance*

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

*has approximation ratio greater or equal to  $\min\{1 + a, 1 + \frac{2}{a}\}$ .*

*Proof.* We will prove it by contradiction. Assume that there is a mechanism with smaller approximation ratio.

Suppose that we change the set of tasks  $t$  by decreasing  $t_{33}$  to 0. We claim that a mechanism should keep the same allocation. If not, an application of Lemma 4.1 guarantees that only the allocations of the first and second machine can change. The only possibility is that the second machine should get task 4. We then employ again Lemma 4.1 and decrease  $t_{11}$  and  $t_{15}$  to 0. The changes are as follows (where starred values correspond to the allocation):

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

The final allocation has cost  $1 + a$ , while the optimum is 1. The approximation ratio is  $1 + a$ , a contradiction.

So, when we decrease  $t_{33}$  to 0 the allocation remains the same. Similarly, when we decrease  $t_{22}$  to 0. So far, we have established that the following sets of tasks have the indicated allocations:

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

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

Using now Lemma 4.4, we can change both  $t_{22}$  and  $t_{33}$  to 0 and still get the same allocation, as follows:

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

The last step in the proof is to increase  $t_{11}$  to  $a$ . This does not change the allocation of the first machine—the argument is identical to the proof of Lemma 4.1.

But then for the tasks and the allocation below

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

the cost is  $2+a$ , while the optimum cost is  $a$ . The approximation ratio is  $1 + 2/a$ , a contradiction.

We proceed to prove, in Lemma 5.3 below, that the premise of the previous lemma holds. But first, we prove a simple result that will be used in the proof.

LEMMA 5.2. *For the set of tasks*

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

*if the first machine doesn't get both tasks 4 and 5, then the approximation ratio of the mechanism is at least  $1 + a$ .*

*Proof.* Suppose that the premises of the lemma hold. We set the value of  $t_{11}$  to 0. This, by Lemma 4.1, will not affect machine's 1 allocation. As a result, one of machines 2 and 3 will get one of tasks 4 and 5. The cost is at least  $1 + a$ , while the optimal cost is 1.

LEMMA 5.3. *If a truthful mechanism has approximation ratio less than  $1 + a$  then the first machine should get both tasks 4 and 5 of the set of tasks*

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

*Proof.* Consider the set of tasks

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

Without loss of generality, the third machine gets none of the tasks 4 and 5. We now increase  $t_{34}$  to  $\infty$ . By Lemma 4.1, machine 3 still gets only task 3. However, this may change the allocation for the machines 1 and 2. We have two cases.

The first case is easy. Suppose that machine 1 gets both tasks 4 and 5 (similarly we can work for machine 2). Then, using Lemma 4.1, we can lower the values of  $t_{14}$  and  $t_{15}$  to 1 without changing the allocation. So we have the tasks and the indicated allocation

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

We will show that the same allocation for the second case in which each of machines 1 and 2 get one of the tasks 4 and 5. 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 5.2) we showed that the middle set of tasks below must have the allocation shown. By symmetry, the same is true for the third set of

tasks below.

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

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

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

This is the crucial point in our proof. We use Lemma 4.3 for  $(j, k) = (4, 5)$ , and we conclude that the following set of tasks have the indicated allocation

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

To finish the proof, we raise  $t_{25}$  to  $\infty$ . This will not change the allocation of machine 2. If it changes the allocation of machine 3, we set task  $t_{11}$  and  $t_{14}$  to 0. Then machine 1 still doesn't get task 5 and the cost is  $1 + a$ . The optimum cost is 1, a contradiction. Therefore, when we raise  $t_{25}$  to  $\infty$ , both tasks 4 and 5 are still allocated to machine 1. This is what we wanted to prove.

We now have all the necessary ingredients to prove our main theorem. The above lemma shows that the assumptions of Lemma 5.1 holds. This implies an approximation ratio of  $\min\{1 + a, 1 + 2/a\}$ . This is at least  $1 + \sqrt{2}$  when  $a = \sqrt{2}$ .

We generalize the result to mechanisms with more than three machines by setting all processing times of players  $i = 4, \dots, n$  to  $\infty$ .

**THEOREM 5.1.** *There is no deterministic mechanism for the scheduling problem with 3 or more machines with approximation ratio less than  $1 + \sqrt{2}$ .*

## 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 4.2, 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 Confer-*

- ence 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] 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.
- [9] 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.
- [10] Hongwei Gui, Rudolf Müller, and Rakesh V. Vohra. Dominant strategy mechanisms with multidimensional types. In *Computing and Markets*, 2005.
- [11] D.S. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co. Boston, MA, USA, 1996.
- [12] Roberts Kevin. The characterization of implementable choice rules. *Aggregation and Revelation of Preferences*, pages 321–348, 1979.
- [13] Annamaria Kovacs. Fast monotone 3-approximation algorithm for scheduling related machines. In *Algorithms - ESA 2005: 13th Annual European Symposium*, pages 616–627, Spain, 2005.
- [14] 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.
- [15] J.K. Lenstra, D.B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(1):259–271, 1990.
- [16] 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.
- [17] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.
- [18] 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.