HARVARD | BUSINESS | SCHOOL

An Optimal Lower Bound for Anonymous Scheduling Mechanism

Itai Ashlagi
Shahar Dobzinski
Ron Lavi

Working Paper

09-070

Copyright © 2008 by Itai Ashlagi, Shahar Dobzinski, Ron Lavi

Working papers are in draft form. This working paper is distributed for purposes of comment and discussion only. It may not be reproduced without permission of the copyright holder. Copies of working papers are available from the author.

An Optimal Lower Bound for Anonymous Scheduling Mechanisms

Itai Ashlagi* Shahar Dobzinski † Ron Lavi ‡

November 17, 2008

Abstract

We consider the problem of designing truthful mechanisms to minimize the makespan on m unrelated machines. In their seminal paper, Nisan and Ronen [14] showed a lower bound of 2, and an upper bound of m, thus leaving a large gap. They conjectured that their upper bound is tight, but were unable to prove it. Despite many attempts that yield positive results for several special cases, the conjecture is far from being solved: the lower bound was only recently slightly increased to 2.61 [5, 10], while the best upper bound remained unchanged.

In this paper we show the optimal lower bound on truthful anonymous mechanisms: no such mechanism can guarantee an approximation ratio better than m. This is the first concrete evidence to the correctness of the Nisan-Ronen conjecture, especially given that the classic scheduling algorithms are anonymous, and all state-of-the-art mechanisms for special cases of the problem are anonymous as well.

*Harvard Business School, Harvard. Email: iashlagi@hbs.edu.

†The School of Computer Science and Engineering, the Hebrew University of Jerusalem. Supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities, and by a grant from the Israeli Academy of Sciences. Email: shahard@cs.huji.ac.il.

‡IE&M, Technion. Supported by grants from the Israeli Science Foundation and the Binational Science Foundation. Email: ronlavi@ie.technion.ac.il.

1 Introduction

1.1 Background

In their STOC'99 paper, Nisan and Ronen [14] introduced the field of Algorithmic Mechanism Design. In their paper Nisan and Ronen study a problem that takes a central place in the field of algorithmic mechanism design ever since: truthful scheduling on unrelated machines. In this problem, n jobs are to be allocated to m machines, where it takes machine i t_i^j time units to process job j (and this is also the cost machine i incurs from executing job j). The processing times are private information of the machines. The load of machine i is the sum of processing times of the jobs i is assigned. The designer aims to allocate the jobs to minimize the maximal load over all machines (the "makespan").

Nisan and Ronen design a simple mechanism that achieves an approximation ratio of m (the VCG mechanism). They also show that *no* mechanism, even one with unlimited computational power, can guarantee an approximation ratio better than 2. Nisan and Ronen were unable to close this large gap. However, they conjecture that their upper bound is tight:

Conjecture (Nisan-Ronen): No mechanism for the truthful scheduling problem on unrelated machines can achieve an approximation ratio better than m.

A decade later, although widely believed and extensively studied, this conjecture is far from being solved. In fact, despite many efforts by the community, no additional evidence, to either support the conjecture or to weaken the belief in it, has been provided since. Only recently, Christodoulou, Koutsoupias, and Vidali [5] were able to slightly improve the lower bound from 2 to 2.41, further improving it to 2.61 later on [10]. Mu'alem and Schapira [13] and Christodoulou et al. [4] prove a lower bound of 2 for randomized and fractional mechanisms, respectively. Dobzinski and Sundararajan [9] and Christodoulou, Koutsoupias, and Vidali [3] attempted to characterize truthful mechanisms for this problem, but succeeded in doing so only for the very limited case of 2 machines. All these papers involve many non-trivial observations and technicalities, further emphasizing the basic difficulties that this problem entails.

A different line of research attempted to *relax* the problem, by finding tractable special cases. Archer and Tardos [2] observe that if we consider related machines, then the problem becomes single-parameter[1]. Mechanisms for single-parameter problems are easier to design as only monotonicity has to be ensured, a relatively easy task. Indeed, a truthful PTAS that essentially concludes all efforts in this direction was recently constructed [8]. Back in the multi-parameter setting, Lavi and Swamy [12] consider a special case where processing times can take only two possible values, "low" and "high", and give several truthful algorithms that guarantee a constant-factor approximation ratios.

Interestingly, all the known lower bounds are just small constants, and there are no super-constant lower bounds. Furthermore, as mentioned before, several truthful mechanisms that provide good approximation ratios for non-trivial special cases have been presented. At this point, the skeptical reader may start doubting the correctness of the Nisan-Ronen conjecture. Perhaps one should interpret the low values of the upper bounds for the special cases and the previous lower bounds as a signal that a truthful mechanism with a good approximation ratio do exist?

1.2 Our Result

In this paper we give the first strong, concrete evidence to the correctness of the Nisan-Ronen conjecture.

Theorem: No *anonymous* mechanism for the truthful scheduling problem on unrelated machines can achieve an approximation ratio better than m.

An algorithm is anonymous, roughly speaking, if whenever two machines switch costs, the job assignments

[1] A problem is *single parameter* when, informally, the private information of each player consists of only one number.

of the two machines also switch[2]. Note that this is the best lower bound possible as the Nisan-Ronen algorithm is anonymous. Let us explicitly spell out why the class of anonymous mechanisms is of interest:

- **Algorithmic Perspective:** The classic algorithms for scheduling unrelated machines are indeed anonymous. There does not seem to be an *algorithmic* reason that explains why a specific *naming* of the machines can help.
- **Mechanism Design Perspective:** All state-of-the-art mechanisms for the special cases in the literature are indeed anonymous [12, 8]. This might suggest that anonymous mechanisms for this problem are as powerful as non-anonymous mechanisms[3]. In fact, a separation between the power of these two classes will be remarkable.
- **Game-Theoretic Importance:** Not only are anonymous games interesting from a mechanism-design perspective, they are well-studied also from a wider game-theoretic point of view. In particular, anonymous mechanisms are desired from a pure game theoretic point of view as there is no discrimination between the bidders. For further motivation and recent examples of FOCS/STOC papers studying anonymous games see [6, 7][4].

To the very least, our result shows that if the Nisan-Ronen conjecture is false and there are mechanisms that provide a reasonable approximation ratio, then they must be very "strange".

1.3 Tools and Techniques

Our proof shows that the "difficult" instance is actually the most simple instance from an algorithmic point of view: it is the instance in which all costs are between 1 and $1 + \epsilon$, and the cost vectors are ordered so that one cost vector completely dominates the former one, coordinate-wise. We prove that every truthful anonymous mechanism with a finite approximation ratio to the makespan must allocate *all jobs* to the machine with the lowest cost vector. This immediately yields a makespan which is m times the optimum. It is the same difficulty that the VCG mechanism encounters, and we show that all anonymous mechanisms suffer from the same drawback.

To show that we have to develop significant amount of new machinery. Specifically, the proof works inductively, starting from the observation that if we have only a single job, then every anonymous mechanism must allocate it to the lowest-cost machine. Unfortunately, this simple fact is not true when more jobs are added (for mechanisms that do not provide a good approximation ratio). The proof proceeds by following a subtle inductive process that requires substantial amount of technical work, which allows us to consider instances with increasing number of jobs. The approximation property is used in few crucially important places throughout the induction.

As the previous proofs do, we bootstrap the basic difficulty that truthfulness casts: given a specific assignment for one instance, truthfulness implies some restrictions on the possible assignments of all other instances that differ from the original instance by exactly one machine. Nisan and Ronen prove their lower bound by simply considering two such neighboring instances (i.e., a single transition from one instance to another). The other lower bounds that were mentioned above consider slightly longer paths of instances, thus improving the lower bound only slightly. The inductive techniques we develop let us tackle much longer paths of instances. This is the key point that enables us to obtain the optimal lower bound. We believe that this is the main novelty of our proof.

[2]For the mechanism to "notice" that the machines have switched costs, we of course require that the cost vectors of the machines are unique. See the preliminaries section for a formal definition.

[3]Indeed, there is a single example in a different context ("digital goods") where it is proved that anonymous mechanisms are less powerful [1]. However, this setting is a single-parameter one, comparing to our much complicated multi-parameter setting. Thus, one may doubt whether a complicated derandomization process, similar to the one used in [1], might be applicable in our multi-parameter setting. Furthermore, we have no evidence that randomized mechanisms for scheduling are significantly more powerful than deterministic ones.

[4]We note that the definition of anonymity in [6, 7] is slightly different than ours, but very related.

Another important technical reason for our success in proving the lower bound is the way we exploit the *weak monotonicity* property. It is known that every truthful mechanism is also weakly monotone (see the preliminaries for a definition). However, the existing lower bounds prior to our work mainly used a limited and straightforward corollary of weak monotonicity: a machine that declares a vector of costs t and is allocated a bundle S, will be allocated the same bundle S when raising the costs of the jobs not in S and lowering the costs of the jobs in S. We use the weak monotonicity property in more sophisticated ways, by taking into account also the *amount* of which the costs of the jobs changes, whether allocated to the machine or not. This is another novel technical tool that enables us to prove the optimal lower bound. We believe that this tool will turn out to be even more helpful, as it might be used in solving other intriguing problems in algorithmic mechanism design.

1.4 Future Directions

Our proof gives strong evidence that deterministic mechanisms for this central problem do not have much power. Can the anonymity assumption be dropped? Our novel inductive method enables us to reach what seems to be the "correct" difficulty of truthful mechanisms. Thus, we believe that the inductive method and the new technical machinery we introduce might be the basis for further enhancements, to construct lower bounds without the anonymity assumption (though we have not managed to do so yet).

To this end, an interesting observation is that the anonymity property can easily be dropped in the *fractional* case, since, given any truthful approximation mechanism, one can construct an anonymous mechanism that averages over all permutations of players. This keeps the truthfulness and the approximation properties, and inserts anonymity. It might be possible, thus, to extend our proof to the fractional case, assuming anonymity without loss of generality, and obtain a general lower bound using this route.

2 Preliminaries

2.1 Problem Definition and Notation

Let m be the number of machines, and let N, $|N| = n$, be the set of jobs. We denote by $t_i^j \geq 0$ the cost for machine i to process job j (this is also the time it takes machine i to process job j). A cost vector for machine i is a vector $t_i = (t_i^j)_{j=1}^n$. For every bundle $S \subseteq N$ we denote by $t_i(S)$ the total cost for machine i to process the jobs in S. That is, $t_i(S) = \sum_{j \in S} t_i^j$.

Note that an instance of this problem can be viewed as a matrix in which the i^{th} row is the cost vector of machine i, $\overrightarrow{t_i}$. We often use matrices to present instances, and use stars ,"*", to indicate jobs assignments; a star next to the entry t_i^j indicates that machine i is assigned job j. For example, in the following matrix machine i's cost vector is $(t_i^1, \ldots, t_i^n)$ and all the jobs are assigned to machine 1:

$$\begin{pmatrix} t_1^1 * & \cdots & t_1^n * \\ & \vdots & \\ t_{m-1}^1 & \cdots & t_{m-1}^n \\ t_m^1 & \cdots & t_m^n \end{pmatrix}$$

2.2 Truthfulness and Anonymity

A mechanism consists of an allocation function f, and a payment function p_i for every machine i. I.e., for every valuation profile $t = (t_1, \ldots, t_m)$, $f(t)$ is the allocation ($f_i(t)$ is the bundle assigned to machine i), and $p_i(t)$ denotes the payment for machine i when the valuation profile is t. The mechanism is *truthful* if, for every machine i, declaring its true cost vector is at least as good as declaring any other cost vector. Formally, for every machine i, every t_i and every t_{-i},

$$p_i(t_i, t_{-i}) - t_i(f_i(t)) \geq p_i(t_i', t_{-i}) - t_i(f_i(t_i', t_{-i}))$$

where $f_i(t)$ denotes the bundle assigned to machine i in the cost vector $t = (t_1, \ldots, t_m)$. We follow the standard notation: t_{-i} denotes the costs of all machines but i. In particular if $t = (t_1, \ldots, t_m)$ then $t = (t_i, t_{-i})$.

The following is a well-known necessary condition for an allocation function to be *implementable* (i.e., there exist payment functions that make the resulting mechanism truthful):

Definition 2.1 (Weak Monotonicity [11]) *An allocation function f is called* weakly monotone *if for every machine i, and every t_i, t'_i, t_{-i} the following property is satisfied: suppose that $f_i(t_i, t_{-i}) = S_i$ and that $f_i(t'_i, t_{-i}) = S'_i$, then $t_i(S_i) - t_i(S'_i) \leq t'_i(S_i) - t'_i(S'_i)$.*

A direct consequence of weak monotonicity, which we often use in our proof, is the following: if a machine is allocated a bundle S (including the empty set) in some given instance, then it will be allocated the same bundle if it lowers the costs for the jobs in S and raises the costs for all other jobs, while all other machines costs are fixed. Formally,

Claim 2.2 *Let f be a truthful mechanism for the makespan minimization problem. Let $t = (t_1, \ldots, t_m)$ be some set of valuations, and suppose that machine i is allocated some bundle S in $f(t)$. I.e., $f_i(t) = S$. Let t'_i be some other cost vector of machine i where $t'_i(\{j\}) < t_i(\{j\})$ for all $j \in S$, and $t'_i(\{j\}) > t_i(\{j\})$ for all $j \notin S$. Then, $f_i(t'_i, t_{-i}) = S$.*

This paper deals with anonymous mechanisms. Roughly speaking, one expects a mechanism to be anonymous if for every two machines that switch their costs their assigned bundles switch too. There are some technical issues with this definition; for example, if all machines have the same costs, "switching the costs" makes no sense. To handle this and similar problems we only require the following: if machine k has a unique cost vector (a cost vector is said to be *unique* if it differs from every other cost vector in at least one coordinate) and is assigned bundle S, then whenever k switches costs with some other machine l, machine l is allocated the bundle S (we stress that no constraints are imposed on what all other machines are assigned after the switch, in particular machine k, nor on instances where there is no machine with a unique cost vector).

We note that this is only one possible definition of anonymity. This definition was chosen only because it is simple to present. Our proof seems to be general enough to hold using essentially any other reasonable definition of anonymity. In particular, the proof uses the anonymity property only in instances where all machines have unique cost vectors.

Definition 2.3 (An Anonymous Mechanism) *A mechanism f is called* anonymous *if for every m cost vectors, $\overrightarrow{t_1}, \ldots, \overrightarrow{t_m}$, and for each k such that $\overrightarrow{t_k}$ is unique, it holds that for every $k \neq l$:*

$$f_l(\overrightarrow{t}_1, \ldots, \overrightarrow{t}_{k-1}, \overrightarrow{t_l}, \overrightarrow{t_{k+1}} \ldots, \overrightarrow{t_{l-1}}, \overrightarrow{t_k}, \overrightarrow{t_{l+1}}, \ldots, \overrightarrow{t_m}) = f_k(\overrightarrow{t_1}, \ldots, \overrightarrow{t_m})$$

3 The Lower Bound

In this section we prove our main result:

Theorem 3.1 *Let f be a truthful anonymous mechanism for m machines and n jobs that provides a c-approximation to the makespan. Consider an instance in which for every job j, $t^j_m > t^j_{m-1} > \ldots > t^j_1$.*

$$\begin{pmatrix} t^1_1* & t^2_1* & \ldots & t^n_1* \\ t^1_2 & t^2_2 & \ldots & t^n_2 \\ & & \vdots & \\ t^1_{m-1} & t^2_{m-1} & \ldots & t^n_{m-1} \\ t^1_m & t^2_m & \ldots & t^n_m \end{pmatrix}$$

Then, f allocates all jobs to machine 1 (the allocation is indicated by stars). As a corollary, if for each j, $1 + \epsilon > t^j_m > t^j_{m-1} > \ldots > t^j_2 > t^j_1 = 1$ and $m = n$ the approximation ratio of f approaches m as $\epsilon \to 0$.

We prove this theorem by induction on the number of jobs, first considering mechanisms for one job, then gradually increasing the number of jobs. Next we formally define the induction hypothesis.

3.1 The Induction Hypothesis and the Base Case

We will actually prove a stronger version of the main theorem. The stronger version also serves as our induction hypothesis:

The Induction Hypothesis: Fix $\beta > 0$. Let f be a truthful anonymous mechanism for m machines and n jobs that is guaranteed to provide a c-approximation to the makespan, but only to instances with an optimal makespan of at least β. Consider an instance in which for every job j, $t_m^j \geq t_{m-1}^j \geq \ldots \geq t_2^j > t_1^j$, and $t_m^1 > t_{m-1}^1 > \ldots > t_1^1$.

$$\begin{pmatrix} t_1^1* & t_1^2* & \ldots & t_1^n* \\ t_2^1 & t_2^2 & \ldots & t_2^n \\ & & \vdots & \\ t_{m-1}^1 & t_{m-1}^2 & \ldots & t_{m-1}^n \\ t_m^1 & t_m^2 & \ldots & t_m^n \end{pmatrix}$$

Then, f allocates all jobs to machine 1 (the allocation is indicated by stars).

All the instances we consider during the proof will have an optimal makespan larger than some constant β, without specifically specifying that. Observe that given an instance with a small optimal makespan, we can multiply all costs by a large enough number to achieve it. In fact, this will give us Theorem 3.1 under the weaker condition that f is guaranteed to provide a finite approx ratio only for instances with optimal makespan larger than β.

We first prove the induction hypothesis for the base case where there is only one job. In this case the proof is very simple.

Lemma 3.2 *Let f be an anonymous mechanism for one job and m machines. Suppose that $t_1 < t_2 < \ldots < t_m$. Then, machine 1 must be allocated the job in the following instance:*

$$\begin{pmatrix} t_1* \\ \vdots \\ t_{m-1} \\ t_m \end{pmatrix} \tag{1}$$

Proof: Assume towards a contradiction that the job is allocated to machine $i \neq 1$. Now lower machine i's cost to t_1. By weak monotonicity (see Claim 2.2) machine i keeps the job. In particular, machine 1 is not allocated the job. Now raise machine 1's cost to t_i. The job is still not allocated to machine 1, by weak monotonicity. Notice however that machine 1 and i have switched their costs comparing to the instance (1), so by anonymity machine 1 should have received the job. A contradiction. □

3.2 Proving the Induction Step

We now prove the lemma below. We get the induction step by choosing $k = n$.

Lemma 3.3 *Let f be a truthful and anonymous mechanism that provides a c-approximation ratio for m machines and n jobs. Suppose that for every job j, $t_m^j \geq t_{m-1}^j \geq \ldots \geq t_2^j > t_1^j$, and that a is small enough*

constant (to be defined later). For any $k \leq n$, machine 1 is allocated all jobs in the following instance:

$$\begin{pmatrix} t_1^1* & t_1^2* & \dots & t_1^k* & a* & \dots & a* \\ t_2^1 & t_2^2 & \dots & t_2^k & t_2^{k+1} & \dots & t_2^n \\ & & & \vdots & & & \\ t_{m-1}^1 & t_{m-1}^2 & \dots & t_{m-1}^k & t_{m-1}^{k+1} & \dots & t_{m-1}^n \\ t_m^1 & t_m^2 & \dots & t_m^k & t_m^{k+1} & \dots & t_m^n \end{pmatrix}$$

Notice that it is not clear that machine 1 will get even the jobs $k+1, \dots, n$ for which its cost is a, although allocating these jobs to machine 1 seems like the "right" thing to do, since a is very small. For example, allocating jobs $1, \dots, k$ to machine 1 and the rest of the jobs to machines $2, \dots, m$ might also provide a good approximation. In particular, recall that we cannot assume that these jobs are allocated to machine 1 "without loss of generality": a priori it might be the case that such weird allocations are necessary to construct a truthful mechanism that provides a good approximation ratio.

The proof of the lemma is by induction on k. The case of $k = 0$ follows immediately since f provides a finite approximation ratio. Next we assume the lemma is correct for every $l < k$, and prove it for k. Before proving the lemma, we provide an informal overview of the proof.

An Informal Overview of the Proof of Lemma 3.3

The proof of Lemma 3.3 is, yet again, inductive. Notice that the base case of $k = 0$ follows immediately using the fact that f provides a finite approximation ratio, without referring to incentives arguments at all. Below we give an overview of how to prove the induction step for an arbitrary k.

$$\begin{pmatrix} t_1^1* & t_1^2* & \dots & t_1^k* & a & \dots & a \\ t_2^1 & t_2^2 & \dots & t_2^k & a & \dots & a \\ & & & & \vdots & & \\ t_{m-1}^1 & t_{m-1}^2 & \dots & t_{m-1}^k & a & \dots & a \\ t_m^1 & t_m^2 & \dots & t_m^k & a & \dots & a \end{pmatrix}$$

Step 1: We show that if the mechanism was "restricted" only to the first k jobs, then machine 1 was allocated all of them. Using this we show that even in the unrestricted mechanism machine 1 is still allocated the first k jobs (but not necessarily jobs $k+1, \dots, n$).

$$\begin{pmatrix} t_1^1 - \epsilon* & t_1^2 - \epsilon* & \dots & t_1^k - \epsilon* & \delta* & \dots & \delta* \\ t_2^1 & t_2^2 & \dots & t_2^k & a & \dots & a \\ & & & & \vdots & & \\ t_{m-1}^1 & t_{m-1}^2 & \dots & t_{m-1}^k & a & \dots & a \\ t_m^1 & t_m^2 & \dots & t_m^k & a & \dots & a \end{pmatrix}$$

Step 2: Here we verify that machine 1 is allocated all jobs. We do that by reducing the costs of the first k jobs by more than a, and reducing the costs of jobs $k+1, \dots, n$ from a to a much smaller δ. Weak monotonicity and approximation arguments guarantee that machine 1 is allocated all jobs.

$$\begin{pmatrix} t_1^1* & t_1^2* & \dots & t_1^k* & \delta* & \dots & \delta* \\ & & & & \vdots & & \\ t_{i-1}^1 & t_{i-1}^2 & \dots & t_{i-1}^k & a & \dots & a \\ t_i^1 & t_i^2 & \dots & t_i^k & t_i^{k+1} & \dots & t_i^n \\ t_{i+1}^1 & t_{i+1}^2 & \dots & t_{i+1}^k & t_{i+1}^{k+1} & \dots & t_{i+1}^n \\ & & & & \vdots & & \\ t_m^1 & t_m^2 & \dots & t_m^k & t_m^{k+1} & \dots & t_m^n \end{pmatrix}$$

Step 3: This is the most technically involved step. We raise the costs of machines 2 to m one by one, starting from machine m, and show that machine 1 is still allocated all jobs during the process.

Step 1 – Adding Small Jobs

This step shows that "adding" small jobs that all machines value with the same very low cost does not change the allocation of the first k jobs.

Lemma 3.4 *Let f be a truthful and anonymous mechanism that provides a finite approximation ratio for m machines and n jobs. Suppose that for every job j, $t_m^j \geq t_{m-1}^j \geq \ldots \geq t_2^j > t_1^j$, and that a is small enough. Then, jobs $1, \ldots, k$ are allocated to machine 1 in the instance below (no claim is made regarding the allocation of jobs $k+1, \ldots, n$):*

$$\begin{pmatrix} t_1^1* & t_1^2* & \dots & t_1^k* & a & \dots & a \\ t_2^1 & t_2^2 & \dots & t_2^k & a & \dots & a \\ & & & & \vdots & & \\ t_{m-1}^1 & t_{m-1}^2 & \dots & t_{m-1}^k & a & \dots & a \\ t_m^1 & t_m^2 & \dots & t_m^k & a & \dots & a \end{pmatrix} \tag{2}$$

Proof: Consider the following allocation function f^k on m machines and k jobs, which is defined as follows: given a cost vector $\overrightarrow{t_i}$ of k jobs for machine i define the cost vector t_i': $t_i'(\{j\}) = t_i(\{j\})$ for $j \leq k$, and $t_i'(\{j\}) = a$ otherwise. Let $(S_1, \ldots, S_m) = f(t_1', \ldots, t_m')$. Set $f^k(t_1, \ldots, t_n) = (S_1 \setminus \{k+1, \ldots, n\}, \ldots, S_m \setminus \{k+1, \ldots, n\})$. In other words, f^k "runs" f when the cost of each of the last $n-k$ jobs is a, and allocates its k jobs the same way f allocates its first k jobs.

Proposition 3.5 *The allocation function f^k is implementable.*

Proof: Recall the following characterization of truthful mechanisms:

Theorem 3.6 (Saks and Yu [15]) *Let f be an allocation function defined on a convex domain of valuations. f is truthfully implementable if and only if it is weakly monotone.*

By this theorem (which is applicable since the scheduling domain is convex), all we have to prove is that f^k satisfies weak monotonicity. Fix some cost vectors $\overrightarrow{t_1}, \ldots, \overrightarrow{t_{i-1}}, \overrightarrow{t_i}, \ldots \overrightarrow{t_m}$ on n jobs of all machines but machine i, where the cost of each machine i for jobs $k+1, \ldots, n$ is a. Suppose machine i is assigned S_i when declaring t_i and S_i' when declaring t_i' in f (the cost of machine i in t_i' for jobs $k+1, \ldots, n$ is still a). The function f is implementable and thus satisfies weak monotonicity. Hence,

$$t_i(S_i) - t_i(S_i') \leq t_i'(S_i) - t_i'(S_i').$$

Therefore, since machine i did not change its costs of jobs $k+1, \ldots, n$, we also have that

$$t_i(S_i \setminus \{k+1, \ldots, n\}) - t_i(S_i' \setminus \{k+1, \ldots, n\}) \leq t_i'(S_i \setminus \{k+1, \ldots, n\}) - t_i'(S_i' \setminus \{k+1, \ldots, n\})$$

which implies that f^k also satisfies weak monotonicity. □

By construction f and f^k allocate jobs $j = 1, \ldots, k$ identically. Therefore, it is enough to show that f^k assigns jobs $j = 1, \ldots, k$ to machine 1. Towards this end, observe that f^k is anonymous, since the costs of all machines for jobs $k+1, \ldots, n$ is a and f is anonymous. Since a is very small, jobs $k+1, \ldots, n$ have very little affect on the optimal makespan. Therefore, since f provides a finite approximation ratio, f^k provides a finite approximation ratio too, at least for instances with optimal makespan much bigger than $(n-k) \cdot a$ (notice that the instance restricted to jobs $1, \ldots, k$ falls into this category). Hence, by the induction hypothesis machine 1 is allocated jobs $1, \ldots, k$[5]. □

Step 2 – Lowering the Cost of Machine 1

The massaging process of our instance will be much easier by guaranteing that all jobs, not just jobs $1, \ldots, k$, are allocated to machine 1.

Lemma 3.7 *Let f be a truthful and anonymous mechanism that provides a finite approximation ratio for m machines and n jobs. Suppose that for every job j, $t_m^j \geq t_{m-1}^j \geq \ldots \geq t_2^j > t_1^j$, $t_2^j - t_1^j >> \epsilon >> a >> \delta > 0$. All jobs are allocated to machine 1 in the following instance:*

$$\begin{pmatrix} t_1^1 - \epsilon* & t_1^2 - \epsilon* & \ldots & t_1^k - \epsilon* & \delta* & \ldots & \delta* \\ t_2^1 & t_2^2 & \ldots & t_2^k & a & \ldots & a \\ & & & & \vdots & & \\ t_{m-1}^1 & t_{m-1}^2 & \ldots & t_{m-1}^k & a & \ldots & a \\ t_m^1 & t_m^2 & \ldots & t_m^k & a & \ldots & a \end{pmatrix} \tag{3}$$

Proof: Notice that only machine 1 changed its costs comparing to instance (2). To see that in the instance (3) machine 1 gets all jobs we use two claims. Both weak monotonicity arguments and the assumption that f provides a finite approximation ratio are used to prove this claim.

Claim 3.8 *Machine 1 is allocated at least jobs $1, \ldots, k$ in the instance (3).*

Proof: We have shown that in the instance (2) machine 1 gets some bundle S where S contains all jobs $1, \ldots, k$, and possibly some more jobs. By our choice of ϵ, the cost of every job in S decreased more than the cost of all jobs $k+1, \ldots, n$ together. Thus, by weak monotonicity machine 1 is allocated in the instance (3) some bundle that includes all jobs $1, \ldots, k$. □

Claim 3.9 *Machine 1 is allocated all jobs $1, \ldots, n$ in the instance (3).*

Proof: Suppose not. Therefore, by the previous claim, machine 1 is allocated all jobs $1, \ldots, k$ but is not allocated some jobs from $k+1, \ldots, n$:

$$\begin{pmatrix} t_1^1 - \epsilon* & t_1^2 - \epsilon* & \ldots & t_1^k - \epsilon* & \delta & \ldots & \delta* \\ t_2^1 & t_2^2 & \ldots & t_2^k & a & \ldots & a \\ & & & & \vdots & & \\ t_{m-1}^1 & t_{m-1}^2 & \ldots & t_{m-1}^k & a & \ldots & a \\ t_m^1 & t_m^2 & \ldots & t_m^k & a & \ldots & a \end{pmatrix} \tag{4}$$

[5]Here is where we need the restriction on the value of the optimal makespan. In particular, $\beta < n \cdot a$, as otherwise the optimal makespan might be dominated by the "a" jobs, and in particular we will not be able to require that the induced mechanism will provide a finite approximation ratio on jobs $1, \ldots, k$.

Consider the following change in machine 1's costs: the costs of all jobs that it receives in the instance (4) are lowered to $\frac{\delta}{2}$, and the costs of the jobs that it did not receive are raised to 2δ:

$$\begin{pmatrix} \frac{\delta}{2}* & \frac{\delta}{2}* & \dots & \frac{\delta}{2}* & 2\delta & \dots & \frac{\delta}{2}* \\ t_2^1 & t_2^2 & \dots & t_2^k & a & \dots & a \\ & & & & \vdots & & \\ t_{m-1}^1 & t_{m-1}^2 & \dots & t_{m-1}^k & a & \dots & a \\ t_m^1 & t_m^2 & \dots & t_m^k & a & \dots & a \end{pmatrix}$$

By weak monotonicity the allocation of machine 1 stays the same. However, the mechanism do not provide a good approximation ratio: the optimal makespan is less than $2n\delta$ (allocate all jobs to machine 1), while the makespan provided here is at least $a >> 2n\delta$. In particular, choosing $\delta = \frac{a}{200nc}$, will give us that the approximation ratio the mechanism provides on this instance is $100c$, and not c. A contradiction. □

□

Step 3 – Raising the Costs of Machines 2 to m

In this step we raise machine i's costs of every job j, $k+1 \leq j \leq n$ from a to t_i^j (starting with the instance (3)), and show that machine 1 still receives all jobs. We start by raising the cost of machine m, then machine $m-1$ and so on. The definition of the the family I_i aims to capture our state after raising the cost of machine i.

Definition 3.10 *For each $i \geq 2$, an instance U belongs to the family I_i if there exists t_i^j's such that for each j, $t_m^j \geq t_{m-1}^j \geq \ldots \geq t_{i+1}^j \geq t_i^j > t_{i-1}^j \geq t_{i-2}^j \geq \cdots \geq t_2^j > t_1^j$, $t_2^j - t_1^j >> a >> \delta > 0$, and U can be represented in the following form:*

$$\begin{pmatrix} t_1^1* & t_1^2* & \dots & t_1^k* & \delta* & \dots & \delta* \\ t_2^1 & t_2^2 & \dots & t_2^k & a & \dots & a \\ & & & & \vdots & & \\ t_{i-1}^1 & t_{i-1}^2 & \dots & t_{i-1}^k & a & \dots & a \\ t_i^1 & t_i^2 & \dots & t_i^k & t_i^{k+1} & \dots & t_i^n \\ t_{i+1}^1 & t_{i+1}^2 & \dots & t_{i+1}^k & t_{i+1}^{k+1} & \dots & t_{i+1}^n \\ & & & & \vdots & & \\ t_m^1 & t_m^2 & \dots & t_m^k & t_m^{k+1} & \dots & t_m^n \end{pmatrix} \tag{5}$$

Notice that the definition requires at least two strict inequalities for each job j: $t_i^j > t_{i-1}^j$ and $t_2^j > t_1^j$.

Lemma 3.11 *In every instance that belongs to the family I_i, for $i \geq 2$, machine 1 is allocated all jobs.*

Notice that proving that machine 1 is allocated all jobs in instances that belong to I_2 will immediately give us Lemma 3.3 for this value of k.

Proof: The proof of the lemma will make use of the following claims:

Claim 3.12 *In every instance that belongs to the family I_i, machine 1 is either allocated all jobs $1, \ldots, k$ or none of them.*

Proof: Suppose that machine 1 is allocated bundle T that includes some of the jobs $1, \ldots, k$ (but not all of them). Raise the cost of each job $j \notin \{1, \ldots, k\} \setminus T$ to some t'^j_1, so that for each $j \in T$ we still have that $t'^j_1 < t_2^j$. Lower the cost of the rest of the jobs in T to δ, and keep the costs of the other jobs the same. By weak monotonicity the allocation of machine 1 stays the same. However, by induction Lemma

3.3 is already proved for smaller values of k, and this lemma says that if the cost of more than $n-k$ jobs to machine 1 is δ, then machine 1 is allocated all jobs[6]. This is a contradiction and the claim follows. □

Claim 3.13 *In every instance that belongs to the family I_i, if machine 1 is allocated jobs $1,\ldots,k$ then it is allocated all jobs.*

Proof: The proof is similar to the proof of Claim 3.9. Suppose towards a contradiction that machine 1 is not allocated some set T of jobs $k+1,\ldots,n$ (and is allocated the rest of the jobs). Lower the costs of all jobs that machine 1 received to $\frac{\delta}{2}$, and raise the cost of all jobs in T to 2δ. Notice that the allocation to machine 1 stays the same, in particular machine 1 does not get the jobs in T. However, the approximation ratio provided is poor: the optimal solution allocates all jobs to machine 1 and has a makespan of less than $2\delta n$, while the allocation provided by the mechanism has a makespan of at least $a >> 2\delta n$. □

By the previous two claims, all we have to examine is the case where machine 1 is allocated none of the jobs $1,\ldots,k$. We start with a subcase, assuming machines $i,\ldots,m$ do not receive jobs in every instance that belong to I_i.

Claim 3.14 *If machines $i,\ldots,m$ do not receive jobs in every instance that belong to I_i then machine 1 is allocated all jobs (for $i \geq 2$).*

Proof: Consider the following process: raise the cost of machine 1 for each job j by some small $\gamma > 0$, ensuring that the cost of machine 2 for each job is still higher. In particular, γ should be such that $a >> \gamma >> \delta$. By weak monotonicity machine 1 does not receive any of the jobs $1,\ldots,k$. Since there are infinitely many valid choices of γ, there must exist two values of γ, γ_1 and γ_2, $\gamma_2 > \gamma_1$, such that the same machine r receives some of the jobs $1,\ldots k$. Notice that the instance still belongs to I_i, so by the conditions of the claim $i-1 \geq r > 1$.

Now suppose that machine 1 adds a cost of γ_2 for each of its jobs. Lower machine r costs of each job j in $1,\ldots,k$ to $t_1^j + \gamma_1$, and the cost of the rest of the jobs to $\delta + \gamma_1$. Machine r received some of the jobs $1,\ldots,k$ before this change. So we argue that by weak monotonicity it must get some of the jobs $1,\ldots,k$ after the change in the costs. This follows by our choice of a: the cost of *every* job $1,\ldots,k$ has decreased more that the sum of the costs of jobs $k+1\ldots,n$ together, so the cost of machine r for the set of jobs it held before has decreased more that its cost of any bundle that contains only jobs from $k+1,\ldots,n$, so machine r will not be allocated (after the change in the costs) bundles that contains jobs from $k+1,\ldots,n$ only, by weak monotonicity. Observe that this instance belongs to the family I_i, and thus, by Claims 3.12 and 3.13 machine r is allocated all jobs. In particular, machine 1 is allocated no jobs. Now raise the costs of machine 1 for every job j to the cost of machine r before the change in the cost. By weak monotonicity machine 1 is still assigned no jobs. However, in the instance where machine 1 raised its cost by γ_1 machine r was allocated some jobs, while in the current instance machines r and 1 have switched costs; by the anonymity of the mechanism, the machines should have switched allocations too. This is a contradiction, and the claim follows. □

Now we are ready to prove Lemma 3.11. The proof is by induction, from m to 1. We start by considering I_m. Notice that the only difference between the instance (3) and an instance that belongs to I_m is the bigger costs of machine m. Hence, by weak monotonicity we have that machine m is allocated no jobs at all in every instance that belong to I_m. This proves that the conditions of Lemma 3.14 hold for the family of instances I_m, so machine 1 is allocated all jobs in every instance that belong to I_m.

We now prove the induction step. We first prove that the machine with the l'th-highest costs in instances that belong to I_l does not receive any jobs. Consider machine l raising its costs, i.e., consider

[6]Formally speaking, we have proved Lemma 3.3 for smaller values of k only for the case where the value of the right-most jobs is δ, and the value of the rest of the jobs is more than δ, while here we use the lemma for instances where this is not the case. Notice, however, that the numbering (or positioning) of the jobs was not used during the proof, hence the Lemma holds for every possible numbering of the jobs.

an instance in the family I_l. Similarly to before, machine l did not receive any jobs in I_{l+1} and now have higher costs, so by weak monotonicity it is not receiving any jobs in instances that belongs to I_l.

It remains to show that machine w, $w > l$ does not receive jobs in I_l. Suppose it does. Now lower the costs of machine w for each job j from t_w^j to some t'^j_w, where $t_l^j > t'^j_w \geq t_{l-1}^j$ (such t'^j_w exists since $t_l^j > t_{l-1}^j$ by the conditions of the lemma). Machine w received some jobs before, so weak monotonicity implies that after reducing its costs of all jobs it must receive at least one job now. Notice that machine w has the l'th-highest costs in the new instance. However, this instance belongs to the family I_l, and we have showed in the previous paragraph that the machine with the l'th-highest costs receives no jobs. A contradiction. This finishes the proof of Lemma 3.11. □

Acknowledgments

We thank Michal Feldman, Amos Fiat, Haim Kaplan, and Svetlana Olonetsky for helpful discussions.

References

[1] Gagan Aggarwal, Amos Fiat, Andrew V. Goldberg, Jason D. Hartline, Nicole Immorlica, and Madhu Sudan. Derandomization of auctions. In STOC'05.

[2] Aaron Archer and Éva Tardos. Truthful mechanisms for one-parameter agents. In *FOCS'01*.

[3] G. Christodoulou, E. Koutsoupias, and A. Vidali. A characterization of 2-player mechanisms for scheduling. In *ESA*, September 2008.

[4] George Christodoulou, Elias Koutsoupias, and Annamária Kovács. Mechanism design for fractional scheduling on unrelated machines. In *ICALP'07*.

[5] George Christodoulou, Elias Koutsoupias, and Angelina Vidali. A lower bound for scheduling mechanisms. In *SODA '07*.

[6] Constantinos Daskalakis and Christos H. Papadimitriou. Computing equilibria in anonymous games. In *FOCS*, pages 83–93, 2007.

[7] Constantinos Daskalakis and Christos H. Papadimitriou. Discretized multinomial distributions and nash equilibria in anonymous games. In *FOCS*, 2008.

[8] Peerapong Dhangwatnotai, Shahar Dobzinski, Shaddin Dughmi, and Tim Roughgarden. Truthful approximation schemes for single-parameter agents. In FOCS'08.

[9] Shahar Dobzinski and Mukund Sundararajan. On characterizations of truthful mechanisms for combinatorial auctions and scheduling. In EC'08.

[10] Elias Koutsoupias and Angelina Vidali. A lower bound of 1+phi for truthful scheduling mechanisms. In MFCS'07.

[11] Ron Lavi, Ahuva Mu'alem, and Noam Nisan. Towards a characterization of truthful combinatorial auctions. In *FOCS'03*.

[12] Ron Lavi and Chaitanya Swamy. Truthful mechanism design for multi-dimensional scheduling via cycle monotonicity. In EC'07.

[13] Ahuva Mu'alem and Michael Schapira. Setting lower bounds on truthfulness: extended abstract. In *SODA*, 2007.

[14] Noam Nisan and Amir Ronen. Algorithmic mechanism design. In *STOC*, 1999.

[15] M. E. Saks and L. Yu. Weak monotonicity suffices for truthfulness on convex domains. In *Proceedings of the 6th ACM conference on Electronic commerce*, pages 286–293, 2005.