MATHEMATICS OF OPERATIONS RESEARCH
Vol. 37, No. 2, May 2012, pp. 244–258
ISSN 0364-765X (print) | ISSN 1526-5471 (online)

http://dx.doi.org/10.1287/moor.1110.0534

Optimal Lower Bounds for Anonymous Scheduling Mechanisms

Itai Ashlagi
Sloan School of Management, MIT, Cambridge, Massachusetts, 02142, USA,
iashlagi@mit.edu, http://web.mit.edu/iashlagi/www/

Shahar Dobzinski
Computer Science Department, Cornell University, Ithaca, NY, 14853, USA,
shahar@cs.cornell.edu, https://sites.google.com/site/dobzin/

Ron Lavi
Faculty of Industrial Engineering and Management, Technion, Haifa, 32000, Israel,
ronlavi@ie.technion.ac.il, http://ie.technion.ac.il/~ronlavi/

We consider the problem of designing truthful mechanisms on m unrelated machines, to minimize some optimization goal. Nisan and Ronen [Nisan, N., A. Ronen. 2001. Algorithmic mechanism design. *Games Econom. Behav.* **35** 166–196] consider the specific goal of makespan minimization, and show a lower bound of 2, and an upper bound of m. This large gap inspired many attempts that yielded positive results for several special cases, but very partial success for the general case: the lower bound was slightly increased to 2.61 by Christodoulou et al. [Christodoulou, G., E. Koutsoupias, A. Kovács. 2010. Mechanism design for fractional scheduling on unrelated machines. *ACM Trans. Algorithms* (*TALG*) **6**(2) 1–18] and Koutsoupias and Vidali [Koutsoupias, E., A. Vidali. 2007. A lower bound of 1+phi for truthful scheduling mechanisms. *Proc. 32nd Internat. Sympos. Math. Foundations Comput. Sci.* (*MFCS*)], while the best upper bound remains 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. Moreover, our proof yields similar optimal bounds for two other optimization goals: the sum of completion times and the l_p norm of the schedule.

Key words: anonymous; mechanism; scheduling; truthfulness
MSC2000 subject classification: Primary: 91B26; secondary: 91A80
OR/MS subject classification: Primary: games/group decisions, noncooperative; secondary: production/scheduling, multiple machine
History: Received November 29, 2010; revised September 21, 2011.

1. Introduction.

1.1. Background and motivation.

Nisan and Ronen [14] introduced the field of "algorithmic mechanism design" by studying the problem of truthful scheduling on unrelated machines, a problem that has taken a central place in this field ever since. The problem of job scheduling on unrelated machines is classic in the disciplines of Computer Science and Operations Research. 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 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 when the processing of all jobs will be completed. Other popular goals are to minimize the sum of completion times of the schedule and to minimize the l_p norm of the schedule.

Nisan and Ronen add 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 to compensate them for their efforts.

An easy observation made by Nisan and Ronen 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

[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 equal 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.

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 to the makespan.

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

- **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 et al. [7] were able to slightly improve the lower bound from 2 to 2.41, and it was later improved to 2.61 later by Koutsoupias and Vidali [11]. Mu'alem and Schapira [13] and Christodoulou et al. [5] prove a similar lower bound of $2-(1/m)$ for randomized and fractional mechanisms, respectively. Dobzinski and Sundararajan [9] and Christodoulou et al. [6] attempted to characterize truthful mechanisms for this problem, but succeeded in doing so only for the very limited case of two machines. The result of Dobzinski and Sundararajan also yields a lower bound of 2 on the approximation ratio w.r.t. the weighted sum of completion times of any truthful mechanism. All these papers involve many nontrivial observations and technicalities, further emphasizing the basic difficulties that this problem presents.

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 two recent notable truthful PTASs that essentially conclude all computational efforts in this direction by Dhangwatnotai et al. [8] and Christodoulou and Kovacs [4]. 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 mechanisms with constant-factor approximation ratios. Yu [16] 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 nontrivial 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

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

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 with respect to the makespan.*

The theorem abstracts away from the specific technicalities of the makespan optimization goal, and thus we are able to show with almost no additional technical effort similar lower bounds for other optimization criteria as well. For example, we show a lower bound of m for the sum of completion times, and a lower bound of $m^{1-1/p}$ for minimizing the l_p norm. In all cases we show that the VCG mechanism obtains exactly these bounds, and is thus, in this respect, the best truthful mechanism that is anonymous.

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 nonanonymous mechanisms. However, are nonanonymous mechanisms more powerful than anonymous ones? All state-of-the-art mechanisms for the special cases in the recent literature are anonymous (for example Lavi and Swamy [12] and Dhangwatnotai et al. [8]). This might suggest that anonymous mechanisms for this problem are as powerful as nonanonymous 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.

At 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 "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 (with respect to a variety of optimization goals) 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 a substantial amount of technical work, which allows us to consider instances with an 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

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

[4] There is a single example in a different context ("digital goods") where it is proved by Aggarwal et al. [1] that anonymous mechanisms are less powerful. 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 by Aggrawal et al. 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.

lower bounds that were mentioned above consider longer paths of instances, 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 the *amount* the costs of the jobs change, 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, because, 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, then, 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 the various lower bounds it implies, while §4 describes the proof of the main theorem. Section 6 summarizes and discusses a few conceptual issues.

2. Preliminaries.

2.1. Job scheduling for various optimization goals. We have m machines and n jobs, where $N = \{1, \ldots, n\}$. Let $t_i^j > 0$ 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 $t_i(S) = \sum_{j \in S} t_i^j$ the total time it takes machine i to process the jobs in S. We sometimes describe an instance of this problem by a matrix in which the ith 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:

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

Let $T = \Re_{>0}^{m \times n}$ be the space of all possible instances, and let A be the space of all possible allocations of jobs to machines. An allocation rule is a function $f\colon T \to A$ that allocates the jobs to the machines; i.e., for some instance t, $f(t) = S = (S_1, \ldots, S_m) \in A$, where S_i is the set of jobs allocated to machine i by f.

A designer wishes to implement an allocation rule $f\colon T \to A$ that optimizes some global criterion that measures the quality of the schedule. We consider the following widely studied criteria:

- **Makespan:** The "makespan" of a schedule $S = (S_1, \ldots, S_m)$ is defined as $m(S) = \max_{i=1}^m t_i(S_i)$. This is the time by which all jobs will be processed, according to the schedule. It is probably the most popular criterion studied in the scheduling literature.
- **The l_p norm:** The l_p norm of a schedule, for some $p \geq 1$, is defined as $L_p(S) = (\sum_{i=1}^m t_i(S_i)^p)^{1/p}$. L_∞ is exactly the makespan criterion defined above, and the rationale to move to L_p for some finite p is to give more weight to all the load differences between the machines, not only to the machine with the heaviest load.

[5] A notable exception is Koutsoupias and Vidali [11] which also uses induction. The induction used here is very different.

• **Sum of completion times:** The completion time of a job in a given schedule is the time at which the job ends.[6] This criterion is appropriate if jobs' output is useful even if other jobs have not finished. In this case, for example, the designer may prefer to schedule short jobs before long jobs.

The literature considers many more optimization goals, for example, a weighted version of the last goal. We focus attention on three optimization criteria for the sake of conciseness, but we should mention that we could not find a criterion for which our method does not work.

For each such goal, the designer aims to minimize the value of the criterion for the outputted schedule. For a given criterion C, let OPT be the allocation rule that outputs a schedule with the minimal value of C, for any instance $t \in T$. Because the allocation rule will be required to satisfy other properties (e.g., truthfulness as detailed below) the designer may choose a rule different than OPT. It is common to compare a given allocation rule f to the optimal allocation rule, OPT, by its "approximation ratio":

Definition 2.1 (Approximation Ratio). An allocation rule f is an "α-approximation with respect to C," for some real number $\alpha \geq 1$, if for any $t \in T$, f outputs a schedule with a value of C at most α times the optimal value of C for t. α is termed the "approximation ratio" of f.

For example, f is an α-approximation with respect to the makespan if $m(f(t)) \leq \alpha \cdot m(\mathrm{OPT}(t))$ for every $t \in T$. These definitions imply that we study only deterministic allocation rules.

A common property of the above goals, as well as of most other goals that are studied in the scheduling literature, is the following decisiveness property. Our proof abstracts away from the technicalities of the various optimization goals by relying instead on decisiveness.

Definition 2.2 (Decisiveness). An allocation rule f is "*decisive*" if there exists a constant $c \geq 1$ (that may depend on m, n, but not on the input cost vector t) that satisfies the following condition for every $t \in T$: if there exists a machine i such that, for every $j \in N$, $t_i^j < \min_{i' \in \{1,\ldots,i-1,i+1,\ldots,m\},\, j' \in N} t_{i'}^{j'}/c$, then $f(t)$ assigns all jobs to machine i.

In other words, an allocation rule is decisive if a machine may always receive all jobs by declaring some "attractive" costs for these jobs, compared to the other declared costs. All the criteria described above imply decisiveness.

Proposition 2.1. *If an allocation rule f is an α-approximation with respect to either the makespan, the l_p norm, or the sum of completion times, then f is decisive.*

Proof. **Makespan and the l_p norm:** Fix $c = \alpha n$. We show that if there exists a machine i such that $t_i^j < \min_{i' \in \{1,\ldots,i-1,i+1,\ldots,m\},\, j' \in N} t_{i'}^{j'}/c$ for every $j \in N$, then all jobs must be assigned to i. Assume towards a contradiction that there exists a job j' that is assigned to some machine $l \neq i$. Therefore, both the makespan and the l_p norm of this schedule have value at least $t_l^{j'}$. For every job j we have $t_i^j < t_l^{j'}/(\alpha n)$. Thus $\sum_j t_i^j < \cdot t_l^{j'}/\alpha$ or equivalently, $t_l^{j'} > \alpha \sum_j t_i^j$. The optimal makespan and the optimal l_p norm for t, on the other hand, are at most $\sum_j t_i^j$ by assigning all jobs to machine i. Thus the approximation in this case is larger than α, a contradiction.

Sum of completion times: We show that $c = \alpha \cdot n^2$. If the allocation rule assigns a job j' to some machine $l \neq i$, then the completion time of j' alone is at least $t_l^{j'}$. For every job j we have $t_i^j < t_l^{j'}/(\alpha n^2)$. Thus $\sum_j t_i^j < n \cdot t_l^{j'}/(\alpha n^2)$ or equivalently, $t_l^{j'} > \alpha n \sum_j t_i^j$. The optimal solution, on the other hand, is at most $n \sum_j t_i^j$ by assigning all jobs to machine i. Thus the approximation in this case is larger than α, a contradiction. □

2.2. A mechanism-design setup. Following Nisan and Ronen [14], 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.[7] A direct mechanism consists of an allocation function f, and a payment function p_i: $T \to \Re$ for every machine i. That is, the players are being asked to report a type, given these reports $t = (t_1, \ldots, t_m)$ 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 such mechanism is the VCG mechanism, which (in this case) assigns each job independently

[6] In this case the schedule should additionally specify an order over all jobs assigned to the same machine, but one can show that on a given machine jobs should be executed by increasing job lengths.

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

to the machine with the smallest cost for that job.[8] VCG has an approximation ratio of m, the number of machines. To see this, define an "ordered instance" t to be a tuple of types $t_1, \ldots, t_m$ such that, for every job j, $t_m^j > \cdots > t_2^j > t_1^j$. 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 $\epsilon \to 0$. Because m can be arbitrarily bad, 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 for all the optimization criteria described above. 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 2.3 (No Ties). An instance 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 2.4 (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 because we do not permute all players, and because 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 $p_1, \ldots, p_m$ such that the mechanism $M = (f, p_1, \ldots, p_m)$ is truthful. The following is a necessary condition for implementability.

Definition 2.5 (Weak Monotonicity; Bikhchandani et al. [3]). An allocation function f is *weakly monotone* if for every machine i, every $t_{-i} \in T_{-i}$,[9] 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 2.1. *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. Then,*

(i) *Fix $\tilde{t}_i \in T_i$ that satisfies, $\forall\, j \in S_i$, $\tilde{t}_i^j < t_i^j$, and $\forall\, j \in N \backslash S_i$, $\tilde{t}_i^j > t_i^j$. Then $S_i = f_i(\tilde{t}_i, t_{-i})$.*

(ii) *Fix $J \subseteq N \backslash S_i$, 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 $\forall\, j \in N \backslash J \backslash S_i$, $t_i^j - \tilde{t}_i^j < \delta$. Let $\tilde{S}_i = f_i(\tilde{t}_i, t_{-i})$. Then $J \cap \tilde{S}_i = \varnothing$.*

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 $t_i(S_i \backslash \tilde{S}_i) - t_i(\tilde{S}_i \backslash S_i) \leq \tilde{t}_i(S_i \backslash \tilde{S}_i) - \tilde{t}_i(\tilde{S}_i \backslash S_i)$. Because $t_i(S_i \backslash \tilde{S}_i) > \tilde{t}_i(S_i \backslash \tilde{S}_i)$ and $t_i(\tilde{S}_i \backslash S_i) < \tilde{t}_i(\tilde{S}_i \backslash S_i)$, it must follow that $S_i \backslash \tilde{S}_i = \tilde{S}_i \backslash S_i = \varnothing$, implying the claim.

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

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

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. Main theorem and implied lower bounds. Recall that an *ordered instance* t is a tuple of types $t_1, \ldots, t_m$ such that, for every job j, $t_m^j > \cdots > t_2^j > t_1^j$. Our main theorem is:

Theorem 3.1. *Let $M = (f, p)$ be a truthful anonymous mechanism that is decisive. Then, for every ordered instance t, M allocates all jobs to machine* 1.

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

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

The proof of this theorem is given in §4. We believe that on top of the applications of the theorem, the proof itself is of interest. This theorem enables us to give optimal lower bounds to the various optimization goals we consider.

PROPOSITION 3.1. *VCG has the best approximation ratio for the makespan, the l_p norm, and the sum of completion times, among all truthful and anonymous mechanisms. Specifically,*

(i) *Every truthful anonymous mechanism has an approximation ratio of at least m with respect to the makespan. Furthermore, VCG provides exactly this approximation ratio.*

(ii) *Every truthful anonymous mechanism has an approximation ratio at least $m^{1-1/p}$ with respect to the l_p norm for all $p \geq 1$. Furthermore, VCG provides exactly this approximation ratio.*

(iii) *Every truthful anonymous mechanism has an approximation ratio of at least m with respect to the sum of completion times. Furthermore, VCG provides exactly this approximation ratio.*[10]

PROOF. **Makespan:** To see that any truthful anonymous mechanism has an approximation ratio at least m, fix some $\epsilon > 0$ and consider an ordered instance with $n = m$ jobs such that $1 + \epsilon > t_m^j > \cdots > t_2^j > t_1^j = 1$ for every job j. By Theorem 3.1 any truthful anonymous mechanism assigns all jobs to machine 1; hence, the makespan is m. The optimal makespan is at most $1 + \epsilon$, and the claim follows. Nisan and Ronen [14] show that VCG has an approximation ratio of m.

The l_p norm: To see that any truthful anonymous mechanism has an approximation ratio at least $m^{1-1/p}$, consider the same ordered instance from the previous paragraph. Any truthful anonymous mechanism assigns all jobs to machine 1; hence, the l_p norm of its schedule is m. The optimal schedule assigns one job to each machine; thus the load of each machine is at most $1 + \epsilon$, and the l_p norm of this schedule is at least $m^{1/p}$. Thus the approximation ratio of any truthful anonymous mechanism is at least $m^{1-1/p}$.

We now show that the approximation ratio of VCG is $m^{1-1/p}$. Fix some instance t, and let $(a_1, \ldots, a_m)$ and $(o_1, \ldots, o_m)$ be VCG's resulting load vector and OPT's resulting load vector, respectively. We need to show that $[\sum_{i=1}^m (a_i)^p / \sum_{i=1}^m (o_i)^p]^{1/p} \leq m^{1-1/p}$. Let $x = \sum_{i=1}^m a_i$. First, note that $\sum_{i=1}^m o_i \geq x$ since VCG assigns every job j to a machine i with minimal t_i^j. This implies that $\sum_{i=1}^m (o_i)^p \geq m[x/m]^p$ as $\sum_{i=1}^m (o_i)^p$ is minimal when all the loads o_i are equal. We also have $\sum_{i=1}^m (a_i)^p \leq [\sum_{i=1}^m (a_i)]^p = x^p$. All this implies:

$$\left[\frac{\sum_{i=1}^m (a_i)^p}{\sum_{i=1}^m (o_i)^p}\right]^{1/p} \leq \left[\frac{x^p}{m[x/m]^p}\right]^{1/p} = m^{1-1/p},$$

and the claim follows.

Sum of completion times: To see that any truthful anonymous mechanism has an approximation ratio at least m, fix some $\epsilon > 0$ and some integer β, and consider an ordered instance with $n = \beta m$ jobs such that $1 + \epsilon > t_m^j > \cdots > t_2^j > t_1^j = 1$ for every job j. By Theorem 3.1 any truthful anonymous mechanism assigns all jobs to machine 1. Therefore its sum of completion times is $1 + 2 + \cdots + n = (n(n+1))/2$. If, however, every machine is assigned $\beta = n/m$ jobs, then the total completion time is $m(1 + 2 + \cdots + n/m) = (n(\beta + 1))/2$ (we neglect the ϵ in the calculation). Therefore the approximation ratio of the mechanism is at least $((n(n+1))/2)/((n(\beta+1))/2) = (m(n+1))/(n+m) \leq m$. As n grows to infinity this term approaches m, and the claim follows.

To see that the approximation ratio of VCG is $(m(n+1))/(n+m)$, fix some $t \in T$, and let $a_j = \min_i t_i^j$ (i.e. a_j is the time that VCG needs to process j). Suppose w.l.o.g. that $a_1 \leq a_2 \leq \cdots \leq a_n$. We assume that on each machine the fastest job assigned to it is processed first, the second fastest job second, and so on. Thus the total completion time of VCG is at most $X = a_1 + (a_1 + a_2) + (a_1 + a_2 + a_3) + \cdots + (a_1 + \cdots + a_n)$. Let OPT denote the optimal sum of completion times. We bound X/OPT by looking at the problem of scheduling on *identical* machines. Consider the problem of scheduling n jobs whose processing times are $a_1, \ldots, a_n$ (on every machine because machines are identical). Let C_r be the optimal (minimal) sum of completion times for scheduling these jobs on r identical machines, for any $r \geq 1$. Note that $OPT \geq C_m$ because lowering the costs of the jobs from t_i^j to a_j cannot increase the sum of completion times, and that $C_1 \geq X$ by definition. Eastman et al. [10] show that $C_m \geq C_1(n+m)/(m(n+1))$, implying the claim. □

[10] The proof shows the more concrete approximation ratio $(m(n+1))/(n+m) \leq m$ for VCG, and that this is tight if $n \geq m$. If $n < m$, it can be shown that the approximation ratio of VCG is $(n+1)/2$, and that this is the best possible for any anonymous truthful mechanism, using similar arguments.

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

FIGURE 1. An illustration of an instance that is a $(\{1,\ldots,j\},i)$-projection of t, as in Definition 4.1.

4. Proof of the main theorem. In this section we prove our main theorem. Recall that an *ordered instance* t is a tuple of types $t_1,\ldots,t_m$ such that, for every job j, $t_m^j > \cdots > t_2^j > t_1^j$. We need to show that, if $M=(f,p)$ is a truthful anonymous mechanism that is decisive, then for every ordered instance t, M allocates all jobs to machine 1. Let c be the constant implied by the definition of decisiveness. 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 4.1. Fix $J \subseteq N$ and $2 \le i \le m+1$. Let $\bar{J} = N \backslash J$. An instance s is a (J,i)-projection of an ordered instance t if there exist m real numbers $a_m > \cdots > a_2 > \delta > 0$ that satisfy

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

(where c is the constant that follows from the decisiveness property), such that s has the following structure (see also the illustration in Figure 1):

(i) $s_1^j = t_1^j$ for every $j \in J$, and $s_1^j = \delta$ for every $j \in \bar{J}$.
(ii) For every machine $2 \le i' \le i-1$, $s_{i'}^j = t_{i'}^j$ for every $j \in J$, and $s_{i'}^j = a_{i'}$ for every $j \in \bar{J}$.
(iii) For every machine $i \le i' \le m$, $s_{i'} = t_{i'}$.

It can be verified that, for any instance $t \in T$ and for any $J \subseteq N$ and $2 \le i \le m+1$, the set of (J,i)-projections of t is nonempty.

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. Because an (N,i)-projection of t is t itself (regardless of i), Theorem 3.1 follows from the inductive hypothesis. Our inductive argument advances over $|J| = 1,\ldots,n$, and, for every fixed J, over $i = m+1,\ldots,2$. 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 $2 \le i' \le m+1$, or $J' = J$ and $i < i' \le m+1$. The base of the induction ($|J| = 1, i = m+1$) is proved exactly like the inductive step, as all claims below hold for this case as well.

We divide the proof into three parts. In §4.1 we prove three basic claims that will be used in the main argument. In §4.2 we construct several specific scenarios and prove their technical properties. Section 4.3 uses these scenarios to prove the inductive hypothesis.

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

FIGURE 2. Illustration of the argument in Claim 4.1. 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.

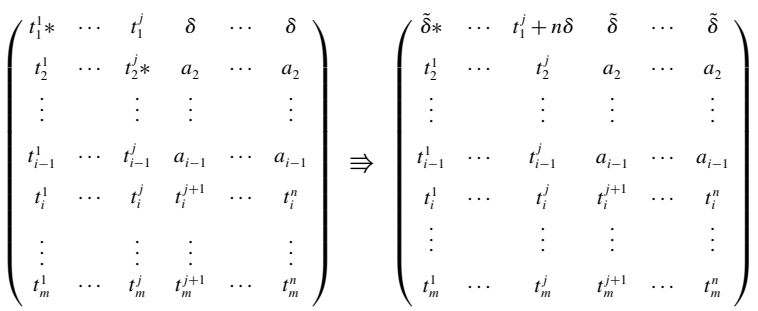

FIGURE 3. Illustration of the argument in Claim 4.2. If there exists an instance similar to the left instance, where machine 1 receives some of the jobs in J but not all of them (in the figure—job 1 but not job j), we move to an instance similar to the right instance, which is (shown to be) a (J^*, i)-projection 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.

4.1. Core arguments. We prove three claims that serve as a starting point to the inductive proof: (1) that machines $i, \ldots, m$ 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 $1, \ldots, n$. These claims hold also for $|J| = 1$. Each claim is accompanied by a figure that illustrates the main argument.

CLAIM 4.1. *In any (J, i)-projection of any ordered instance t, machines $i, \ldots, m$ do not receive any job (i.e., machines $1, \ldots, i-1$ receive all jobs).*

PROOF. (See illustration in Figure 2.) For $i = m+1$, the claim is immediate; thus assume $i \le m$. Assume towards a contradiction that there exists an instance $s = (s_1, \ldots, s_m)$ which is a (J, i)-projection of some ordered instance $t = (t_1, \ldots, t_m)$ (with constants $a_m > \cdots > a_2 > \delta$), and a machine $r \ge i$ such that machine r receives a nonempty set of jobs in instance s. Because all the inequalities in Definition 4.1 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 nonempty set of jobs in s implies that machine r must receive a nonempty set of jobs in $\tilde{s}$, a contradiction. □

CLAIM 4.2. *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 $2 \le i' \le m+1$.*[11] *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 towards a contradiction that $J \cap S \neq \varnothing$ and $J \setminus S \neq \varnothing$. Let $a_m > \cdots > a_2 > \delta$ be the constants from Definition 4.1. 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.[12]

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

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

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

[11] 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).

[12] Notice that by Definition 4.1, 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 $(t_2^j - t_1^j)$.

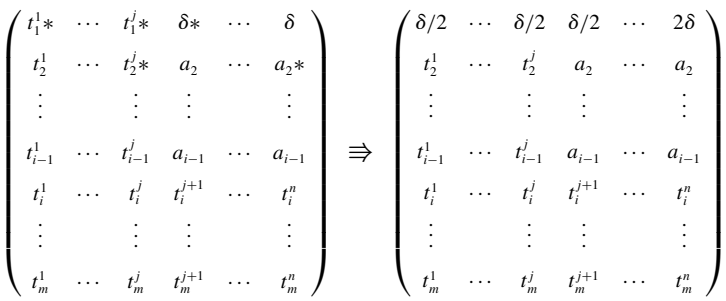

FIGURE 4. Illustration of the argument in Claim 4.3. If there exists an instance similar to the left instance, where machine 1 receives all jobs in J but not all jobs $1,\dots,n$ (in the figure—jobs $1,\dots,j+1$ but not job n), we move to an instance similar to the right instance. Weak monotonicity 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.

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 Definition 4.1. By construction of these constants, we have verified that $\tilde{s}$ is a $(J^*,2)$-projection of itself. Because $|J^*|<|J|$, then by the inductive assumption, machine 1 must receive all jobs $1,\dots,n$ in $\tilde{s}$. However, weak monotonicity and the transition from s to $\tilde{s}$ imply (using Claim 2.1(ii)) that machine 1 cannot receive any job from J^* in $\tilde{s}$, which is a contradiction. □

CLAIM 4.3. *Fix some instance s that is a (J',i')-projection of an ordered instance t, for any $J'\subseteq N$ and for any $2\le i'\le m+1$. If machine 1 receives all jobs in J' in the instance s, then it must receive all jobs $1,\dots,n$ 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 towards a 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 4.1). 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 2.1(i)) that machine 1 receives S in $\tilde{s}$ as well. Thus not all jobs are allocated to machine 1 in $\tilde{s}$. However, in $\tilde{s}$ we have for any job j that $\tilde{s}_1^j\le 2\delta<a_2/c\le\min_{j\in N,\,l=2,\dots,m}\tilde{s}_l^j/c$. Because f is decisive, this implies that machine 1 must be allocated all jobs, a contradiction. □

4.2. Analysis of some useful instances. The proof of the inductive assumption starts by assuming towards a contradiction that there exists an instance s for which machine 1 does not receive all jobs $1,\dots,n$, and that s is a (J,i)-projection of some ordered instance t, with parameters $\{a_i\}_{i=2}^m,\delta$. By Claims 4.2 and 4.3, 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 $\tilde{y}=\min_{j\in J}((t_2^j-t_1^j)/(2n))-a_m-\delta$ (by Definition 4.1 $\tilde{y}>0$) and $y=\min(\delta,\tilde{y})$. For any $\gamma\in(n\delta,n\delta+y)$, let $t_1(\gamma)$ be the following cost vector:

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

CLAIM 4.4. *If machine 1 does not receive any job $j\in J$ in s, then all jobs in J are assigned to machines $2,\dots,i-1$ 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 2.1(ii)) 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 $\tilde{\delta}=(\gamma-n\delta)/(4nc)^{2m+1}$, since:

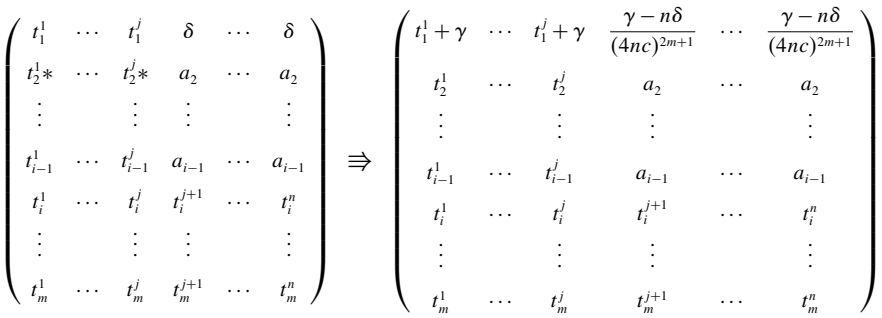

FIGURE 5. Illustration of the argument in Claim 4.4. Given an instance s (similar to the left instance) which is a (J,i)-conversion of some ordered instance t, we define an instance $s(\gamma)$ for $\gamma \in (n\delta, n\delta+y)$, similar to the right instance. If machine 1 does not receive any job in J in the instance s, then weak monotonicity implies that machine 1 does not receive any job in J in $s(\gamma)$.

- $\forall\, j \in J$ 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.
- $\tilde{\delta} < (\tilde{y} <) \min_{j\in J}((t_2^j - t_1^j(\gamma))/(2n)) - a_m$.

Therefore, by Claim 4.1, in $s(\gamma)$ machines $2,\ldots,i-1$ receive all jobs in J. □

CLAIM 4.5. *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*

(i) *there exists a machine* $r \in \{2,\ldots,i-1\}$ *that receives some of the jobs in* J *in both instances* $s(\gamma_1)$ *and* $s(\gamma_2)$, *and,*

(ii) *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.*[13]

PROOF. (See illustration in Figure 6.) Let $x = y/(8nc)^{2m+1}$, and $\alpha = 8nc$. Define $2m+1$ subintervals of $(n\delta, n\delta+y)$, where the dth interval $(d=0,1,\ldots,2m)$ 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. Because $\alpha > 2$ 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 $d=1,\ldots,2m$ and label the interval as r, where machine r receives some of the jobs in J in instance $s(\gamma)$. Because there are $2m$ intervals and at most $m-1$ possible labels (recall that only machines $2,\ldots,i-1$ receive a nonempty set of jobs in J in $s(\gamma)$), there exist two nonadjacent 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 $\delta_1 = (\gamma_1 - n\delta)/(4nc)^{2m+1}$ and $\delta_2 = (\gamma_2 - n\delta)/(4nc)^{2m+1}$. We rely on two inequalities:

- $\delta_1 < \delta_2/(2nc)$: 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, because 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

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

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

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 4.1 hold:

[13] Definition 4.1 requires that machine 1 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 lth lowest machines for $l=1,\ldots,m$. We avoid this formalization as it just complicates notation.

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

FIGURE 6. An illustration of the instance $\tilde{s}$ of Claim 4.5, assuming $r=2$.

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

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

4.3. Bottom-line argument.

We now formally conclude the inductive proof:

CLAIM 4.6. *Fix $J \subseteq N$ and $2 \le i \le m+1$. In any (J,i)-projection of any ordered instance t, machine 1 must be allocated all jobs.*

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

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

FIGURE 7. Illustration of the argument in Claim 4.6, 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. Because in $s(\gamma_1)$ machine r receives a nonempty set of jobs, anonymity implies that machine 1 must receive a nonempty set of jobs in (C), and we reach a contradiction.

PROOF. (See illustration in Figure 7.) We prove by induction on $|J|, i$, where we advance over $|J| = 1, \ldots, m$, and, for every fixed J, over $i = m+1, \ldots, 2$. Thus, we assume that the claim is true for any (J', i')-projection of any ordered instance t, where either $|J'| < |J|$ and $2 \le i' \le m+1$, or $J' = J$ and $i < i' \le m+1$, 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 4.1. Thus, assume $i \ge 3$. Assume towards a contradiction that there exists an instance s for which machine 1 does not receive all jobs $1, \ldots, n$, and that s is a (J, i)-projection of some ordered instance t, with parameters $\{a_i\}_{i=2}^m, \delta$. By Claims 4.2 and 4.3, 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 4.4). By construction (Claims 4.4 and 4.5) 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 4.5). Recall that $2 \le r \le i-1$. For any $j \in J$, $na_r < t_r^j - t_1^j(\gamma_1)$ because $\gamma_1 \le 2n\delta < a_r$ 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 $\tilde{s}$ because 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. Because this instance is a $(J, 2)$-projection of itself, Claims 4.2 and 4.3 imply that machine r must receive all jobs $1, \ldots, n$. 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. Because in $s(\gamma_1)$ machine r receives a nonempty set of jobs (Claim 4.5), and the instance $s(\gamma_1)$ is with no ties, anonymity implies that machine 1 must receive a nonempty 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 $1, \ldots, n$ in the instance s, as claimed. □

Because a (N, i)-projection of t is t itself (regardless of i), Theorem 3.1 follows from Claim 4.6.

5. Example: Illustrating the induction. To give a concrete feel to the induction process in the proof we go over its various steps for the following example instance:

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix}.$$

We will advance in a top-to-bottom fashion, as follows.

5.1. The phase of a $(\{1, 2, 3\}, 4)$-projection. Assuming the correctness of all claims for all smaller projections, we wish to show that, in the original instance, all jobs are allocated to machine 1. We follow the bottom-line argument in §4.3. By Claim 4.2 (we demonstrate its proof as well in §5.2 below), machine 1 either receives all jobs, or none of them. Suppose by contradiction that machine 1 does not receive any job in this instance, and that machine 3 receives some jobs both in the original instance and in the following instance:

$$\begin{pmatrix} 1.5 & 1.5 & 1.5 \\ 2 & 2 & 2* \\ 3* & 3* & 3 \end{pmatrix}.$$

(In this second instance, machine 1 raises all its costs; by weak monotonicity it still does not receive any job.) Weak monotonicity implies that machine 3 receives some jobs when it lowers all its costs to be 1, as in the following instance:

$$\begin{pmatrix} 1.5 & 1.5 & 1.5 \\ 2 & 2 & 2 \\ 1* & 1* & 1* \end{pmatrix}.$$

(Claim 4.2 again implies that machine 3 receives all jobs in this instance.) In particular, machine 1 does not receive any job. Now, when machine 1 increases all costs to 3, as in the instance below, weak monotonicity implies that machine 1 still does not receive any job:

$$\begin{pmatrix} 3 & 3 & 3 \\ 2 & 2 & 2* \\ 1* & 1* & 1 \end{pmatrix}.$$

However, this is a contradiction to anonymity (compare to the original instance; since machines 1 and 3 switched costs and in the original instance machine 3 received some jobs, in the last instance machine 1 should receive the same set of jobs). We have therefore obtained a contradiction, and the claim follows.

5.2. Claim 4.2 for the $(\{1,2,3\},4)$-projection. We wish to show that, in the original instance, if machine 1 receives some of the jobs, then it receives all jobs. Assume towards a contradiction that machine 1 receives a strict nonempty subset of the jobs, for example:

$$\begin{pmatrix} 1* & 1* & 1 \\ 2 & 2 & 2* \\ 3 & 3 & 3 \end{pmatrix}.$$

Weak monotonicity implies the following assignment in the following instance:

$$\begin{pmatrix} 0.0001* & 0.0001* & 1.01 \\ 2 & 2 & 2 \\ 3 & 3 & 3* \end{pmatrix}$$

(weak monotonicity implies that the third job can be placed on either machine 2 or 3, but not on machine 1). However, this instance is a $(\{3\},2)$-projection of the instance:

$$\begin{pmatrix} 1.01 & 1.01 & 1.01 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{pmatrix}$$

(for example, with parameters $\delta = 0.0001, a_2 = 0.08, a_3 = 0.09$, and taking the decisiveness parameter c to be 10). But this contradicts the inductive assumption (in a $(\{3\},2)$-projection, machine 1 receives all jobs). This shows that, in the original instance, if machine 1 receives some of the jobs, then it receives all of them.

6. 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 $i = 1, \ldots, m-1$).

The proof uses weak monotonicity to reason about the connection between various different instances, related to an ordered instance t. It 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 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 because 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 to take our anonymity assumption into account, we must consider only symmetric equilibria

of indirect mechanisms. 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 nonanonymous 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 [15] 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 nonaffine-maximizers—for example—allocating each job independently using some nonaffine-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, because without the extra conditions many other implementable functions exist.

We should emphasize that while we characterize "many" instances, we certainly do not characterize all of them, and the general characterization question remains open (and very interesting). Christodoulou et al. [6] show such a characterization for two players/machines, which involves "threshold mechanisms." Thus, a conjecture in the spirit of Robert's result would be to show that every mechanism is either a threshold or affine minimizer.

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 rich domain of possible job costs. What happens if the domain is bounded, or being restricted in other meaningful ways? Lavi and Swamy [12] demonstrate 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.

Acknowledgements. The third author is supported in part by grants from the Bi-National Science Foundation (BSF), the Israeli Ministry of Science, and by the Google Inter-University Center for Electronic Markets and Auctions.

References

[1] Aggarwal, G., A. Fiat, A. V. Goldberg, J. D. Hartline, N. Immorlica, M. Sudan. 2005. Derandomization of auctions. *Proc.* 37*th ACM Sympos. Theory of Comput.* (*STOC*), ACM Press, New York.
[2] Archer, A., E. Tardos. 2001. Truthful mechanisms for one-parameter agents. *Proc.* 42*nd IEEE Sympos. Foundations Comput. Sci.* (*FOCS*), IEEE Computer Society, Washington, DC. 482–491.
[3] Bikhchandani, S., S. Chatterji, R. Lavi, A. Mu'alem, N. Nisan, A. Sen. 2006. Weak monotonicity characterizes deterministic dominant-strategy implementation. *Econometrica* **74**(4) 1109–1132.
[4] Christodoulou, G., A. Kovacs. 2010. A deterministic truthful PTAS for scheduling related machines. *Proc.* 21*st ACM-SIAM Sympos. Discrete Algorithms* (*SODA*), SIAM, Philadelphia.
[5] Christodoulou, G., E. Koutsoupias, A. Kovács. 2010. Mechanism design for fractional scheduling on unrelated machines. *ACM Trans. Algorithms* (*TALG*) **6**(2) 1–18.
[6] Christodoulou, G., E. Koutsoupias, A. Vidali. 2008. A characterization of 2-player mechanisms for scheduling. *Proc.* 16*th Eur. Sympos. Algorithms* (*ESA*), Springer LNCS, Berlin.
[7] Christodoulou, G., E. Koutsoupias, A. Vidali. 2009. A lower bound for scheduling mechanisms. *Algorithmica* **55**(4) 729–740.
[8] Dhangwatnotai, P., S. Dobzinski, S. Dughmi, T. Roughgarden. 2008. Truthful approximation schemes for single-parameter agents. *Proc. IEEE* 49*th IEEE Sympos. Foundations Comput. Sci.* (*FOCS*), IEEE Computer Society, Washington, DC.
[9] Dobzinski, S., M. Sundararajan. 2008. On characterizations of truthful mechanisms for combinatorial auctions and scheduling. *Proc.* 9*th ACM Conf. Electronic Commerce* (*ACM-EC*), ACM Press, New York.
[10] Eastman, W. L., S. Even, I. M. Isaacs. 1964. Bounds for the optimal scheduling of n jobs on m processors. *Management Sci.* **11**(2) 268–279.
[11] Koutsoupias, E., A. Vidali. 2007. A lower bound of 1+phi for truthful scheduling mechanisms. *Proc.* 32*nd Internat. Sympos. Math. Foundations Comput. Sci.* (*MFCS*), Springer LNCS, Berlin.
[12] Lavi, R., C. Swamy. 2009. Truthful mechanism design for multidimensional scheduling via cycle monotonicity. *Games and Economic Behavior* **67**(1) 99–124.
[13] Mu'alem, A., M. Schapira. 2007. Setting lower bounds on truthfulness: Extended abstract. *Proc.* 18*th ACM-SIAM Sympos. Discrete Algorithms* (*SODA*), SIAM, Philadephia.
[14] Nisan, N., A. Ronen. 2001. Algorithmic mechanism design. *Games Econom. Behav.* **35**(1–2) 166–196.
[15] Roberts, K. 1979. The characterization of implementable choice rules. J.-J. Laffont, ed. *Aggregation Revelation Preferences*, North Holland, Amsterdam, 321–348.
[16] Yu, C. 2009. Truthful mechanisms for two-range-values variant of unrelated scheduling. *Theoretical Comput. Sci.* **410**(21–23) 2196–2206.