An Optimal Lower Bound for Anonymous Scheduling Mechanisms*

Itai Ashlagi
Sloan School of Management
MIT
iashlagi@MIT.EDU

Shahar Dobzinski
Computer Science Department
Cornell University
shahar@cs.cornell.edu

Ron Lavi†
Faculty of Industrial Engineering and Management
The Technion – Israel Institute of Technology
ronlavi@ie.technion.ac.il

Abstract

We consider the problem of designing truthful mechanisms to minimize the makespan on m unrelated machines. In their seminal paper, Nisan and Ronen [13] 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 [6, 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.

*Preliminary version appeared in EC'09.

†Supported by grants from the Israeli Science Foundation, the Binational Science Foundation, and the Israeli ministry of Science.

1 Introduction

1.1 Background and Motivation

Nisan and Ronen [13] introduced the field of "Algorithmic Mechanism Design" by studying the problem of truthful scheduling on unrelated machines, a problem that takes a central place in this field ever since. The problem of job scheduling on unrelated machines is classic in the disciplines of CS and OR. In this problem, a designer needs to assign n jobs to m machines, where it takes machine i t_i^j time units to process job j. If machine i is assigned a set S_i of jobs it needs $\sum_{j \in S_i} t_i^j$ time units in order to complete its assignment. This is termed the "load" of the machine. A popular goal is to find an assignment that minimizes the *makespan* – the maximal load over all machines. This corresponds to the time where the processing of all jobs will be completed.

The paper [13] adds a game-theoretic viewpoint to this model, in the spirit of mechanism design: the machines are viewed as selfish "workers", and the processing times of each machine/worker are private information to that machine. In addition, each machine incurs a fixed cost of say one dollar for each time unit it devotes to processing its given tasks. The utility of each machine is quasi-linear – the payment she receives for her work minus the cost incurred to her by performing her assigned work. Before assigning the jobs to the machines, the designer now needs to first extract the processing times from the different machines (considering the possibility that a machine may misreport her true processing times if this will increase her utility), and in addition, determine payments to the machines in order to compensate them for their efforts.

An easy observation made by [13] is that the well-known VCG mechanism guarantees a makespan of at most m times the optimal makespan, while making it the dominant strategy of each machine to report its true type.[1] Following the OR/CS literature this m factor is called the approximation ratio of the mechanism. Clearly, an m approximation ratio is a very large ratio. A main technical effort of Nisan and Ronen was to show that nothing better is possible. They succeeded in this only very partially, showing that no truthful deterministic mechanism[2] can obtain an approximation ratio better than 2, i.e. every truthful mechanism must sometimes result in an assignment with makespan at least twice the optimal makespan. Nisan and Ronen were unable to close the large gap between 2 and m, but conjectured that the upper bound is tight:

Conjecture (Nisan-Ronen): No truthful deterministic mechanism for unrelated job scheduling can achieve an approximation ratio better than m.

This problem gained increasing importance in the algorithmic-game-theory community over the past decade, for three main reasons:

[1] The VCG mechanism, in our context, assigns each job to the machine i that has the lowest cost for this job, and makes a payment equals to the second lowest cost of this job to machine i. One can verify that, in this mechanism, and regardless of the reports of the other machines, machine i will maximize its utility by reporting her true cost vector. Equivalently, truthfulness is a dominant strategy.

[2] A truthful mechanism is a direct mechanism in which reporting the true type is a dominant strategy.

- **Demonstrates the interaction between game-theory, OR, and CS:** This scheduling problem is fundamental in OR and CS, and captures many real-world situations. The assumption that the machines are selfish entities is a very natural extension of the scheduling model that has been studied for decades. Successful application of game-theoretic tools that will yield scheduling methods that are resistant to selfish behavior may therefore be influential both theoretically and practically.

- **Suggests unconventional goals for mechanism design models:** Mechanism design usually considers two possible design goals – either maximize the "social welfare" or the designer's revenue. While these two goals are indeed very natural, the unrelated job-scheduling problem demonstrates that settings adjusted from different disciplines may yield new goals. In fact, one may view the makespan-minimization goal as a variant of a max-min fairness condition, connecting this mechanism design problem to social choice issues. The problem may also be viewed as a very basic principal-agent model. Regardless of the interpretation, the point is that the study of an implementation goal different than welfare maximization or revenue maximization in a classic incomplete information setting may enrich the set of tools that mechanism design offers.

- **Serves as a tool to study the possibility – impossibility border of dominant-strategy implementability in multi-dimensional settings:** The theory of mechanism design is still vague and unclear about what kinds of dominant-strategy mechanisms other than VCG exist when the domain of players' types is multi-dimensional. This problem is an excellent example, and one may hope that a successful resolution will generalize to other settings as well, or at least yield new methods for the exploration of the general issue.

A decade later, although gaining increasing importance and being extensively studied, the conjecture of Nisan and Ronen 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. Only recently, Christodoulou, Koutsoupias, and Vidali [6] 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 [12] and Christodoulou et al. [4] prove a similar lower bound of $2 - (1/m)$ for randomized and fractional mechanisms, respectively. Dobzinski and Sundararajan [9] and Christodoulou, Koutsoupias, and Vidali [5] 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 find special cases of the problem that can be successfully solved. Archer and Tardos [2] suggested to restrict players' types to be "one-parameter" by studying the model of related machines: the processing time of a job j on machine i is determined by dividing its "basic" processing time (which is now fixed and known) by the "speed" of machine i (which

is the only private parameter of the machine). Mechanisms for single-dimensional problems are easier to design, and indeed this model gives rise to many positive results. Archer and Tardos show that there exists a truthful mechanism that always results in the optimal makespan for this setting. Many other works examined the computational complexity of the various possible mechanisms, with a recent notable truthful PTAS that essentially concludes all computational efforts in this direction in [8] and [7]. Back in the multi-parameter setting, Lavi and Swamy [11] consider a special case where processing times can take only two possible values, "low" and "high", and give several truthful mechanisms with constant-factor approximation ratios. Yu [15] extends this result to a two-range-values variant of the problem.

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 does 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* truthful mechanism for job-scheduling on unrelated machines can achieve an approximation ratio better than m.

A mechanism is anonymous, roughly speaking, if whenever two machines switch costs, the job assignments of the two machines also switch.[3] 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 on 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:** Indeed, it is very easy to come up with non-anonymous mechanisms. However, are non-anonymous mechanisms more powerful than anonymous ones? All state-of-the-art mechanisms for the special cases in the recent literature are anonymous (for example [11, 8]). This might suggest that anonymous mechanisms for this problem are as

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

powerful as non-anonymous mechanisms.[4] 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, anonymity is a compelling design requirement in many contexts as there is no discrimination between the players, and is therefore commonly studied in game theory as a whole.

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 such 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 by taking an instance with m machines and $n = m$ jobs. It is the difficulty that the VCG mechanism encounters, and we show that all anonymous mechanisms suffer from the same drawback.

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 crucially used in the induction, as without it the claim is false (i.e. there exist truthful mechanisms that do not assign all jobs to the machine with the lowest cost vector, but also do not guarantee any finite approximation ratio).

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

[4] 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.

consider longer paths of instances, by this improving the lower bound. The inductive techniques we develop let us tackle much longer paths of instances.[5] 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.

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 previous proofs 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 a broader way, taking into account also the *amount* of which the costs of the jobs changes, whether allocated to the machine or not.

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.

1.5 Paper Organization

The rest of the paper is organized as follows. Section 2 details the definition of the problem, the various notations we use, and the weak monotonicity condition and some of its corollaries. Section 3 describes the main theorem and its proof. Section 4 summarizes and discusses few conceptual issues.

[5] A notable exception is [10] which also uses induction. The induction used here is very different.

2 Preliminaries

2.1 Job Scheduling for Makespan Minimization

We have m machines and n jobs, where $N = \{1, ..., n\}$. Let $t_i^j \geq 0$ denote the time that machine i takes to process job j. A "cost vector" for machine i is a vector $t_i = (t_i^j)_{j=1}^n$. For every subset $S \subseteq N$ we denote by $t_i(S)$ the total time it takes machine i to process the jobs in S. That is, $t_i(S) = \sum_{j \in S} t_i^j$. We sometimes describe an instance of this problem by a matrix in which the i^{th} row is the cost vector of machine i, $\vec{t_i}$. We 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:

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

Let $T = \Re_{\geq 0}^{m \times n}$ be the space of all possible instances, and A be the space of all possible allocations of jobs to machines. For an instance t, $f(t) = S = (S_1, ..., S_m) \in A$, where S_i is the set of jobs allocated to machine i. Our goal is to design an allocation rule $f : T \to A$ that allocates the jobs to the machines in order to minimize the "makespan" of the outputted schedule, defined as $m(S) = \max_{i=1}^n t_i(S_i)$. In words, the makespan of a schedule is the time by which all jobs will be processed, according to the schedule. We compare a given allocation rule f to the optimal allocation rule, OPT, that given any instance t always outputs a schedule with the minimal possible makespan for t. We say that f is a "c-approximation", for some real number $c \geq 1$, if for any $t \in T$ f always outputs a schedule with makespan at most c times the optimal makespan, i.e. $m(f(t)) \leq c \cdot m(OPT(t))$ for every $t \in T$. c is the "approximation ratio" of f.

Notice that the above definitions imply that we study only deterministic allocation rules.

2.2 A Mechanism-Design Setup

Following [13], we study a setup where each machine i is an individual utility-maximizing entity ("worker"), that privately knows her cost vector t_i. We assume that the cost of one time-unit for the worker is one monetary unit, hence the identification between the cost vector and the time vector. The machine may receive a payment to compensate her for the cost of processing her assignment, and the machine's utility is assumed to be quasi-linear: total payment minus total cost.

A designer needs to assign the jobs to the machines in order to minimize the makespan of the schedule, and for this purpose she constructs a direct mechanism.[6] A direct mechanisms consists of

[6]See Section 4 for a discussion on the implications of our result on indirect mechanisms.

an allocation function f, and a payment function $p_i : T \to \Re$ for every machine i. I.e., the players are being asked to report a type, given the reports of the players $t = (t_1, \ldots, t_m)$, the mechanism announces $f(t)$ as the allocation, and pays $p_i(t)$ monetary units to machine i.

A "truthful mechanism" is a direct mechanism in which reporting the true cost vector is a dominant strategy for every player. One well-known such mechanism is the VCG mechanism, which (in this case) assigns each job independently to the machine with the smallest cost for that job.[7] The approximation ratio of VCG is m, the number of machines. To see this, define an "ordered instance" t to be a tuple of types $t_1, ..., t_m$ such that, for every job j, $t_m^j > \ldots > t_2^j > t_1^j$. Thus, if t is an ordered instance, VCG assigns all jobs to machine 1, and if for each j, $1+\epsilon > t_m^j > t_{m-1}^j > \ldots > t_2^j > t_1^j = 1$ and $m = n$ the approximation ratio of f approaches m as $\epsilon \to 0$. This is quite disappointing as m can be arbitrarily bad. In this paper we ask if there exist other truthful mechanisms with a better approximation ratio.

The VCG mechanism is anonymous, in the sense that for every permutation of the reports, the job-assignments permute in the same way. In other words, VCG does not rely on the machines' identities. This paper shows that VCG provides the best approximation ratio among all truthful anonymous mechanisms. In the formal definition of this notion, there is one issue we need to be careful about – the case where two machines have identical cost vectors. In this case, switching their cost vectors will not change the instance, and so it does not make sense to require that their job allocation will switch. Therefore we require anonymity only for instances with "no ties":

Definition 2.1 (No ties) *An instances t is with "no ties" if for every two machines i, i' and every job j, $t_i^j \neq t_{i'}^j$. In other words, in the matrix representation of t, there are no two identical entries in the same column (but there might be identical entries in the same row).*

Definition 2.2 (Anonymity) *An allocation rule f is* anonymous *if for every instance t with no ties, and for every two machines i, i', if machine i receives S_i in $f(t)$ then machine i' receives S_i in $f(\tilde{t})$, where in $\tilde{t}$ the cost vector of machine i is $t_{i'}$, the cost vector of machine i' is t_i, and the rest of the cost vectors are as in t.*

We note that we cast no requirements on instances with ties. We also note that this is a rather weak anonymity requirement since we do not permute all players, and since we do not require that the job allocation of the other players remain the same.

2.3 Monotonicity and Implementability

An allocation rule f is implementable if there exist payment rules $p_1, ..., p_m$ such that the mechanism $M = (f, p_1, ..., p_m)$ is truthful. The following is a well-known necessary condition for an allocation rule to be implementable:

[7] The payment for each job is the second-lowest cost for that job.

Definition 2.3 (Weak Monotonicity; Bikhchandani et al. [3]) *An allocation function f is* weakly monotone *if for every machine i, every $t_{-i} \in T_{-i}$[8] and $t_i, t_i' \in T_i$ 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 $t_i(S_i) - t_i(S_i') \leq t_i'(S_i) - t_i'(S_i')$.*

Bikhchandani et al. [3] show that if f is implementable then f must be weakly monotone. Throughout our proofs we will use this property instead of the property of truthfulness, which will save us the trouble of handling mechanisms and payments. Thus our theorem holds for all weakly monotone allocation rules. Weak monotonicity casts several implications that we use in our proofs:

Claim 2.4 *Suppose that f is weakly monotone, fix $t \in T$, and let $S_i = f_i(t)$ be the set of jobs assigned to machine i by f in t.*

1. *Fix $\tilde{t}_i \in T_i$ that satisfies, $\forall j \in S_i$, $\tilde{t}_i^j < t_i^j$, and $\forall j \in N \setminus S_i$, $\tilde{t}_i^j > t_i^j$. Then $S_i = f_i(\tilde{t}_i, t_{-i})$.*
2. *Fix $J \subseteq N \setminus S_i$, and a real number $\delta > 0$. Fix $\tilde{t}_i \in T_i$ that satisfies, $\forall j \in J$, $\tilde{t}_i^j - t_i^j \geq n\delta$, $\forall j \in S_i$, $\tilde{t}_i^j \leq t_i^j$, and $\forall j \in N \setminus J \setminus S_i$, $t_i^j - \tilde{t}_i^j < \delta$. Let $\tilde{S}_i = f_i(\tilde{t}_i, t_{-i})$. Then $J \cap \tilde{S}_i = \emptyset$.*

Proof:

1. Let $\tilde{S}_i = f_i(\tilde{t}_i, t_{-i})$. Weak monotonicity implies $t_i(S_i) - t_i(\tilde{S}_i) \leq \tilde{t}_i(S_i) - \tilde{t}_i(\tilde{S}_i)$. Therefore $t_i(S_i \setminus \tilde{S}_i) - t_i(\tilde{S}_i \setminus S_i) \leq \tilde{t}_i(S_i \setminus \tilde{S}_i) - \tilde{t}_i(\tilde{S}_i \setminus S_i)$. Since $t_i(S_i \setminus \tilde{S}_i) > \tilde{t}_i(S_i \setminus \tilde{S}_i)$ and $t_i(\tilde{S}_i \setminus S_i) < \tilde{t}_i(\tilde{S}_i \setminus S_i)$ it must follow that $S_i \setminus \tilde{S}_i = \tilde{S}_i \setminus S_i = \emptyset$, implying the claim.

2. Suppose by contradiction that $|J \cap \tilde{S}_i| > 0$. Then

$$\tilde{t}_i(\tilde{S}_i \setminus S_i) - t_i(\tilde{S}_i \setminus S_i) \geq n\delta + \tilde{t}_i(\tilde{S}_i \setminus S_i \setminus J) - t_i(\tilde{S}_i \setminus S_i \setminus J) > 0 \geq \tilde{t}_i(S_i \setminus \tilde{S}_i) - t_i(S_i \setminus \tilde{S}_i).$$

Thus $\tilde{t}_i(\tilde{S}_i) - t_i(\tilde{S}_i) > \tilde{t}_i(S_i) - t_i(S_i)$, which contradicts weak monotonicity.

□

3 The Lower Bound

In this section we prove our main result. Recall that an *ordered instance* t is a tuple of types $t_1, ..., t_m$ such that, for every job j, $t_m^j > \ldots > t_2^j > t_1^j$. We show:

Theorem 3.1 *Let f be a truthful anonymous mechanism for m machines and n jobs that provides a c-approximation to the makespan (for some real number $c \geq 1$). Then, for every ordered instance t, f allocates all jobs to machine 1. As a corollary, if for each j, $1+\epsilon > t_m^j > t_{m-1}^j > \ldots > t_2^j > t_1^j = 1$ and $m = n$ the approximation ratio of f approaches m as $\epsilon \to 0$.*

[8] We follow the standard notation: t_{-i} denotes the costs of all machines but i.

$$\begin{pmatrix} t_1^1 & \cdots & t_1^j & \delta & \cdots & \delta \\ t_2^1 & \cdots & t_2^j & a_2 & \cdots & a_2 \\ \vdots & & \vdots & \vdots & & \vdots \\ t_{i-1}^1 & \cdots & t_{i-1}^j & a_{i-1} & \cdots & a_{i-1} \\ t_i^1 & \cdots & t_i^j & t_i^{j+1} & \cdots & t_i^n \\ \vdots & & \vdots & \vdots & & \vdots \\ t_m^1 & \cdots & t_m^j & t_m^{j+1} & \cdots & t_m^n \end{pmatrix}$$

Figure 1: An illustration of an instance that is a $(\{1,...,j\},i)$-projection of t, as in Def. 3.2

The rest of the section formally proves this theorem. The proof is by induction, using various intermediary cost vectors (instances) that are variations over the original cost vector t. For this purpose we use the following notation:

Definition 3.2 *Fix $J \subseteq N$ and $2 \leq i \leq m+1$. Let $\bar{J} = N \setminus J$. An instance s is a (J,i)-projection of an ordered instance t if there exist m real numbers $a_m > ... > a_2 > \delta > 0$ that satisfy*

$$a_m < \min_{j \in \bar{J}} t_2^j, \quad \delta < \frac{a_2}{2nc}, \quad \delta < \min_{j \in J} \frac{t_2^j - t_1^j}{2n} - a_m$$

such that s has the following structure (see also the illustration in Figure 1):

1. *$s_1^j = t_1^j$ for every $j \in J$, and $s_1^j = \delta$ for every $j \in \bar{J}$.*
2. *For every machine $2 \leq i' \leq i-1$, $s_{i'}^j = t_{i'}^j$ for every $j \in J$, and $s_{i'}^j = a_{i'}$ for every $j \in \bar{J}$.*
3. *For every machine $i \leq i' \leq m$, $s_{i'} = t_{i'}$.*

We prove by induction on $|J|, i$ that in any (J,i)-projection of any ordered instance t, machine 1 must be allocated all jobs. Since an (N,i)-projection of t is t itself (regardless of i), Theorem 3.1 will follow as a result of the inductive hypothesis. Our inductive argument advances over $|J| = 1,..,n$, and, for every fixed J, over $i = m+1,...,2$. Thus, throughout, we fix J, i and assume that the inductive hypothesis is true for any (J',i')-projection of any ordered instance t, where either $|J'| < |J|$ and $2 \leq i' \leq m+1$, or $J' = J$ and $i < i' \leq m+1$. The base of the induction ($|J| = 1, i = m+1$) is proved exactly like the inductive step, as all the claims below are true also for this case.

We divide the proof to three parts. In Section 3.1 we prove three basic claims that will be used in the main argument. In Section 3.2 we construct several specific scenarios and prove their technical properties. Section 3.3 uses these scenarios to prove the inductive hypothesis.

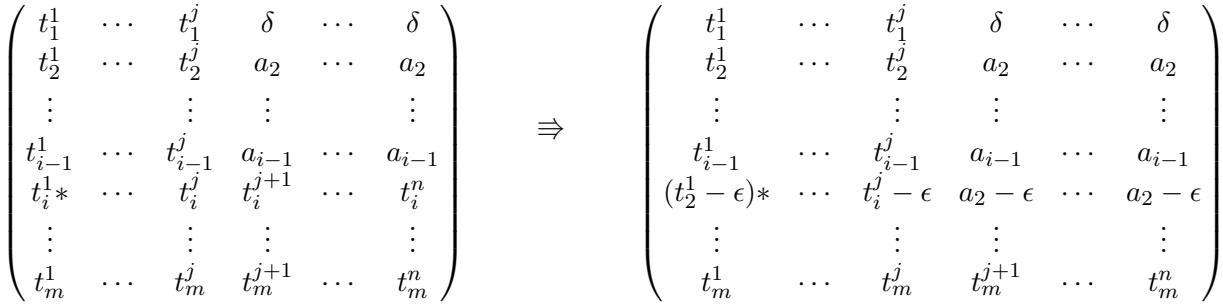

Figure 2: Illustration of the argument in claim 3.3. If there exists an instance similar to the left instance, where machine i receives a job in J, we move to an instance similar to the right instance, which is (shown to be) a $(J, i+1)$-projection of some ordered instance. Weak monotonicity and the transition from the left instance to the right instance imply that machine i must receive a job in J in the right instance, contradicting the inductive assumption.

3.1 Core Arguments

We prove three claims that serve as a starting point to the inductive proof: (1) that machines $i, ..., m$ cannot be allocated any job, (2) that machine 1 either receives all jobs in J or none of them, and (3) that if machine 1 receives all jobs in J then it receives all jobs $1, ..., n$. We note that these claims are true also for $|J| = 1$. Each claim in the sequel is followed by a figure that illustrates the main argument.

Claim 3.3 *In any (J, i)-projection of any ordered instance t, machines $i, ..., m$ do not receive any job (i.e. machines $1, ..., i-1$ receive all jobs).*

Proof: (See illustration in Figure 2). For $i = m + 1$ the claim is immediate, thus assume $i \leq m$. Assume by contradiction that there exists an instance $s = (s_1, ..., s_m)$ which is a (J, i)-projection of some ordered instance $t = (t_1, ..., t_m)$ (with constants $a_m > ... > a_2 > \delta$), and a machine $r \geq i$ such that machine r receives a non-empty set of jobs in instance s. Since all the inequalities in Definition 3.2 are strict, there exists $\epsilon^* > 0$ such that $(s_2 - \epsilon^*, s_{-2})$ is a (J, i)-projection of $(t_2 - \epsilon^*, t_{-2})$ (the subtraction is coordinate-wise). Therefore if we let $\tilde{t}_r = t_2 - \epsilon^*$ and $\tilde{s}_r = s_2 - \epsilon^*$ we get that $\tilde{s} = (\tilde{s}_r, s_{-r})$, i.e. the instance where the cost vector of machine r is $\tilde{s}_r$ and the other cost vectors are as in s, is a $(J, i+1)$-projection of $(\tilde{t}_r, t_{-r})$, with constants $a_{-r}, a_2 - \epsilon^*, \delta$. By the inductive hypothesis machine 1 receives all jobs in $\tilde{s}$. However by weak monotonicity since in the transition from s to $\tilde{s}$ only machine r changed its cost vector, from s_r to $\tilde{s}_r$, and since $\tilde{s}_r$ is strictly smaller than s_r (coordinate-wise), the fact that machine r receives a non-empty set of jobs in s implies that machine r must receive a non-empty set of jobs in $\tilde{s}$, a contradiction. □

$$\begin{pmatrix} t_1^1* & \cdots & t_1^j & \delta & \cdots & \delta \\ t_2^1 & \cdots & t_2^j* & a_2 & \cdots & a_2 \\ \vdots & & \vdots & \vdots & & \vdots \\ t_{i-1}^1 & \cdots & t_{i-1}^j & a_{i-1} & \cdots & a_{i-1} \\ t_i^1 & \cdots & t_i^j & t_i^{j+1} & \cdots & t_i^n \\ \vdots & & \vdots & \vdots & & \vdots \\ t_m^1 & \cdots & t_m^j & t_m^{j+1} & \cdots & t_m^n \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \tilde{\delta}* & \cdots & t_1^j + n\delta & \tilde{\delta} & \cdots & \tilde{\delta} \\ t_2^1 & \cdots & t_2^j & a_2 & \cdots & a_2 \\ \vdots & & \vdots & \vdots & & \vdots \\ t_{i-1}^1 & \cdots & t_{i-1}^j & a_{i-1} & \cdots & a_{i-1} \\ t_i^1 & \cdots & t_i^j & t_i^{j+1} & \cdots & t_i^n \\ \vdots & & \vdots & \vdots & & \vdots \\ t_m^1 & \cdots & t_m^j & t_m^{j+1} & \cdots & t_m^n \end{pmatrix}$$

Figure 3: Illustration of the argument in claim 3.4. If there exists an instance similar to the left instance, where machine 1 receives some of the jobs in J but not all of them (in the figure – job 1 but not job j), we move to an instance similar to the right instance, which is (shown to be) a (J^*, i)-projection of itself for some J^* we define in the proof. Weak monotonicity and the transition from the left instance to the right instance imply that machine 1 cannot receive any job in J^* (in particular, job j in the figure), contradicting the inductive assumption.

Claim 3.4 *Fix some instance s that is a (J, i')-projection of some ordered instance t, for the currently fixed J in the induction argument, and for any $2 \leq i' \leq m+1$.* [9] *If machine 1 receives some of the jobs in J in instance s, then it must receive all jobs in J in the instance s.*

Proof: (See illustration in Figure 3). If $|J| = 1$ the claim is trivially true, so assume $|J| > 1$. Suppose that the mechanism assigns a set S of jobs to machine 1 in the instance s, and assume by contradiction that $J \cap S \neq \emptyset$ and $J \setminus S \neq \emptyset$. Let $a_m > ... > a_2 > \delta$ be the constants from Definition 3.2. Let $J^* = J \setminus S$. Note that $0 < |J^*| < |J|$ and $J^* \subseteq J$. Consider the following cost vector $\tilde{s}_1$ for machine 1, where $\tilde{\delta} < \delta$ will be chosen below such that $\tilde{s} = (\tilde{s}_1, s_{-1})$ will be a $(J^*, 2)$-projection of itself. [10]

$$\tilde{s}_1^j = \begin{cases} t_1^j + n\delta & j \in J^* \\ \tilde{\delta} & \text{otherwise} \end{cases}$$

We choose $\tilde{\delta}$ as follows. Since $\delta < \min_{j \in J} \frac{t_2^j - t_1^j}{2n} - a_2$ then $\min_{j \in J} \frac{t_2^j - (t_1^j + n\delta)}{2n} - a_2 > 0$. Now,

- Fix $\tilde{a}_m < a_2 < \min_{j \in J^*} \frac{t_2^j - (t_1^j + n\delta)}{2n}$, and additional $m-2$ real numbers $\tilde{a}_2 < \tilde{a}_3 < ... < \tilde{a}_m$.
- Define $\tilde{\delta} < \delta$ such that $\tilde{\delta} < \tilde{a}_2/(2nc)$ and $\tilde{\delta} < \min_{j \in J^*} \frac{t_2^j - (t_1^j + n\delta)}{2n} - \tilde{a}_m$.

Note that the constants $\{\tilde{a}_i\}_{i=2}^m$ do not appear in $\tilde{s}$ – we need them only to verify the projection according to Def. 3.2. By construction of these constants, we have verified that $\tilde{s}$ is a $(J^*, 2)$-

[9] Recall that we are inside a proof by induction, i.e. we prove for (J, i) by assuming correctness for "previous" instances. However we emphasize that this claim is stated and proved for *any* (J, i'), i.e. also for $i' < i$ coupled with the set J that is fixed in the current induction step (the set J is not changed in this claim, only the index i).

[10] Notice that by Definition 3.2, an instance t may be a $(J, 2)$-projection of itself (but only for $i = 2$), if $t_1^j = \delta$ for every $j \in \bar{J}$, and if δ is very small relative to the other job costs and relative to the difference $(t_2^j - t_1^j)$.

$$\begin{pmatrix} t_1^1* & \cdots & t_1^j* & \delta* & \cdots & \delta \\ t_2^1 & \cdots & t_2^j* & a_2 & \cdots & a_2* \\ \vdots & & \vdots & \vdots & & \vdots \\ t_{i-1}^1 & \cdots & t_{i-1}^j & a_{i-1} & \cdots & a_{i-1} \\ t_i^1 & \cdots & t_i^j & t_i^{j+1} & \cdots & t_i^n \\ \vdots & & \vdots & \vdots & & \vdots \\ t_m^1 & \cdots & t_m^j & t_m^{j+1} & \cdots & t_m^n \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \delta/2 & \cdots & \delta/2 & \delta/2 & \cdots & 2\delta \\ t_2^1 & \cdots & t_2^j & a_2 & \cdots & a_2 \\ \vdots & & \vdots & \vdots & & \vdots \\ t_{i-1}^1 & \cdots & t_{i-1}^j & a_{i-1} & \cdots & a_{i-1} \\ t_i^1 & \cdots & t_i^j & t_i^{j+1} & \cdots & t_i^n \\ \vdots & & \vdots & \vdots & & \vdots \\ t_m^1 & \cdots & t_m^j & t_m^{j+1} & \cdots & t_m^n \end{pmatrix}$$

Figure 4: Illustration of the argument in claim 3.5. If there exists an instance similar to the left instance, where machine 1 receives all jobs in J but not all jobs $1, ..., n$ (in the figure – jobs $1, ..., j+1$ but not job n), we move to an instance similar to the right instance. Weak monotonicity and the transition from the left instance to the right instance imply that machine 1 cannot receive all jobs (in particular, job n in the figure), contradicting the approximation assumption.

projection of itself. Since $|J^*| < |J|$ then by the inductive assumption machine 1 must receive all jobs $1, ..., n$ in $\tilde{s}$. However weak monotonicity and the transition from s to $\tilde{s}$ imply (using claim 2.4, part 2) that machine 1 cannot receive any job from J^* in $\tilde{s}$, which is a contradiction. □

Claim 3.5 *Fix some instance s that is a (J', i')-projection of an ordered instance t, for any $J' \subseteq N$ and for any $2 \leq i' \leq m+1$. If machine 1 receives all jobs in J' in the instance s, then it must receive all jobs $1, ..., n$ in s.*

Proof: (See illustration in Figure 4). Let S be the set of jobs allocated to machine 1 in the instance s, and assume by contradiction that $J' \subseteq S \neq N$. Let $\{a_i\}_{i=2}^m, \delta$ be constants showing that s is a (J', i')-projection of t (according to Definition 3.2). Consider the following cost vector $\tilde{s}_1$ for machine 1:

$$\tilde{s}_1^j = \begin{cases} \delta/2 & j \in S \\ 2\delta & \text{otherwise} \end{cases}$$

Weak monotonicity and the transition from s to $\tilde{s} = (\tilde{s}_1, s_{-1})$ imply (using claim 2.4, part 1) that machine 1 receives S in $\tilde{s}$ as well. Thus there is a job that is allocated to some machine $i' > 1$, and the resulting makespan for $\tilde{s}$ is at least a_2. The optimal makespan for $\tilde{s}$, however, is at most $2n\delta < a_2/c$, which contradicts the approximation assumption. □

3.2 Analysis of Some Useful Instances

The proof of the inductive assumption starts by assuming (by contradiction) that there exists an instance s for which machine 1 does not receive all jobs $1, ..., n$, and that s is a (J, i)-projection

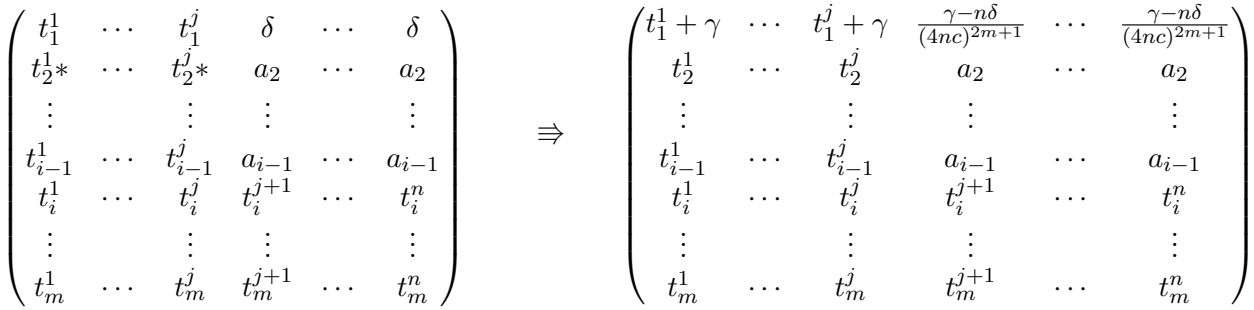

Figure 5: Illustration of the argument in claim 3.6. Given an instance s (similar to the left instance) which is a (J,i)-conversion of some ordered instance t, we define an instance $s(\gamma)$ for $\gamma \in (n\delta, n\delta+y)$, similar to the right instance. If machine 1 does not receive any job in J in the instance s then weak monotonicity implies that machine 1 does not receive any job in J in $s(\gamma)$.

of some ordered instance t, with parameters $\{a_i\}_{i=2}^m, \delta$. By claims 3.4 and 3.5 machine 1 does not receive any of the jobs in J in the instance s. In this section we consider several other scenarios, related to s, and prove some of their properties. These will be used in the next section to lay out the main argument that shows a contradiction to the assumption about the existence of such s, by this concluding the proof of the inductive hypothesis.

Let $\tilde{y} = \min_{j\in J} \frac{t_2^j - t_1^j}{2n} - a_m - \delta$ (by Definition 3.2 $\tilde{y} > 0$) and $y = \min(\delta, \tilde{y})$. For any $\gamma \in (n\delta, n\delta + y)$, let $t_1(\gamma)$ be the following cost vector:

$$t_1^j(\gamma) = \begin{cases} t_1^j + \gamma & j \in J \\ \frac{\gamma - n\delta}{(4nc)^{2m+1}} & \text{otherwise} \end{cases}$$

Claim 3.6 *If machine 1 does not receive any job $j \in J$ in s then all jobs in J are assigned to machines $2, ..., i-1$ in $s(\gamma) = (t_1(\gamma), s_{-1})$.*

Proof: (See illustration in Figure 5). By definition, for any $j \in \bar{J}$ we have $t_1^j(\gamma) < y < \delta$, and for any $j \in J$ we have $t_1^j(\gamma) - t_1^j > n\delta$. Thus weak monotonicity and the transition from s to $s(\gamma)$ imply (using claim 2.4, part 2) that machine 1 cannot receive any job from J in $s(\gamma)$. Furthermore, $s(\gamma)$ is a (J,i)-projection of $t(\gamma) = (t_1(\gamma), t_{-1})$, using $\{a_i\}_{i=2}^m$ and $\tilde{\delta} = \frac{\gamma - n\delta}{(4nc)^{2m+1}}$, since:

- For any $j \in J$ we have $t_1^j(\gamma) < t_1^j + 2n\delta < t_1^j + a_2 < t_2^j$ and therefore $t(\gamma)$ is an ordered instance.
- $\tilde{\delta} < (\tilde{y} <) \min_{j\in J} \frac{t_2^j - t_1^j(\gamma)}{2n} - a_m$.

Therefore, by claim 3.3, in $s(\gamma)$ machines $2, ..., i-1$ receive all jobs in J. □

$$\begin{pmatrix} t_1^1+\gamma_2 & \cdots & t_1^j+\gamma_2 & \frac{\gamma_2-n\delta}{(4nc)^{2m+1}} & \cdots & \frac{\gamma_2-n\delta}{(4nc)^{2m+1}} \\ t_1^1+\gamma_1 & \cdots & t_1^j+\gamma_1 & \frac{\gamma_1-n\delta}{(4nc)^{2m+1}} & \cdots & \frac{\gamma_1-n\delta}{(4nc)^{2m+1}} \\ t_3^1 & \cdots & t_3^j & a_3 & \cdots & a_3 \\ \vdots & & \vdots & \vdots & & \vdots \\ t_{i-1}^1 & \cdots & t_{i-1}^j & a_{i-1} & \cdots & a_{i-1} \\ t_i^1 & \cdots & t_i^j & t_i^{j+1} & \cdots & t_i^n \\ \vdots & & \vdots & \vdots & & \vdots \\ t_m^1 & \cdots & t_m^j & t_m^{j+1} & \cdots & t_m^n \end{pmatrix}$$

Figure 6: An illustration of the instance $\tilde{s}$ of Claim 3.7, assuming $r = 2$.

Claim 3.7 *Assume that machine 1 does not receive any job $j \in J$ in s. Then there exist $\gamma_1, \gamma_2 \in (n\delta, n\delta + y)$, $\gamma_1 < \gamma_2$, such that*

1. *there exists a machine $r \in \{2, ..., i-1\}$ that receives some of the jobs in J in both instances $s(\gamma_1)$ and $s(\gamma_2)$, and,*
2. *the instance $\tilde{s}$ in which the cost vector of machine 1 is $t_1(\gamma_2)$, the cost vector of machine r is $t_1(\gamma_1)$, and the rest of the cost vectors are as in s is a $(J,2)$-projection of itself.* [11]

Proof: (See illustration in Figure 6). Let $x = \frac{y}{(8nc)^{2m+1}}$, and $\alpha = 8nc$. Define $2m+1$ subintervals of $(n\delta, n\delta + y)$, where the d'th interval ($d = 0, 1, ..., 2m$) is $(n\delta + x\sum_{d'=0}^{d-1}\alpha^{d'}, n\delta + x\sum_{d'=0}^{d}\alpha^{d'})$. Thus, the first interval (d=0) is $(n\delta, n\delta + x)$ (i.e. of length x), the second interval ($d = 1$) is $(n\delta + x, n\delta + x + x\alpha)$ (i.e. of length $x\alpha$), the third interval ($d = 2$) is $(n\delta + x + x\alpha, n\delta + x + x\alpha + x\alpha^2)$ (i.e. of length $x\alpha^2$), and so on. Since $\alpha > 2$ we have $x\sum_{d'=0}^{2m}\alpha^{d'} < x\alpha^{2m+1} = y$, which implies that all intervals are in $(n\delta, n\delta + y)$. Choose an arbitrary point γ in each interval $d = 1, ..., 2m$ and label the interval as r, where machine r receives some of the jobs in J in instance $s(\gamma)$. Since there are $2m$ intervals and at most $m-1$ possible labels (recall that only machines $2, ..., i-1$ receive a non-empty set of jobs in J in $s(\gamma)$), there exist two non-adjacent subintervals that are labeled by the same r, and we choose γ_1, γ_2 as the points taken from these two intervals. This shows the first claimed property.

We now show that the instance $\tilde{s}$ where the cost vector of machine 1 is $t_1(\gamma_2)$, the cost vector of machine r is $t_1(\gamma_1)$, and the rest of the cost vectors are as in s is a $(J,2)$-projection of itself (notice that $\tilde{s}$ is indeed an ordered instance). Let $\delta_1 = \frac{\gamma_1 - n\delta}{(4nc)^{2m+1}}$ and $\delta_2 = \frac{\gamma_2 - n\delta}{(4nc)^{2m+1}}$. We rely on two inequalities:

[11] Definition 3.2 requires that machine 1 will be the lowest machine, and in $\tilde{s}$ machine r is the lowest machine, but anonymity ensures that switching the cost vectors of machines 1 and r results in an equivalent instance, and this instance is a $(J,2)$-projection of itself. In fact we do not really need to rely on anonymity in this specific place, we can alternatively define projection by using some permutation π of the machines, so that the machine $\pi(l)$ is the l'th lowest machines for $l = 1, ..., m$. We avoid this formalization as it just complicates notation.

- $\delta_1 < \delta_2/(2nc)$: suppose that γ_1, γ_2 are from intervals d_1, d_2, respectively, where $d_2 \geq d_1 + 2$. Then, $\gamma_2 - n\delta > x \sum_{d'=0}^{d_2-1} \alpha^{d'} > x\alpha \sum_{d'=0}^{d_1} \alpha^{d'} > \alpha(\gamma_1 - n\delta)$. Therefore,

$$\frac{\delta_2}{\delta_1} = \frac{\gamma_2 - n\delta}{\gamma_1 - n\delta} > \alpha > 2nc$$

- $\delta_2 < (\gamma_2 - \gamma_1)/(4n)$: We have $\gamma_2 - \gamma_1 = (\gamma_2 - n\delta) - (\gamma_1 - n\delta)$ and from the previous bullet we have $\gamma_2 - n\delta > \alpha(\gamma_1 - n\delta)$. Also, since γ_1 belongs to some interval $d \geq 1$ (i.e. does not belong to the first interval $(n\delta, n\delta + x)$), we have $\gamma_1 > n\delta + x$. Thus

$$\frac{\gamma_2 - \gamma_1}{4n} = \frac{(\alpha - 1)(\gamma_1 - n\delta)}{4n} \geq \gamma_1 - n\delta > x = \frac{y}{\alpha^{2m+1}} > \frac{\gamma_2 - n\delta}{\alpha^{2m+1}} = \delta_2$$

Notice that the inequality $\delta_2 < (\gamma_2 - \gamma_1)/(4n)$ also implies that $\delta_2 < \frac{\gamma_2 - \gamma_1}{2n} - \delta_2$.

To show that $\tilde{s}$ is a $(J, 2)$-projection of itself, choose small enough ϵ such that the inequalities in the bullets above continue to hold when δ_2 is replaced by $\delta_2 - \epsilon$. Choose constants $\{\tilde{a}_i\}_{i=2}^m$ (needed for the $(J, 2)$-projection) such that $\delta_2 - \epsilon < \tilde{a}_2 < \cdots < \tilde{a}_m < \delta_2$. Let us verify that the required inequalities of Definition 3.2 hold:

- $\tilde{a}_m < \min_{j \in \bar{J}} t_1^j(\gamma_2) = \delta_2$ by construction.
- $\delta_1 < \frac{\tilde{a}_2}{2nc}$ since $\delta_1 < \frac{\delta_2 - \epsilon}{2nc}$.
- $\delta_1 < \min_{j \in J} \frac{t_1^j(\gamma_2) - t_1^j(\gamma_1)}{2n} - \tilde{a}_m$, since $\delta_1 < \delta_2 < \frac{\gamma_2 - \gamma_1}{2n} - \delta_2$.

Thus we have verified that $\tilde{s}$ is a $(J, 2)$-projection of itself, and the claim follows.

□

3.3 Bottom-line Argument

We now formally conclude the inductive proof:

Claim 3.8 *Fix $J \subseteq N$ and $2 \leq i \leq m + 1$. In any (J, i)-projection of any ordered instance t, machine 1 must be allocated all jobs.*

Proof: (See illustration in Figure 7). We prove by induction on $|J|, i$, where we advance over $|J| = 1, .., m$, and, for every fixed J, over $i = m + 1, ..., 2$. Thus, we assume that the claim is true for any (J', i')-projection of any ordered instance t, where either $|J'| < |J|$ and $2 \leq i' \leq m + 1$, or $J' = J$ and $i < i' \leq m + 1$, and we prove for (J, i). The base of the induction ($|J| = 1, i = m + 1$) is by the same proof below.

If $i = 2$ the claim follows immediately from claim 3.3. Thus assume $i \geq 3$. Assume by contradiction that there exists an instance s for which machine 1 does not receive all jobs $1, ..., n$, and that s is a (J, i)-projection of some ordered instance t, with parameters $\{a_i\}_{i=2}^m, \delta$. By claims 3.4 and 3.5 machine 1 does not receive any of the jobs in J in the instance s. Consider the following sequence of instances:

A. The cost vector of machine 1 is $t_1(\gamma_2)$ and the other cost vectors are as in s (this is the instance $s(\gamma_2)$, as defined in the statement of claim 3.6). By construction machine 1 does not receive any job from J and machine r receives some job(s) from J.

B. The cost vector of machine 1 is $t_1(\gamma_2)$, the cost vector of machine r is $t_1(\gamma_1)$, and the other cost vectors are as before (this is the instance $\tilde{s}$, as defined in Claim 3.7). Recall that $2 \leq r \leq i-1$. For any $j \in J$, $na_r < t_r^j - t_1^j(\gamma_1)$ since $\gamma_1 \leq 2n\delta < a_r$ and $t_r^j - t_1^j > 2na_r$. Therefore weak monotonicity and the transition from s_r to $t_1(\gamma_1)$ imply that machine r must receive at least one job from J in $\tilde{s}$ since the decrease in the cost of any job $j \in J$ is more than na_r and the total decrease of all jobs $j \in \bar{J}$ is less than na_r. Since this instance is a $(J, 2)$-projection of itself, claims 3.4 and 3.5 imply that machine r must receive all jobs $1, ..., n$. In particular machine 1 does not receive any job.

C. The cost of machine 1 is s_r, the cost of machine r is $t_1(\gamma_1)$, and the other cost vectors are as in s. We call this instance s^*. By weak monotonicity machine 1 does not receive any job in the instance s^*. However s^* can be obtained from the instance $s(\gamma_1)$ by switching the costs of machine 1 and machine r. Since in $s(\gamma_1)$ machine r receives a non-empty set of jobs (Claim 3.2), and the instance $s(\gamma_1)$ is with no ties, anonymity implies that machine 1 must receive a non-empty set of jobs in s^*, and we reach a contradiction.

We have reached a contradiction, and thus we conclude that machine 1 receives all jobs $1, ..., n$ in the instance s, as claimed. □

Since a (N, i)-projection of t is t itself (regardless of i), Theorem 3.1 follows from claim 3.8.

$$\begin{pmatrix} t_1^1+\gamma_2 & \cdots & t_1^j+\gamma_2 & \delta_2 & \cdots & \delta_2 \\ t_2^1* & \cdots & t_2^j & a_2 & \cdots & a_2 \\ t_3^1 & \cdots & t_3^j & a_3 & \cdots & a_3 \\ \vdots & & \vdots & \vdots & & \vdots \\ t_{i-1}^1 & \cdots & t_{i-1}^j & a_{i-1} & \cdots & a_{i-1} \\ t_i^1 & \cdots & t_i^j & t_i^{j+1} & \cdots & t_i^n \\ \vdots & & \vdots & \vdots & & \vdots \\ t_m^1 & \cdots & t_m^j & t_m^{j+1} & \cdots & t_m^n \end{pmatrix} \Rrightarrow \begin{pmatrix} t_1^1+\gamma_2 & \cdots & t_1^j+\gamma_2 & \delta_2 & \cdots & \delta_2 \\ t_1^1*+\gamma_1 & \cdots & t_1^j*+\gamma_1* & \delta_1* & \cdots & \delta_1* \\ t_3^1 & \cdots & t_3^j & a_3 & \cdots & a_3 \\ \vdots & & \vdots & \vdots & & \vdots \\ t_{i-1}^1 & \cdots & t_{i-1}^j & a_{i-1} & \cdots & a_{i-1} \\ t_i^1 & \cdots & t_i^j & t_i^{j+1} & \cdots & t_i^n \\ \vdots & & \vdots & \vdots & & \vdots \\ t_m^1 & \cdots & t_m^j & t_m^{j+1} & \cdots & t_m^n \end{pmatrix}$$

$$\Rrightarrow \quad \begin{pmatrix} t_2^1 & \cdots & t_2^j & a_2 & \cdots & a_2 \\ t_1^1+\gamma_1 & \cdots & t_1^j+\gamma_1 & \delta_1 & \cdots & \delta_1 \\ t_3^1 & \cdots & t_3^j & a_3 & \cdots & a_3 \\ \vdots & & \vdots & \vdots & & \vdots \\ t_{i-1}^1 & \cdots & t_{i-1}^j & a_{i-1} & \cdots & a_{i-1} \\ t_i^1 & \cdots & t_i^j & t_i^{j+1} & \cdots & t_i^n \\ \vdots & & \vdots & \vdots & & \vdots \\ t_m^1 & \cdots & t_m^j & t_m^{j+1} & \cdots & t_m^n \end{pmatrix}$$

Figure 7: Illustration of the argument in Claim 3.8, assuming $r = 2$. We start (A) in the top-left instance, in which machine 2 receives some job(s) in J. We then move to the top-right instance (B), by lowering the job-costs of machine 2, and we show that now machine 2 receives all jobs. We finally move (C) to the bottom instance by increasing the costs of machine 1. By weak monotonicity it does not receive any job in (C). However the instance (C) can be obtained from the instance $s(\gamma_1)$ by switching the costs of machine 1 and machine r. Since in $s(\gamma_1)$ machine r receives a non-empty set of jobs, anonymity implies that machine 1 must receive a non-empty set of jobs in (C), and we reach a contradiction.

4 Summary and Discussion

We have shown that every truthful anonymous mechanism for the problem of job-scheduling on unrelated machines must obtain an approximation ratio of at least m, the number of machines. The proof shows that every truthful anonymous mechanism with a finite approximation ratio assigns the jobs as the VCG mechanism, for "ordered instances" (i.e. instances in which the cost vector of machine $i+1$ is larger (coordinate-wise) than the cost vector of machine i, for every $i = 1, ..., m-1$.

The proof uses weak monotonicity to reason about the connection between various different instances, at the ends connecting to an ordered instance t. The proof starts with a simple instance in which the machines' costs for job 1 are identical to their costs of the job in t, and the costs of the other jobs are "small enough", and shows that in this instance machine 1 must be assigned all jobs. The proof then gradually increases the costs of the machines to be the costs in t, in a specific way that repeatedly uses weak monotonicity to argue that the fact that the in the current instance all jobs are assigned to machine 1 implies that in the next instance this variant will still hold. Anonymity is used in the process in one specific crucial place.

The approximation assumption is also used in one crucial place, and is necessary since without it the claim is false. For example, maximal-in-range mechanisms other than VCG are also truthful. A maximal-in-range mechanism fixes a strict subset $\tilde{A}$ of all possible assignments A and chooses the assignment with maximal welfare among the assignments in $\tilde{A}$.

In this paper we study direct mechanisms that have truthfulness as a dominant-strategy. One may also consider indirect scheduling mechanisms. By the revelation principle, any indirect mechanism can be converted to a direct one, and in order to take our anonymity assumption into account, we must consider only symmetric equilibria of indirect mechanism. Thus, our theorem implies that there does not exist an indirect scheduling mechanism with a symmetric ex-post equilibrium outcome that is a c-approximation to the optimal makespan. Indirect mechanisms with asymmetric equilibria translate to direct non-anonymous mechanisms, for which the problem remains open.

The striking simplicity of VCG, that manages to "internalize" the goal of the mechanism, may tempt the naive mechanism designer to believe that such methods are possible also for other implementation goals. Roberts [14] shows that this is not true if the domain of preferences is unrestricted – only VCG and other affine maximizers are implementable. We show a conceptually similar claim for a scheduling domain – every anonymous truthful mechanism identifies with VCG over a large family of instances. However this conceptual similarity is misleading, as the scheduling domain is so restricted that it admits many implementable non-affine-maximizers. For example, allocating each job independently using some non-affine-maximizer mechanism for single-dimensional domains (those are abundant). Thus our result is not implied by Roberts' result, nor is it a generalization of it. It belongs to a different class of results, that add conditions on top of implementability (here, we require makespan approximation), and show using these extra conditions that VCG is the best possible. This is different than Roberts in a subtle way, since without the extra conditions many

other implementable functions exist.

We have already mentioned in the Introduction the future research direction of understanding the power of randomized scheduling mechanisms. Another interesting direction is to limit the domain so that machines' costs will be limited. It is clear from our proof that we need an extremely reach domain of possible job costs. What happens if the domain is bounded, or being restricted in other meaningful ways? The paper [11] demonstrates a possibility in this direction, and it is interesting to know if more can be done. Another possible direction is to consider a probability distribution over the instances, and/or use weaker solution concepts.

References

[1] G. Aggarwal, A. Fiat, A.V. Goldberg, J.D. Hartline, N. Immorlica, and M. Sudan. Derandomization of auctions. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, page 625, 2005.

[2] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *42nd IEEE Symposium on Foundations of Computer Science, 2001. Proceedings*, pages 482–491, 2001.

[3] S. Bikhchandani, S. Chatterji, R. Lavi, A. Mu'alem, N. Nisan, and A. Sen. Weak monotonicity characterizes deterministic dominant-strategy implementation. *Econometrica*, 74(4):1109–1132, 2006.

[4] G. Christodoulou, E. Koutsoupias, and A. Kovács. Mechanism design for fractional scheduling on unrelated machines. *ACM Transactions on Algorithms (TALG)*, 6(2):1–18, 2010.

[5] G. Christodoulou, E. Koutsoupias, and A. Vidali. A characterization of 2-player mechanisms for scheduling. *Algorithms-ESA 2008*, pages 297–307, 2008.

[6] G. Christodoulou, E. Koutsoupias, and A. Vidali. A lower bound for scheduling mechanisms. *Algorithmica*, 55(4):729–740, 2009.

[7] G. Christodoulou and A. Kovacs. A deterministic truthful PTAS for scheduling related machines. In *Proceedings of the 21st annual ACM-SIAM symposium on Discrete algorithms*, 2010.

[8] P. Dhangwatnotai, S. Dobzinski, S. Dughmi, and T. Roughgarden. Truthful approximation schemes for single-parameter agents. In *IEEE 49th Annual IEEE Symposium on Foundations of Computer Science, 2008. FOCS'08*, pages 15–24, 2008.

[9] S. Dobzinski and M. Sundararajan. On characterizations of truthful mechanisms for combinatorial auctions and scheduling. In *Proceedings of the 9th ACM conference on Electronic commerce*, pages 38–47, 2008.

[10] Elias Koutsoupias and Angelina Vidali. A lower bound of 1+phi for truthful scheduling mechanisms. In *32nd International Symposium on Mathematical Foundations of Computer Science*, 2007.

[11] R. Lavi and C. Swamy. Truthful mechanism design for multidimensional scheduling via cycle monotonicity. *Games and Economic Behavior*, 67(1):99–124, 2009.

[12] A. Mu'alem and M. Schapira. Setting lower bounds on truthfulness: extended abstract. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007.

[13] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

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

[15] C. Yu. Truthful mechanisms for two-range-values variant of unrelated scheduling. *Theoretical Computer Science*, 410(21-23):2196–2206, 2009.