Truthful Mechanism Design for Multi-Dimensional Scheduling via Cycle Monotonicity

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

Chaitanya Swamy
Combinatorics and Optimization
University of Waterloo
cswamy@math.uwaterloo.ca

ABSTRACT

We consider the problem of makespan minimization on m unrelated machines in the context of algorithmic mechanism design, where the machines are the strategic players. This is a *multidimensional* scheduling domain, and the only known positive results for makespan minimization in such a domain are $O(m)$-approximation truthful mechanisms [22, 20]. We study a well-motivated special case of this problem, where the processing time of a job on each machine may either be "low" or "high", and the low and high values are public and job-dependent. This preserves the multidimensionality of the domain, and generalizes the restricted-machines (i.e., $\{p_j, \infty\}$) setting in scheduling. We give a general technique to convert *any* c-approximation algorithm to a $3c$-approximation truthful-in-expectation mechanism. This is one of the few known results that shows how to export approximation algorithms for a multidimensional problem into truthful mechanisms in a *black-box* fashion. When the low and high values are the same for all jobs, we devise a *deterministic* 2-approximation truthful mechanism. These are the *first* truthful mechanisms with non-trivial performance guarantees for a multidimensional scheduling domain.

Our constructions are novel in two respects. First, we do not utilize or rely on explicit price definitions to prove truthfulness; instead we design algorithms that satisfy *cycle monotonicity.* Cycle monotonicity [23] is a necessary and sufficient condition for truthfulness, is a generalization of value monotonicity for multidimensional domains. However, whereas value monotonicity has been used extensively and successfully to design truthful mechanisms in *single-dimensional* domains, ours is the first work that leverages cycle monotonicity in the multidimensional setting. Second, our randomized mechanisms are obtained by first constructing a *fractional* truthful mechanism for a fractional relaxation of the problem, and then converting it into a truthful-in-expectation mechanism. This builds upon a technique of [16], and shows the usefulness of fractional mechanisms in truthful mechanism design.

Categories and Subject Descriptors

F.2 [**Analysis of Algorithms and Problem Complexity**]; J.4 [**Social and Behavioral Sciences**]: *Economics*

General Terms

Algorithms, Economics, Theory

Keywords

Mechanism design, approximation algorithms, scheduling

1. INTRODUCTION

Mechanism design studies algorithmic constructions under the presence of strategic players who hold the inputs to the algorithm. Algorithmic mechanism design has focused mainly on settings were the social planner or designer wishes to maximize the social welfare (or equivalently, minimize social cost), or on auction settings where revenue-maximization is the main goal. Alternative optimization goals, such as those that incorporate fairness criteria (which have been investigated algorithmically and in social choice theory), have received very little or no attention.

In this paper, we consider such an alternative goal in the context of machine scheduling, namely, *makespan minimization.* There are n jobs or tasks that need to be assigned to m machines, where each job has to be assigned to exactly one machine. Assigning a job j to a machine i incurs a load (cost) of $p_{ij} \geq 0$ on machine i, and the load of a machine is the sum of the loads incurred due to the jobs assigned to it; the goal is to schedule the jobs so as to minimize the maximum load of a machine, which is termed the *makespan* of the schedule. Makespan minimization is a common objective in scheduling environments, and has been well studied algorithmically in both the Computer Science and Operations Research communities (see, e.g., the survey [12]). Following the work of Nisan and Ronen [22], we consider each machine to be a *strategic player* or *agent* who privately knows its own processing time for each job, and may misrepresent these values in order to decrease its load (which is its incurred cost). Hence, we approach the problem via mechanism design: the social designer, who holds the set of jobs to be assigned, needs to specify, in addition to a schedule, suitable payments to the players in order to incentivize them to reveal their true processing times. Such a mechanism is called a *truthful mechanism.* The makespan-minimization objective is quite different from the classic goal of *social-welfare maximization*, where one wants to maximize the total welfare (or minimize the total cost) of all players. Instead, it corresponds to maximizing the minimum welfare and the notion of max-min fairness, and appears to be a much harder problem from the viewpoint of mechanism design. In particular, the celebrated VCG [26, 9, 10] family of mechanisms does not apply here, and we need to devise new techniques.

The possibility of constructing a truthful mechanism for makespan minimization is strongly related to assumptions on the players' processing times, in particular, the "dimensionality" of the domain. Nisan and Ronen considered the setting of *unrelated machines* where the p_{ij} values may be arbitrary. This is a *multidimensional domain*, since a player's private value is its entire *vector* of processing times $(p_{ij})_j$. Very few positive results are known for multidimensional domains in general, and the *only* positive results known for multidimensional scheduling are $O(m)$-approximation truthful mechanisms [22, 20]. We emphasize that *regardless of computational considerations*, even the *existence* of a truthful mechanism with a significantly better (than m) approximation ratio is not known for any such scheduling domain. On the negative side, [22] showed that no truthful deterministic mechanism can achieve approximation ratio better than 2, and strengthened this lower bound to m for two specific classes of deterministic mechanisms. Recently, [20] extended this lower bound to randomized mechanisms, and [8] improved the deterministic lower bound.

In stark contrast with the above state of affairs, much stronger (and many more) positive results are known for a special case of the unrelated machines problem, namely, the setting of *related machines*. Here, we have $p_{ij} = p_j/s_i$ for every i, j, where p_j is public knowledge, and the speed s_i is the *only* private parameter of machine i. This assumption makes the domain of players' types *single-dimensional*. Truthfulness in such domains is equivalent to a convenient *value-monotonicity* condition [21, 3], which appears to make it significantly easier to design truthful mechanisms in such domains. Archer and Tardos [3] first considered the related machines setting and gave a randomized 3-approximation truthful-in-expectation mechanism. The gap between the single-dimensional and multidimensional domains is perhaps best exemplified by the fact that [3] showed that there exists a truthful mechanism that always outputs an *optimal schedule*. (Recall that in the multidimensional unrelated machines setting, it is *impossible* to obtain a truthful mechanism with approximation ratio better than 2.) Various follow-up results [2, 4, 1, 13] have strengthened the notion of truthfulness and/or improved the approximation ratio.

Such difficulties in moving from the single-dimensional to the multidimensional setting also arise in other mechanism design settings (e.g., combinatorial auctions). Thus, in addition to the specific importance of scheduling in strategic environments, ideas from multidimensional scheduling may also have a bearing in the more general context of truthful mechanism design for multidimensional domains.

In this paper, we consider the makespan-minimization problem for a special case of unrelated machines, where the processing time of a job is either "low" or "high" on each machine. More precisely, in our setting, $p_{ij} \in \{L_j, H_j\}$ for every i, j, where the L_j, H_j values are publicly known ($L_j \equiv$"low", $H_j \equiv$"high"). We call this model the "job-dependent two-values" case. This model generalizes the classic "restricted machines" setting, where $p_{ij} \in \{L_j, \infty\}$ which has been well-studied algorithmically. A special case of our model is when $L_j = L$ and $H_j = H$ for all jobs j, which we denote simply as the "two-values" scheduling model. Both of our domains are multidimensional, since the machines are *unrelated*: one job may be low on one machine and high on the other, while another job may follow the opposite pattern. Thus, the private information of each machine is a vector specifying which jobs are low and high on it. Thus, they retain the core property underlying the hardness of truthful mechanism design for unrelated machines, and by studying these special settings we hope to gain some insights that will be useful for tackling the general problem.

Our Results and Techniques We present various positive results for our multidimensional scheduling domains.

Our first result is a *general method* to convert *any c-approximation algorithm* for the job-dependent two values setting into a *$3c$-approximation truthful-in-expectation mechanism*. This is one of the very few known results that use an approximation algorithm in a *black-box* fashion to obtain a truthful mechanism for a multidimensional problem. Our result implies that there *exists* a 3-approximation truthful-in-expectation mechanism for the L_j-H_j setting. Interestingly, the proof of truthfulness is not based on supplying explicit prices, and our construction does not necessarily yield efficiently-computable prices (but the allocation rule is efficiently computable). Our second result applies to the two-values setting ($L_j = L$, $H_j = H$), for which we improve both the approximation ratio and strengthen the notion of truthfulness. We obtain a *deterministic* 2-approximation truthful mechanism (along with prices) for this problem. These are the *first* truthful mechanisms with non-trivial performance guarantees for a multidimensional scheduling domain. Complementing this, we observe that even this seemingly simple setting *does not admit truthful mechanisms that return an optimal schedule* (unlike in the case of related machines). By exploiting the multidimensionality of the domain, we prove that no truthful deterministic mechanism can obtain an approximation ratio better than 1.14 to the makespan (irrespective of computational considerations).

The main technique, and one of the novelties, underlying our constructions and proofs, is that we do not rely on explicit price specifications in order to prove the truthfulness of our mechanisms. Instead we exploit certain algorithmic monotonicity conditions that characterize truthfulness to first design an *implementable* algorithm, i.e., an algorithm for which prices ensuring truthfulness exist, and *then* find these prices (by further delving into the proof of implementability). This kind of analysis has been the method of choice in the design of truthful mechanisms for *single-dimensional* domains, where value-monotonicity yields a convenient characterization enabling one to concentrate on the algorithmic side of the problem (see, e.g., [3, 7, 4, 1, 13]). But for multidimensional domains, almost all positive results have relied on *explicit price specifications* in order to prove truthfulness (an exception is the work on unknown single-minded players in combinatorial auctions [17, 7]), a fact that yet again shows the gap in our understanding of multidimensional vs. single-dimensional domains.

Our work is the *first* to leverage monotonicity conditions for truthful mechanism design in arbitrary domains. The monotonicity condition we use, which is sometimes called *cycle monotonicity*, was first proposed by Rochet [23] (see also [11]). It is a generalization of value-monotonicity and completely characterizes truthfulness in every domain. Our methods and analyses demonstrate the potential benefits of this characterization, and show that *cycle monotonicity can be effectively utilized to devise truthful mechanisms for multidimensional domains*. Consider, for example, our first result showing that any c-approximation algorithm can be "exported" to a $3c$-approximation truthful-in-expectation mechanism. At the level of generality of an arbitrary approximation algorithm, it seems unlikely that one would be able to come up with prices to prove truthfulness of the constructed mechanism. But, cycle monotonicity *does allow* us to prove such a statement. In fact, some such condition based only on the underlying algorithm (and not on the prices) seems necessary to prove such a general statement.

The method for converting approximation algorithms into truthful mechanisms involves another novel idea. Our randomized mechanism is obtained by first constructing a truthful mechanism that returns a *fractional schedule*. Moving to a fractional domain allows us to "plug-in" truthfulness into the approximation algorithm in a rather simple fashion, while losing a factor of 2 in the approximation ratio. We then use a suitable randomized rounding procedure to convert the fractional assignment into a random integral assignment. For this, we use a recent rounding procedure of Kumar et al. [14] that is tailored for unrelated-machine scheduling. This preserves truthfulness, but we lose another additive factor equal to the approximation ratio. Our construction uses and extends some observations of Lavi and Swamy [16], and further demonstrates the benefits of fractional mechanisms in truthful mechanism design.

Related Work Nisan and Ronen [22] first considered the makespan-minimization problem for unrelated machines. They gave an m-approximation positive result and proved various lower bounds. Recently, Mu'alem and Schapira [20] proved a lower bound of 2 on the approximation ratio achievable by truthful-in-expectation mechanisms, and Christodoulou, Koutsoupias, and Vidali [8] proved a $(1+\sqrt{2})$-lower bound for deterministic truthful mechanisms.Archer and Tardos [3] first considered the related-machines problem and gave a 3-approximation truthful-in-expectation mechanism. This been improved in [2, 4, 1, 13] to: a 2-approximation randomized mechanism [2]; an FPTAS for any fixed number of machines given by Andelman, Azar and Sorani [1], and a 3-approximation deterministic mechanism by Kovács [13].

The algorithmic problem (i.e., without requiring truthfulness) of makespan-minimization on unrelated machines is well understood and various 2-approximation algorithms are known. Lenstra, Shmoys and Tardos [18] gave the first such algorithm. Shmoys and Tardos [25] later gave a 2-approximation algorithm for the *generalized assignment problem*, a generalization where there is a cost c_{ij} for assigning a job j to a machine i, and the goal is to minimize the cost subject to a bound on the makespan. Recently, Kumar, Marathe, Parthasarathy, and Srinivasan [14] gave a randomized rounding algorithm that yields the same bounds. We use their procedure in our randomized mechanism.

The characterization of truthfulness for arbitrary domains in terms of cycle monotonicity seems to have been first observed by Rochet [23] (see also Gui et al. [11]). This generalizes the value-monotonicity condition for single-dimensional domains which was given by Myerson [21] and rediscovered by [3]. As mentioned earlier, this condition has been exploited numerous times to obtain truthful mechanisms for single-dimensional domains [3, 7, 4, 1, 13]. For convex domains (i.e., each players' set of private values is convex), it is known that cycle monotonicity is implied by a simpler condition, called weak monotonicity [15, 6, 24]. But even this simpler condition has not found much application in truthful mechanism design for multidimensional problems.

Objectives other than social-welfare maximization and revenue maximization have received very little attention in mechanism design. In the context of combinatorial auctions, the problems of maximizing the minimum value received by a player, and computing an envy-minimizing allocation have been studied briefly. Lavi, Mu'alem, and Nisan [15] showed that the former objective cannot be implemented truthfully; Bezakova and Dani [5] gave a 0.5-approximation mechanism for two players with additive valuations. Lipton et al. [19] showed that the latter objective cannot be implemented truthfully. These lower bounds were strengthened in [20].

2. PRELIMINARIES

2.1 The scheduling domain

In our scheduling problem, we are given n jobs and m machines, and each job must be assigned to exactly one machine. In the unrelated-machines setting, each machine i is characterized by a vector of processing times $(p_{ij})_j$, where $p_{ij} \in \mathbb{R}_{\geq 0} \cup \{\infty\}$ denotes i's processing time for job j with the value ∞ specifying that i cannot process j. We consider two special cases of this problem:

1. **The job-dependent two-values case**, where $p_{ij} \in \{L_j, H_j\}$ for every i, j, with $L_j \leq H_j$, and the values L_j, H_j are known. This generalizes the classic scheduling model of *restricted machines*, where $H_j = \infty$.

2. **The two-values case**, which is a special case of above where $L_j = L$ and $H_j = H$ for all jobs j, i.e., $p_{ij} \in \{L, H\}$ for every i, j.

We say that a job j is low on machine i if $p_{ij} = L_j$, and high if $p_{ij} = H_j$. We will use the terms schedule and assignment interchangeably. We represent a deterministic schedule by a vector $x = (x_{ij})_{i,j}$, where x_{ij} is 1 if job j is assigned to machine i, thus we have $x_{ij} \in \{0, 1\}$ for every i, j, $\sum_i x_{ij} = 1$ for every job j. We will also consider randomized algorithms and algorithms that return a fractional assignment. In both these settings, we will again specify an assignment by a vector $x = (x_{ij})_{i,j}$ with $\sum_j x_{ij} = 1$, but now $x_{ij} \in [0, 1]$ for every i, j. For a randomized algorithm, x_{ij} is simply the probability that j is assigned to i (thus, x is a convex combination of integer assignments).

We denote the *load* of machine i (under a given assignment) by $l_i = \sum_j x_{ij} p_{ij}$, and the *makespan* of a schedule is defined as the maximum load on any machine, i.e., $\max_i l_i$.

The goal in the *makespan-minimization problem* is to assign the jobs to the machines so as to minimize the makespan of the schedule.

2.2 Mechanism design

We consider the makespan-minimization problem in the above scheduling domains in the context of mechanism design. Mechanism design studies strategic settings where the social designer needs to *ensure* the cooperation of the different entities involved in the algorithmic procedure. Following the work of Nisan and Ronen [22], we consider the machines to be the *strategic players* or *agents*. The social designer holds the set of jobs that need to be assigned, but does not know the (true) processing times of these jobs on the different machines. Each machine is a selfish entity, that privately knows its own processing time for each job. on a machine incurs a cost to the machine equal to the true processing time of the job on the machine, and a machine may choose to misrepresent its vector of processing times, which are private, in order to decrease its cost.

We consider direct-revelation mechanisms: each machine reports its (possibly false) vector of processing times, the mechanism then computes a schedule and hands out payments to the players (i.e., machines) to compensate them for the cost they incur in processing their assigned jobs. A (direct-revelation) mechanism thus consists of a tuple (x, P): x specifies the schedule, and $P = \{P_i\}$ specifies the payments handed out to the machines, where both x and the P_is are functions of the reported processing times $p = (p_{ij})_{i,j}$. The mechanism's goal is to compute a schedule that has near-optimal makespan with respect to the *true* processing times; a machine i is however only interested in maximizing its own *utility*, $P_i - l_i$, where l_i is its load under the output assignment, and may declare false processing times if this could increase its utility. The mechanism must therefore incentivize the machines/players to truthfully reveal their processing times via the payments. This is made precise using the notion of *dominant-strategy truthfulness.*

Definition 2.1 (Truthfulness) *A scheduling mechanism is truthful if, for every machine i, every vector of processing times of the other machines, p_{-i}, every true processing-time vector p_i^1 and any other vector p_i^2 of machine i, we have:*

$$P_i^1 - \sum_j x_{ij}^1 p_{ij}^1 \geq P_i^2 - \sum_j x_{ij}^2 p_{ij}^1, \tag{1}$$

where (x^1, P^1) and (x^2, P^2) are respectively the schedule and payments when the other machines declare p_{-i} and machine i declares p_i^1 and p_i^2, i.e., $x^1 = x(p_i^1, p_{-i})$, $P_i^1 = P_i(p_i^1, p_{-i})$ and $x^2 = x(p_i^2, p_{-i})$, $P_i^2 = P_i(p_i^2, p_{-i})$.

To put it in words, in a truthful mechanism, no machine can improve its utility by declaring a false processing time, no matter what the other machines declare.

We will also consider *fractional mechanisms* that return a fractional assignment, and *randomized mechanisms* that are allowed to toss coins and where the assignment and the payments may be random variables. The notion of truthfulness for a fractional mechanism is the same as in Definition 2.1, where x^1, x^2 are now fractional assignments. For a randomized mechanism, we will consider the notion of truthfulness in expectation [3], which means that a machine (player) maximizes her *expected* utility by declaring her true processing-time vector. Inequality (1) also defines truthfulness-in-expectation for a randomized mechanism, where P_i^1, P_i^2 now denote the *expected* payments made to player i, x^1, x^2 are the fractional assignments denoting the randomized algorithm's schedule (i.e., x_{ij}^k is the probability that j is assigned to i in the schedule output for (p_i^k, p_{-i})).

For our two scheduling domains, the informational assumption is that the values L_j, H_j are publicly known. The private information of a machine is which jobs have value L_j (or L) and which ones have value H_j (or H) on it. We emphasize that *both of our domains are multidimensional*, since each machine i needs to specify a *vector* saying which jobs are low and high on it.

3. CYCLE MONOTONICITY

Although truthfulness is defined in terms of payments, it turns out that truthfulness actually boils down to a certain algorithmic condition of monotonicity. This seems to have been first observed for multidimensional domains by Rochet [23] in 1987, and has been used successfully in algorithmic mechanism design several times, but for *single-dimensional domains.* However for multidimensional domains, the monotonicity condition is more involved and there has been no success in employing it in the design of truthful mechanisms. Most positive results for multidimensional domains have relied on *explicit price specifications* in order to prove truthfulness. One of the main contributions of this paper is to demonstrate that the monotonicity condition for multidimensional settings, which is sometimes called *cycle monotonicity*, can indeed be effectively utilized to devise truthful mechanisms. We include a brief exposition on it for completeness. The exposition here is largely based on [11].

Cycle monotonicity is best described in the abstract social choice setting: there is a *finite* set A of alternatives, there are m players, and each player has a private type (valuation function) $v : A \mapsto \mathbb{R}$, where $v_i(a)$ should be interpreted as i's value for alternative a. In the scheduling domain, A represents all the possible assignments of jobs to machines, and $v_i(a)$ is the negative of i's load in the schedule a. Let V_i denote the set of all possible types of player i. A mechanism is a tuple $(f, \{P_i\})$ where $f : V_1 \times \cdots \times V_m \mapsto A$ is the "algorithm" for choosing the alternative, and $P_i : V_1 \times \cdots \times V_m \mapsto A$ is the price *charged* to player i (in the scheduling setting, the mechanism *pays* the players, which corresponds to negative prices). The mechanism is truthful if for every i, every $v_{-i} \in V_{-i} = \prod_{i' \neq i} V_{i'}$, and any $v_i, v_i' \in V_i$ we have $v_i(a) - P_i(v_i, v_{-i}) \geq v_i(b) - P_i(v_i', v_{-i})$, where $a = f(v_i, v_{-i})$ and $b = f(v_i', v_{-i})$. A basic question that arises is given an algorithm $f : V_1 \times \cdots \times V_m \mapsto A$, do there exist prices that will make the resulting mechanism truthful? It is well known (see e.g. [15]) that the price P_i can only depend on the alternative chosen and the others' declarations, that is, we may write $P_i : V_{-i} \times A \mapsto \mathbb{R}$. Thus, truthfulness implies that for every i, every $v_{-i} \in V_{-i}$, and any $v_i, v_i' \in V_i$ with $f(v_i, v_{-i}) = a$ and $f(v_i', v_{-i}) = b$, we have $v_i(a) - P_i(a, v_{-i}) \geq v_i(b) - P_i(b, v_{-i})$.

Now fix a player i, and fix the declarations v_{-i} of the others. We seek an assignment to the variables $\{P_a\}_{a \in A}$ such that $v_i(a) - v_i(b) \geq P_a - P_b$ for every $a, b \in A$ and $v_i \in V_i$ with $f(v_i, v_{-i}) = a$. (Strictly speaking, we should use $A' = f(V_i, v_{-i})$ instead of A here.) Define $\delta_{a,b} := \inf\{v_i(a) - v_i(b) : v_i \in V_i, f(v_i, v_{-i}) = a\}$. We can now rephrase the above price-assignment problem: we seek an assignment to the variables $\{P_a\}_{a \in A}$ such that

$$P_a - P_b \leq \delta_{a,b} \quad \forall a, b \in A \tag{2}$$

This is easily solved by looking at the *allocation graph* and applying a standard basic result of graph theory.

Definition 3.1 (Gui et al. [11]) *The allocation graph of f is a directed weighted graph $G = (A, E)$ where $E = A \times A$ and the weight of an edge $b \to a$ (for any $a, b \in A$) is $\delta_{a,b}$.*

Theorem 3.2 *There exists a feasible assignment to (2) iff the allocation graph has no negative-length cycles. Furthermore, if all cycles are non-negative, a feasible assignment is* *obtained as follows: fix an arbitrary node $a^* \in A$ and set P_a to be the length of the shortest path from a^* to a.*

This leads to the following definition, which is another way of phrasing the condition that the allocation graph have no negative cycles.

Definition 3.3 (Cycle monotonicity) *A social choice function f satisfies cycle monotonicity if for every player i, every $v_{-i} \in V_{-i}$, every integer K, and every $v_i^1, \ldots, v_i^K \in V_i$,*

$$\sum_{k=1}^{K} \Big[v_i^k(a_k) - v_i^k(a_{k+1}) \Big] \geq 0$$

where $a_k = f(v_i^k, v_{-i})$ for $1 \leq k \leq K$, and $a_{K+1} = a_1$.

Corollary 3.4 *There exist prices P such that the mechanism (f, P) is truthful iff f satisfies cycle monotonicity.*[1]

We now consider our specific scheduling domain. Fix a player i, p_{-i}, and any $p_i^1, \ldots, p_i^K$. Let $x(p_i^k, p_{-i}) = x^k$ for $1 \leq k \leq K$, and let $x^{K+1} = x^1$, $p^{K+1} = p^1$. x^k could be a $\{0,1\}$-assignment or a fractional assignment. We have $v_i^k(x^k) = -\sum_j x_{ij}^k p_{ij}^k$, so cycle monotonicity translates to $\sum_{k=1}^{K} \big[-\sum_j x_{ij}^k p_{ij}^k + \sum_j x_{ij}^{k+1} p_{ij}^k \big] \geq 0$. Rearranging, we get

$$\sum_{k=1}^{K} \sum_j x_{ij}^{k+1} \big(p_{ij}^k - p_{ij}^{k+1} \big) \geq 0. \qquad (3)$$

Thus (3) "reduces" our mechanism design problem to a concrete *algorithmic problem*. For most of this paper, we will consequently ignore any strategic considerations and focus on designing an approximation algorithm for minimizing makespan that satisfies (3).

4. A GENERAL TECHNIQUE TO OBTAIN RANDOMIZED MECHANISMS

In this section, we consider the case of job-dependent L_j, H_j values (with $L_j \leq H_j$), which generalizes the classical restricted-machines model (where $H_j = \infty$). We show the power of randomization, by providing a *general technique* that converts *any* c-approximation algorithm into a $3c$-approximation, truthful-in-expectation mechanism. This is one of the few results that shows how to export approximation algorithms for a multidimensional problem into truthful mechanisms when the algorithm is given as a black box.

Our construction and proof are simple, and based on two ideas. First, as outlined above, we prove truthfulness using cycle monotonicity. It seems unlikely that for an arbitrary approximation algorithm given only as a black box, one would be able to come up with payments in order to prove truthfulness; but cycle-monotonicity allows us to prove precisely this. Second, we obtain our randomized mechanism by (a) first moving to a fractional domain, and constructing a *fractional* truthful mechanism that is allowed to return fractional assignments; then (b) using a rounding procedure to express the fractional schedule as a convex combination of integer schedules. This builds upon a theme introduced by Lavi and Swamy [16], namely that of using fractional mechanisms to obtain truthful-in-expectation mechanisms.

[1]It is not clear if Theorem 3.2, and hence, this statement, hold if A is not finite.

We should point out however that one cannot simply plug in the results of [16]. Their results hold for social-welfare-maximization problems and rely on using VCG to obtain a fractional truthful mechanism. VCG however does not apply to makespan minimization, and in our case even the existence of a near-optimal *fractional* truthful mechanism is not known. We use the following result adapted from [16].

Lemma 4.1 (Lavi and Swamy [16]) *Let $M = (x, P)$ be a fractional truthful mechanism. Let $\mathcal{A}$ be a randomized rounding algorithm that given a fractional assignment x, outputs a random assignment X such that $\mathrm{E}\big[X_{ij}\big] = x_{ij}$ for all i, j. Then there exist payments P' such that the mechanism $M' = (\mathcal{A}, P')$ is truthful in expectation. Furthermore, if M is individually rational then M' is individually rational for every realization of coin tosses.*

Let $OPT(p)$ denote the optimal makespan (over integer schedules) for instance p. As our first step, we take a c-approximation algorithm and convert it to a $2c$-approximation fractional truthful mechanism. This conversion works even when the approximation algorithm returns only a fractional schedule (satisfying certain properties) of makespan at most $c \cdot OPT(p)$ for every instance p. We prove truthfulness by showing that the fractional algorithm satisfies cycle monotonicity (3). Notice that the alternative-set of our fractional mechanism is *finite* (although the set of all fractional assignments is infinite): its cardinality is at most that of the input-domain, which is at most 2^{mn} in the two-value case. Thus, we can apply Corollary 3.4 here. To convert this fractional truthful mechanism into a randomized truthful mechanism we need a randomized rounding procedure satisfying the requirements of Lemma 4.1. Fortunately, such a procedure is already provided by Kumar, Marathe, Parthasarathy, and Srinivasan [14].

Lemma 4.2 (Kumar et al. [14]) *Given a fractional assignment x and a processing time vector p, there exists a randomized rounding procedure that yields a (random) assignment X such that,*

1. *for any i, j, $\mathrm{E}\big[X_{ij}\big] = x_{ij}$.*
2. *for any i, $\sum_j X_{ij} p_{ij} < \sum_j x_{ij} p_{ij} + \max_{\{j : x_{ij} \in (0,1)\}} p_{ij}$ with probability 1.*

Property 1 will be used to obtain truthfulness in expectation, and property 2 will allow us to prove an approximation guarantee. We first show that any algorithm that returns a fractional assignment having certain properties satisfies cycle monotonicity.

Lemma 4.3 *Let $\mathcal{A}$ be an algorithm that for any input p, outputs a (fractional) assignment x such that, if $p_{ij} = H_j$ then $x_{ij} \leq 1/m$, and if $p_{ij} = L_j$ then $x_{ij} \geq 1/m$. Then $\mathcal{A}$ satisfies cycle-monotonicity.*

Proof. Fix a player i, and the vector of processing times of the other players p_{-i}. We need to prove (3), that is, $\sum_{k=1}^{K} \sum_j x_{ij}^{k+1} \big(p_{ij}^k - p_{ij}^{k+1} \big) \geq 0$ for every $p_i^1, \ldots, p_i^K$, where index $k = K + 1$ is taken to be $k = 1$. We will show that for every job j, $\sum_{k=1}^{K} x_{ij}^{k+1} \big(p_{ij}^k - p_{ij}^{k+1} \big) \geq 0$.

If p_{ij}^k is the same for all k (either always L_j or always H_j), then the above inequality clearly holds. Otherwise we can divide the indices $1, \dots, K$, into maximal segments, where a maximal segment is a maximal set of consecutive indices $k', k'+1, \dots, k''-1, k''$ (where $K+1 \equiv 1$) such that $p_{ij}^{k'} = H_j \geq p_{ij}^{k'+1} \geq \cdots \geq p_{ij}^{k''} = L_j$. This follows because there must be some k such that $p_{ij}^{k} = H_j > p_{ij}^{k-1} = L_j$. We take $k' = k$ and then keep including indices in this segment till we reach a k such that $p_{ij}^{k} = L_j$ and $p_{ij}^{k+1} = H_j$. We set $k'' = k$, and then start a new maximal segment with index $k''+1$. Note that $k'' \neq k'$ and $k''+1 \neq k'-1$. We now have a subset of indices and we can continue recursively. So all indices are included in some maximal segment. We will show that for every such maximal segment $k', k'+1, \dots, k''$, $\sum_{k'-1 \leq k < k''} x_{ij}^{k+1}\big(p_{ij}^{k} - p_{ij}^{k+1}\big) \geq 0$. Adding this for each segment yields the desired inequality.

So now focus on a maximal segment $k', k'+1, \dots, k''-1, k''$. Thus, there is some k^* such that for $k' \leq k < k^*$, we have $p_{ij}^{k} = H_j$, and for $k^* \leq k \leq k''$, we have $p_{ij}^{k} = L_j$. Now the left hand side of the above inequality for this segment is simply $x_{ij}^{k'}(L_j - H_j) + x_{ij}^{k^*}(H_j - L_j) \geq 0$, since $x_{ij}^{k'} \leq \frac{1}{m} \leq x_{ij}^{k^*}$ as $p_{ij}^{k'} = H_j$ and $p_{ij}^{k^*} = L_j$. $\square$

We now describe how to use a c-approximation algorithm to obtain an algorithm satisfying the property in Lemma 4.3. For simplicity, first suppose that the approximation algorithm returns an integral schedule. The idea is to simply "spread" this schedule. We take each job j assigned to a high machine and assign it to an extent $1/m$ on all machines; for each job j assigned to a low machine, say i, we assign $1/m$-fraction of it to the other machines where it is low, and assign its remaining fraction (which is at least $1/m$) to i. The resulting assignment clearly satisfies the desired properties. Also observe that the load on any machine has at most increased by $\frac{1}{m} \cdot$ (load on other machines) $\leq$ makespan, and hence the makespan has at most doubled. This "spreading out" can also be done if the initial schedule is fractional. We now describe the algorithm precisely.

Algorithm 1 Let $\mathcal{A}$ be any algorithm that on any input p outputs a possibly fractional assignment x such that $x_{ij} > 0$ implies that $p_{ij} \leq T$, where T is the makespan of x. (In particular, note that any algorithm that returns an integral assignment has these properties.) Our algorithm, which we call $\mathcal{A}'$, returns the following assignment x^F. Initialize $x_{ij}^F = 0$ for all i, j. For every i, j,

1. if $p_{ij} = H_j$, set $x_{ij}^F = \sum_{i': p_{i'j} = H_j} x_{i'j}/m$;
2. if $p_{ij} = L_j$, set $x_{ij}^F = x_{ij} + \sum_{i' \neq i: p_{i'j} = L_j} (x_{i'j} - x_{ij})/m + \sum_{i': p_{i'j} = H_j} x_{i'j}/m$.

Theorem 4.4 *Suppose algorithm $\mathcal{A}$ satisfies the conditions in Algorithm 1 and returns a makespan of at most $c \cdot OPT(p)$ for every p. Then, the algorithm $\mathcal{A}'$ constructed above is a $2c$-approximation, cycle-monotone fractional algorithm. Moreover, if $x_{ij}^F > 0$ on input p, then $p_{ij} \leq c \cdot OPT(p)$.*

Proof. First, note that x^F is a valid assignment: for every job j, $\sum_i x_{ij}^F = \sum_i x_{ij} + \sum_{i, i' \neq i: p_{ij} = p_{i'j} = L_j} (x_{i'j} - x_{ij})/m = \sum_i x_{ij} = 1$. We also have that if $p_{ij} = H_j$ then $x_{ij}^F = \sum_{i': p_{i'j} = H_j} x_{i'j}/m \leq 1/m$. If $p_{ij} = L_j$, then $x_{ij}^F = x_{ij}(1 - \ell/m) + \sum_{i' \neq i} x_{i'j}/m$ where $\ell = |\{i' \neq i : p_{i'j} = L_j\}| \leq m - 1$; so $x_{ij}^F \geq \sum_{i'} x_{i'j}/m \geq 1/m$. Thus, by Lemma 4.3, $\mathcal{A}'$ satisfies cycle monotonicity.

The total load on any machine i under x^F is at most $\sum_{j: p_{ij} = H_j} \sum_{i': p_{i'j} = H_j} H_j \cdot \frac{x_{i'j}}{m} + \sum_{j: p_{ij} = L_j} L_j \big(x_{ij} + \sum_{i' \neq i} \frac{x_{i'j}}{m}\big)$, which is at most $\sum_j p_{ij} x_{ij} + \sum_{i' \neq i} \sum_j p_{i'j} x_{i'j}/m \leq 2c \cdot OPT(p)$. Finally, if $x_{ij}^F > 0$ and $p_{ij} = L_j$, then $p_{ij} \leq OPT(p)$. If $p_{ij} = H_j$, then for some i' (possibly i) with $p_{i'j} = H_j$ we have $x_{i'j} > 0$, so by assumption, $p_{i'j} = H_j = p_{ij} \leq c \cdot OPT(p)$. $\square$

Theorem 4.4 combined with Lemmas 4.1 and 4.2, gives a $3c$-approximation, truthful-in-expectation mechanism. The computation of payments will depend on the actual approximation algorithm used. Section 3 does however give an explicit procedure to compute payments ensuring truthfulness, though perhaps not in polynomial-time.

Theorem 4.5 *The procedure in Algorithm 1 converts any c-approximation fractional algorithm into a $3c$-approximation, truthful-in-expectation mechanism.*

Taking $\mathcal{A}$ in Algorithm 1 to be the algorithm that returns an LP-optimum assignment satisfying the required conditions (see [18, 25]), we obtain a 3-approximation mechanism.

Corollary 4.6 *There is a truthful-in-expectation mechanism with approximation ratio 3 for the L_j-H_j setting.*

5. A DETERMINISTIC MECHANISM FOR THE TWO-VALUES CASE

We now present a deterministic 2-approximation truthful mechanism for the case where $p_{ij} \in \{L, H\}$ for all i, j. In the sequel, we will often say that j is assigned to a low-machine to denote that j is assigned to a machine i where $p_{ij} = L$. We will call a job j a low job of machine i if $p_{ij} = L$; the low-load of i is the load on i due to its low jobs, i.e., $\sum_{j: p_{ij} = L} x_{ij} p_{ij}$.

As in Section 4, our goal is to obtain an approximation algorithm that satisfies cycle monotonicity. We first obtain a simplification of condition (3) for our two-values $\{L, H\}$ scheduling domain (Proposition 5.1) that will be convenient to work with. We describe our algorithm in Section 5.1. In Section 5.2, we bound its approximation guarantee and prove that it satisfies cycle-monotonicity. In Section 5.3, we compute explicit payments giving a truthful mechanism. Finally, in Section 5.4 we show that no deterministic mechanism can achieve the optimum makespan. Define

$$n_H^{k,\ell} = \big|\{j : x_{ij}^k = 1, p_{ij}^k = L, p_{ij}^\ell = H\}\big| \qquad (4)$$

$$n_L^{k,\ell} = \big|\{j : x_{ij}^k = 1, p_{ij}^k = H, p_{ij}^\ell = L\}\big|. \qquad (5)$$

Then, $\sum_j x_{ij}^{k+1}(p_{ij}^k - p_{ij}^{k+1}) = (n_H^{k+1,k} - n_L^{k+1,k})(H - L)$. Plugging this into (3) and dividing by $(H - L)$, we get the following.

Proposition 5.1 *Cycle monotonicity in the two-values scheduling domain is equivalent to the condition that, for every player i, every p_{-i}, every integer K, and every $p_i^1, \dots, p_i^K$,*

$$\sum_{k=1}^{K} \big(n_H^{k+1,k} - n_L^{k+1,k}\big) \geq 0. \qquad (6)$$

5.1 A cycle-monotone approximation algorithm

We now describe an algorithm that satisfies condition (6) and achieves a 2-approximation. We will assume that L, H are integers, which is without loss of generality.

A core component of our algorithm will be a procedure that takes an integer load threshold T and computes an integer partial assignment x of jobs to machines such that (a) a job is only assigned to a *low machine*; (b) the load on any machine is at most T; and (c) the number of jobs assigned is maximized. Such an assignment can be computed by solving a max-flow problem: we construct a directed bipartite graph with a node for every job j and every machine i, and an edge (j, i) of infinite capacity if $p_{ij} = L$. We also add a source node s with edges (s, j) having capacity 1, and sink node t with edges (i, t) having capacity $\lfloor T/L \rfloor$. Clearly any integer flow in this network corresponds to a valid integer partial assignment x of makespan at most T, where $x_{ij} = 1$ iff there is a flow of 1 on the edge from j to i. We will therefore use the terms assignment and flow interchangeably. Moreover, there is always an integral max-flow (since all capacities are integers). We will often refer to such a max-flow as the max-flow for (p, T).

We need one additional concept before describing the algorithm. There could potentially be many max-flows and we will be interested in the most "balanced" ones, which we formally define as follows. Fix some max-flow. Let $n^i_{p,T}$ be the amount of flow on edge (i, t) (or equivalently the number of jobs assigned to i in the corresponding schedule), and let $n_{p,T}$ be the total size of the max-flow, i.e., $n_{p,T} = \sum_i n^i_{p,T}$. For any $T' \leq T$, define $n^i_{p,T}|_{T'} = \min(n^i_{p,T}, T')$, that is, we "truncate" the flow/assignment on i so that the total load on i is at most T'. Define $n_{p,T}|_{T'} = \sum_i n^i_{p,T}|_{T'}$. We define a *prefix-maximal* flow or assignment for T as follows.

Definition 5.2 (Prefix-maximal flow) *A flow for the above network with threshold T is prefix-maximal if for every integer $T' \leq T$, we have $n_{p,T}|_{T'} = n_{p,T'}$.*

That is, in a prefix-maximal flow for (p, T), if we truncate the flow at some $T' \leq T$, we are left with a max-flow for (p, T'). An elementary fact about flows is that if an assignment/flow x is not a maximum flow for (p, T) then there must be an *augmenting path* $\mathcal{P} = (s, j_1, i_1, \ldots, j_K, i_K, t)$ in the *residual graph* that allows us to increase the size of the flow. The interpretation is that in the current assignment, j_1 is unassigned, $x_{i_\ell j_\ell} = 0$, which is denoted by the forward edges (j_ℓ, i_ℓ), and $x_{i_\ell j_{\ell+1}} = 1$, which is denoted by the reverse edges $(i_\ell, j_{\ell+1})$. Augmenting x using $\mathcal{P}$ changes the assignment so that each j_ℓ is assigned to i_ℓ in the new assignment, which increases the value of the flow by 1. A simple augmenting path does not decrease the load of any machine; thus, one can argue that a prefix-maximal flow for a threshold T always exists. We first compute a max-flow for threshold 1, use simple augmenting paths to augment it to a max-flow for threshold 2, and repeat, each time augmenting the max-flow for the previous threshold t to a max-flow for threshold $t + 1$ using simple augmenting paths.

Algorithm 2 Given a vector of processing times p, construct an assignment of jobs to machines as follows.

1. Compute $T^*(p) = \min\big\{T \geq H, \quad T \text{ multiple of } L :$
$$n_{p,T} \cdot L + (n - n_{p,T}) \cdot H \leq m \cdot T\big\}.$$
Note that $n_{p,T} \cdot L + (n - n_{p,T}) \cdot H - m \cdot T$ is a decreasing function of T, so $T^*(p)$ can be computed in polynomial time via binary search.

2. Compute a prefix-maximal flow for threshold $T^*(p)$ and the corresponding partial assignment (i.e., j is assigned to i iff there is 1 unit of flow on edge (j, i)).

3. Assign the remaining jobs, i.e., the jobs unassigned in the flow-phase, in a greedy manner as follows. Consider these jobs in an arbitrary order and assign each job to the machine with the current lowest load (where the load includes the jobs assigned in the flow-phase).

Our algorithm needs to compute a prefix-maximal assignment for the threshold $T^*(p)$. The proof showing the existence of a prefix-maximal flow only yields a pseudopolynomial time algorithm for computing it. But notice that the max-flow remains the same for any $T \geq T' = n \cdot L$. So a prefix-maximal flow for T' is also prefix-maximal for any $T \geq T'$. Thus, we only need to compute a prefix-maximal flow for $T'' = \min\{T^*(p), T'\}$. This can be be done in polynomial time by using the iterative-augmenting-paths algorithm in the existence proof to compute iteratively the max-flow for the polynomially many multiples of L up to (and including) T''.

Theorem 5.3 *One can efficiently compute payments that when combined with Algorithm 2 yield a deterministic 2-approximation truthful mechanism for the two-values scheduling domain.*

5.2 Analysis

Let $OPT(p)$ denote the optimal makespan for p. We now prove that Algorithm 2 is a 2-approximation algorithm that satisfies cycle monotonicity. This will then allow us to compute payments in Section 5.3 and prove Theorem 5.3.

5.2.1 Proof of approximation

Claim 5.4 *If $OPT(p) < H$, the makespan is at most $OPT(p)$.*

PROOF. If $OPT(p) < H$, it must be that the optimal schedule assigns all jobs to low machines, so $n_{p,OPT(p)} = n$. Thus, we have $T^*(p) = L \cdot \lceil \frac{H}{L} \rceil$. Furthermore, since we compute a prefix-maximal flow for threshold $T^*(p)$ we have $n_{p,T^*(p)}|_{OPT(p)} = n_{p,OPT(p)} = n$, which implies that the load on each machine is at most $OPT(p)$. So in this case the makespan is at most (and hence exactly) $OPT(p)$. □

Claim 5.5 *If $OPT(p) \geq H$, then $T^*(p) \leq L \cdot \lceil \frac{OPT(p)}{L} \rceil \leq OPT(p) + L$.*

PROOF. Let $n_{OPT(p)}$ be the number of jobs assigned to low machines in an optimum schedule. The total load on all machines is exactly $n_{OPT(p)} \cdot L + (n - n_{OPT(p)}) \cdot H$, and is at most $m \cdot OPT(p)$, since every machine has load at most $OPT(p)$. So taking $T = L \cdot \lceil \frac{OPT(p)}{L} \rceil \geq H$, since $n_{p,T} \geq n_{OPT(p)}$ we have that $n_{p,T} \cdot L + (n - n_{p,T}) \cdot H \leq m \cdot T$. Hence, $T^*(p)$, the smallest such T, is at most $L \cdot \lceil \frac{OPT(p)}{L} \rceil$. □

Claim 5.6 *Each job assigned in step 3 of the algorithm is assigned to a high machine.*

PROOF. Suppose j is assigned to machine i in step 3. If $p_{ij} = L$, then we must have $n^i_{p,T^*(p)} = T^*(p)$, otherwise we could have assigned j to i in step 2 to obtain a flow of larger value. So at the point just before j is assigned in step 3, the load of each machine must be at least $T^*(p)$. Hence, the total load *after* j is assigned is at least $m \cdot T^*(p) + L > m \cdot T^*(p)$. But the total load is also at most $n_{p,T^*(p)} \cdot L + (n - n_{p,T^*(p)}) \cdot H \leq m \cdot T^*(p)$, yielding a contradiction. □

Lemma 5.7 *The above algorithm returns a schedule with makespan at most* $OPT(p) + \max\{L, H(1-\frac{1}{m})\} \leq 2 \cdot OPT(p)$.

PROOF. If $OPT(p) < H$, then by Claim 5.4, we are done. So suppose $OPT(p) \geq H$. By Claim 5.5, we know that $T^*(p) \leq OPT(p) + L$. If there are no unassigned jobs after step 2 of the algorithm, then the makespan is at most $T^*(p)$ and we are done. So assume that there are some unassigned jobs after step 2. We will show that the makespan after step 3 is at most $T + H(1 - \frac{1}{m})$ where $T = \min\{T^*(p), OPT(p)\}$. Suppose the claim is false. Let i be the machine with the maximum load, so $l_i > T + H(1 - \frac{1}{m})$. Let j be the last job assigned to i in step 3, and consider the point just before it is assigned to i. So $l_i > T - H/m$ at this point. Also since j is assigned to i, by our greedy rule, the load on all the other machines must be at least l_i. So the total load *after* j is assigned, is at least $H + m \cdot l_i > m \cdot T$ (since $p_{ij} = H$ by Claim 5.6). Also, for *any* assignment of jobs to machines in step 3, the total load is at most $n_{p,T^*(p)} \cdot L + (n - n_{p,T^*(p)}) \cdot H$ since there are $n_{p,T^*(p)}$ jobs assigned to low machines. Therefore, we must have $m \cdot T < n_{p,T^*(p)} \cdot L + (n - n_{p,T^*(p)}) \cdot H$. But we will argue that $m \cdot T \geq n_{p,T^*(p)} \cdot L + (n - n_{p,T^*(p)}) \cdot H$, which yields a contradiction.

If $T = T^*(p)$, this follows from the definition of $T^*(p)$. If $T = OPT(p)$, then letting $n_{OPT(p)}$ denote the number of jobs assigned to low machines in an optimum schedule, we have $n_{p,T^*(p)} \geq n_{OPT(p)}$. So $n_{p,T^*(p)} \cdot L + (n - n_{p,T^*(p)}) \cdot H \leq n_{OPT(p)} \cdot L + (n - n_{OPT(p)}) \cdot H$. This is exactly the total load in an optimum schedule, which is at most $m \cdot OPT(p)$. □

5.2.2 Proof of cycle monotonicity

Lemma 5.8 *Consider any two instances $p = (p_i, p_{-i})$ and $p' = (p'_i, p_{-i})$ where $p'_i \geq p_i$, i.e., $p'_{ij} \geq p_{ij}\ \forall j$. If T is a threshold such that $n_{p,T} > n_{p',T}$, then* every *maximum flow x' for (p', T) must assign* all *jobs j such that $p'_{ij} = L$.*

PROOF. Let $G_{p'}$ denote the residual graph for (p', T) and flow x'. Suppose by contradiction that there exists a job j^* with $p'_{ij^*} = L$ that is unassigned by x'. Since $p'_i \geq p_i$, all edges (j, i) that are present in the network for (p', T) are also present in the network for (p, T). Thus, x' is a valid flow for (p, T). But it is not a max-flow, since $n_{p,T} > n_{p',T}$. So there exists an augmenting path $\mathcal{P}$ in the residual graph for (p, T) and flow x'. Observe that node i must be included in $\mathcal{P}$, otherwise $\mathcal{P}$ would also be an augmenting path in the residual graph $G_{p'}$ contradicting the fact that x' is a max-flow. In particular, this implies that there is a path $\mathcal{P}' \subset \mathcal{P}$ from i to the sink t. Let $\mathcal{P}' = (i, j_1, i_1, \ldots, j_K, i_K, t)$. All the edges of $\mathcal{P}'$ are also present as edges in $G_{p'}$ — all reverse edges $(i_\ell, j_{\ell+1})$ are present since such an edge implies that $x'_{i_\ell j_{\ell+1}} = 1$; all forward edges (j_ℓ, i_ℓ) are present since $i_\ell \neq i$ so $p'_{i_\ell j_\ell} = p_{i_\ell j_\ell} = L$, and $x'_{i_\ell j_{\ell+1}} = 0$. But then there is an augmenting path $(j^*, i, j_1, i_1, \ldots, j_K, i_K, t)$ in $G_{p'}$ which contradicts the maximality of x'. □

Let $\vec{L}$ denote the all-low processing time vector. Define $T^L_i(p_{-i}) = T^*(\vec{L}, p_{-i})$. Since we are focusing on machine i, and p_{-i} is fixed throughout, we abbreviate $T^L_i(p_{-i})$ to T^L. Also, let $p^L = (\vec{L}, p_{-i})$. Note that $T^*(p) \geq T^L$ for every instance $p = (p_i, p_{-i})$.

Corollary 5.9 *Let $p = (p_i, p_{-i})$ be any instance and let x be any prefix-maximal flow for $(p, T^*(p))$. Then, the low-load on machine i is at most T^L.*

PROOF. Let $T^* = T^*(p)$. If $T^* = T^L$, then this is clearly true. Otherwise, consider the assignment x truncated at T^L. Since x is prefix-maximal, we know that this constitutes a max-flow for (p, T^L). Also, $n_{p,T^L} < n_{p^L,T^L}$ because $T^* > T^L$. So by Lemma 5.8, this truncated flow must assign *all* the low jobs of i. Hence, there cannot be a job j with $p_{ij} = L$ that is assigned to i after the T^L-threshold since then j would not be assigned by this truncated flow. Thus, the low-load of i is at most T^L. □

Using these properties, we will prove the following key inequality: for any $p^1 = (p_{-i}, p^1_i)$ and $p^2 = (p_{-i}, p^2_i)$,

$$n_{p^1,T^L} \geq n_{p^2,T^L} - n^{2,1}_H + n^{2,1}_L \tag{7}$$

where $n^{2,1}_H$ and $n^{2,1}_L$ are as defined in (4) and (5), respectively. Notice that this immediately implies cycle monotonicity, since if we take $p^1 = p^k$ and $p^2 = p^{k+1}$, then (7) implies that $n_{p^k,T^L} \geq n_{p^{k+1},T^L} - n^{k+1,k}_H + n^{k+1,k}_L$; summing this over all $k = 1, \ldots, K$ gives (6).

Lemma 5.10 *If $T^*(p^1) > T^L$, then* (7) *holds.*

PROOF. Let $T^1 = T^*(p^1)$ and $T^2 = T^*(p^2)$. Take the prefix-maximal flow x^2 for (p^2, T^2), truncate it at T^L, and remove all the jobs from this assignment that are counted in $n^{2,1}_H$, that is, all jobs j such that $x^2_{ij} = 1$, $p^2_{ij} = L$, $p^1_{ij} = H$. Denote this flow by x. Observe that x is a valid flow for (p^1, T^L), and the size of this flow is *exactly* $n_{p^2,T^2}|_{T^L} - n^{2,1}_H = n_{p^2,T^L} - n^{2,1}_H$. Also none of the jobs that are counted in $n^{2,1}_L$ are assigned by x since each such job j is high on i in p^2. Since $T^1 > T^L$, we must have $n_{p^1,T^L} < n_{p^L,T^L}$. So if we augment x to a max-flow for (p^1, T^L), then by Lemma 5.8 (with $p = p^L$ and $p' = p^1$), all the jobs corresponding to $n^{2,1}_L$ must be assigned in this max-flow. Thus, the size of this max-flow is at least (size of x) $+\, n^{2,1}_L$, that is, $n_{p^1,T^L} \geq n_{p^2,T^L} - n^{2,1}_H + n^{2,1}_L$, as claimed. □

Lemma 5.11 *Suppose $T^*(p^1) = T^L$. Then* (7) *holds.*

PROOF. Again let $T^1 = T^*(p^1) = T^L$ and $T^2 = T^*(p^2)$. Let x^1, x^2 be the complete assignment, i.e., the assignment after both steps 2 and 3, computed by our algorithm for p^1, p^2 respectively. Let $S = \{j :\ x^2_{ij} = 1 \text{ and } p^2_{ij} = L\}$ and $S'' = \{j :\ x^2_{ij} = 1 \text{ and } p^1_{ij} = L\}$. Therefore, $|S''| = |S| - n^{2,1}_H + n^{2,1}_L$ and $|S| = n^i_{p^2,T^2} = n^i_{p^2,T^2}|_{T^L}$ (by Corollary 5.9). Let $T'' = |S''| \cdot L$. We consider two cases.

Suppose first that $T'' \leq T^L$. Consider the following flow for (p^1, T^L): assign to every machine other than i the low-assignment of x^2 truncated at T^L, and assign the jobs in S'' to machine i. This is a valid flow for (p^1, T^L) since the load on i is $T'' \leq T^L$. Its size is equal to $\sum_{i' \neq i} n^{i'}_{p^2,T^2}|_{T^L} + |S''| = n_{p^2,T^2}|_{T^L} - n^{2,1}_H + n^{2,1}_L = n_{p^2,T^L} - n^{2,1}_H + n^{2,1}_L$. The size of the max-flow for (p^1, T^L) is no smaller, and the claim follows.

Now suppose $T'' > T^L$. Since $|S| \cdot L \leq T^L$ (by Corollary 5.9), it follows that $n_L^{2,1} > n_H^{2,1} \geq 0$. Let $\hat{T} = T'' - L \geq T^L$ since T'', T^L are both multiples of L. Let $M = n_{p^2,T^2} - n_H^{2,1} + n_L^{2,1} = |S''| + \sum_{i' \neq i} n^{i'}_{p^2,T^2}$. We first show that

$$m \cdot \hat{T} < M \cdot L + (n - M) \cdot H. \tag{8}$$

Let N be the number of jobs assigned to machine i in x^2. The load on machine i is $|S| \cdot L + (N - |S|) \cdot H \geq |S''| \cdot L - n_L^{2,1} \cdot L + (N - |S|) \cdot H$ which is at least $|S''| \cdot L > \hat{T}$ since $n_L^{2,1} \leq N - |S|$. Thus we get the inequality $|S''| \cdot L + (N - |S''|) \cdot H > \hat{T}$. Now consider the point in the execution of the algorithm on instance p^2 just before the last high job is assigned to i in Step 3 (there must be such a job since $n_L^{2,1} > 0$). The load on i at this point is $|S| \cdot L + (N - |S| - 1) \cdot H$ which is least $|S''| \cdot L - L = \hat{T}$ by a similar argument as above. By the greedy property, every $i' \neq i$ also has at least this load at this point, so $\sum_j p^2_{i'j} x^2_{i'j} \geq \hat{T}$. Adding these inequalities for all $i' \neq i$, and the earlier inequality for i, we get that $|S''| \cdot L + (N - |S''|) \cdot H + \sum_{i' \neq i} \sum_j p^2_{i'j} x^2_{i'j} > m\hat{T}$. But the left-hand-side is exactly $M \cdot L + (n - M) \cdot H$.

On the other hand, since $T^1 = T^L$, we have

$$m \cdot \hat{T} \geq m \cdot T^L \geq n_{p^1,T^L} \cdot L + (n - n_{p^1,T^L}) \cdot H. \tag{9}$$

Combining (8) and (9), we get that $n_{p^1,T^L} > M = n_{p^2,T^2} - n_H^{2,1} + n_L^{2,1} \geq n_{p^2,T^L} - n_H^{2,1} + n_L^{2,1}$. $\square$

Lemma 5.12 *Algorithm 2 satisfies cycle monotonicity.*

Proof. Taking $p^1 = p^k$ and $p^2 = p^{k+1}$ in (7), we get that $n_{p^k,T^L} \geq n_{p^{k+1},T^L} - n_H^{k+1,k} + n_L^{k+1,k}$. Summing this over all $k = 1, \ldots, K$ (where $K + 1 \equiv 1$) yields (6). $\square$

5.3 Computation of prices

Lemmas 5.7 and 5.12 show that our algorithm is a 2-approximation algorithm that satisfies cycle monotonicity. Thus, by the discussion in Section 3, there exist prices that yield a truthful mechanism. To obtain a *polynomial-time mechanism*, we also need to show how to compute these prices (or payments) in polynomial-time. It is not clear, if the procedure outlined in Section 3 based on computing shortest paths in the allocation graph yields a polynomial time algorithm, since the allocation graph has an exponential number of nodes (one for each output assignment). Instead of analyzing the allocation graph, we will leverage our proof of cycle monotonicity, in particular, inequality (7), and simply spell out the payments.

Recall that the utility of a player is $u_i = P_i - l_i$, where P_i is the payment made to player i. For convenience, we will first specify negative payments (i.e., the P_is will actually be *prices charged* to the players) and then show that these can be modified so that players have non-negative utilities (if they act truthfully). Let $\mathcal{H}^i$ denote the number of jobs assigned to machine i in step 3. By Corollary 5.6, we know that all these jobs are assigned to high machines (according to the declared p_is). Let $\mathcal{H}^{-i} = \sum_{i' \neq i} \mathcal{H}^{i'}$ and $n^{-i}_{p,T} = \sum_{i' \neq i} n^{i'}_{p,T}$. The payment P_i to player i is defined as:

$$\begin{aligned} P_i(p) = &-L \cdot n^{-i}_{p,T^*(p)} - H \cdot \mathcal{H}^{-i}(p) \\ &- (H - L)\big(n_{p,T^*(p)} - n_{p,T^L_i(p_{-i})}\big) \end{aligned} \tag{10}$$

We can interpret our payments as equating the player's cost to a careful modification of the total load (in the spirit of VCG prices). The first and second terms in (10), when subtracted from i's load l_i equate i's cost to the total load. The term $n_{p,T^*(p)} - n_{p,T^L_i(p_{-i})}$ is in fact equal to $n^{-i}_{p,T^*(p)} - n^{-i}_{p,T^*(p)}|_{T^L_i(p_{-i})}$ since the low-load on i is at most $T^L_i(p_{-i})$ (by Claim 5.9). Thus the last term in equation (10) implies that we treat the low jobs that were assigned beyond the $T^L_i(p_{-i})$ threshold (to machines other than i) effectively as high jobs for the total utility calculation from i's point of view. It is not clear how one could have conjured up these payments *a priori* in order to prove the truthfulness of our algorithm. However, by relying on cycle monotonicity, we were not only able to argue the *existence* of payments, but also our proof paved the way for actually inferring these payments. The following lemma explicitly verifies that the payments defined above do indeed give a truthful mechanism.

Lemma 5.13 *Fix a player i and the other players' declarations p_{-i}. Let i's true type be p^1_i. Then, under the payments defined in* (10)*, i's utility when she declares her true type p^1_i is at least her utility when she declares any other type p^2_i.*

Proof. Let c^1_i, c^2_i denote i's total cost, defined as the negative of her utility, when she declares p^1, and p^2, respectively (and the others declare p_{-i}). Since p_{-i} is fixed, we omit p_{-i} from the expressions below for notational clarity. The true load of i when she declares her true type p^1_i is $L \cdot n^i_{p^1,T^*(p^1)} + H \cdot \mathcal{H}^i(p^1)$, and therefore

$$\begin{aligned} c^1_i &= L \cdot n_{p^1,T^*(p^1)} + H \cdot (n - n_{p^1,T^*(p^1)}) \\ &\qquad + (H - L)\big(n_{p^1,T^*(p^1)} - n_{p^1,T^L_i}\big) \\ &= n \cdot H - (H - L) n_{p^1,T^L_i} \end{aligned} \tag{11}$$

On the other hand, i's true load when she declares p^2_i is $L \cdot (n^i_{p^2,T^*(p^2)} - n_H^{2,1} + n_L^{2,1}) + H \cdot (\mathcal{H}^i + n_H^{2,1} - n_L^{2,1})$ (since i's true processing time vector is p^1_i), and thus

$$c^2_i = n \cdot H - (H - L) n_{p^2,T^L_i} + (H - L) n_H^{2,1} - (H - L) n_L^{2,1}.$$

Thus, (7) implies that $c^1_i \leq c^2_i$. $\square$

Price specifications are commonly required to satisfy, in addition to truthfulness, *individual rationality*, i.e., a player's utility should be non-negative if she reveals her true value. The payments given by (10) are not individually rational as they actually charge a player a certain amount. However, it is well-known that this problem can be easily solved by adding a large-enough constant to the price definition. In our case, for example, letting $\vec{H}$ denote the vector of all H's, we can add the term $n \cdot H - (H - L) n_{(\vec{H},p_{-i}),T^L_i(p_{-i})}$ to (10). Note that this is a constant for player i. Thus, the new payments are $P'_i(p) = n \cdot H - L \cdot n^{-i}_{p,T^*(p)} - H \cdot \mathcal{H}^{-i}(p) - (H - L)\big(n_{p,T^*(p)} - n_{p,T^L_i(p_{-i})} + n_{(\vec{H},p_{-i}),T^L_i(p_{-i})}\big)$. As shown by (11), this will indeed result in a non-negative utility for i (since $n_{(\vec{H},p_{-i}),T^L_i(p_{-i})} \leq n_{(p_i,p_{-i}),T^L_i(p_{-i})}$ for any type p_i of player i). This modification also ensures the additionally desired normalization property that if a player receives no jobs then she receives zero payment: if player i receives the empty set for some type p_i then she will also receive the empty set for the type $\vec{H}$ (this is easy to verify for our specific algorithm), and for the type $\vec{H}$, her utility equals zero; thus, by truthfulness this must also be the utility of every other declaration that results in i receiving the empty set. This completes the proof of Theorem 5.3.

5.4 Impossibility of exact implementation

We now show that irrespective of computational considerations, there does not exist a cycle-monotone algorithm for the L-H case with an approximation ratio better than 1.14. Let $H = \alpha \cdot L$ for some $2 < \alpha < 2.5$ that we will choose later. There are two machines I, II and seven jobs. Consider the following two scenarios:

Scenario 1. Every job has the same processing time on both machines: jobs 1–5, are L, and jobs 6, 7 are H. Any optimal schedule assigns jobs 1–5 to one machine and jobs 6, 7 to the other, and has makespan $OPT_1 = 5L$. The second-best schedule has makespan at least $Second_1 = 2H + L$.

Scenario 2. If the algorithm chooses an optimal schedule for scenario 1, assume without loss of generality that jobs 6, 7 are assigned to machine II. In scenario 2, machine I has the same processing-time vector. Machine II lowers jobs 6, 7 to L and increases 1–5 to H. An optimal schedule has makespan $2L + H$, where machine II gets jobs 6, 7 and one of the jobs 1–5. The second-best schedule for this scenario has makespan at least $Second_2 = 5L$.

Theorem 5.14 *No deterministic truthful mechanism for the two-value scheduling problem can obtain an approximation ratio better than* 1.14.

Proof. We first argue that a cycle-monotone algorithm cannot choose the optimal schedule in both scenarios. This follows because otherwise cycle monotonicity is violated for machine II. Taking p^1_{II}, p^2_{II} to be machine II's processing-time vectors for scenarios 1, 2 respectively, we get $\sum_j (p^1_{II,j} - p^2_{II,j})(x^2_{II,j} - x^1_{II,j}) = (L-H)(1-0) < 0$. Thus, any truthful mechanism must return a sub-optimal makespan in at least one scenario, and therefore its approximation ratio is at least $\min\{\frac{Second_1}{OPT_1}, \frac{Second_2}{OPT_2}\} \geq 1.14$ for $\alpha = 2.364$. $\square$

We remark that for the $\{L_j, H_j\}$-case where there is a common ratio $r = \frac{H_j}{L_j}$ for all jobs (this generalizes the restricted-machines setting) one can obtain a fractional truthful mechanism (with efficiently computable prices) that returns a schedule of makespan at most $OPT(p)$ for every p. One can view each job j as consisting of L_j sub-jobs of size 1 on a machine i if $p_{ij} = L_j$, and size r if $p_{ij} = H_j$. For this new instance $\tilde{p}$, note that $\tilde{p}_{ij} \in \{1, r\}$ for every i, j. Notice also that any assignment $\tilde{x}$ for the instance $\tilde{p}$ translates to a fractional assignment x for p, where $p_{ij}x_{ij} = \sum_{j': \text{ sub-job of } j} \tilde{p}_{ij'}\tilde{x}_{ij'}$. Thus, if we use Algorithm 2 to obtain a schedule for the instance $\tilde{p}$, equation (6) translates *precisely* to (3) for the assignment x; moreover, the prices for $\tilde{p}$ translate to prices for the instance p. The number of sub-jobs assigned to low-machines in the flow-phase is simply the total *work* assigned to low-machines. Thus, we can implement the above reduction by setting up a max-flow problem that seems to maximize the total work assigned to low machines. Moreover, since we have a fractional domain, we can use a more efficient greedy rule for packing the unassigned portions of jobs and argue that the fractional assignment has makespan at most $OPT(p)$. The assignment x need not however satisfy the condition that $x_{ij} > 0$ implies $p_{ij} \leq OPT(p)$ for arbitrary r, therefore, the rounding procedure of Lemma 4.2 does not yield a 2-approximation truthful-in-expectation mechanism. But if $r > OPT(p)$ (as in the restricted-machines setting), this condition *does* hold, so we get a 2-approximation truthful mechanism.

Acknowledgments

We thank Elias Koutsoupias for his help in refining the analysis of the lower bound in Section 5.4, and the reviewers for their helpful comments.

6. REFERENCES

[1] N. Andelman, Y. Azar, and M. Sorani. Truthful approximation mechanisms for scheduling selfish related machines. In *Proc. 22nd STACS*, 69–82, 2005.

[2] A. Archer. *Mechanisms for discrete optimization with rational agents.* PhD thesis, Cornell University, 2004.

[3] A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. 42nd FOCS*, pages 482–491, 2001.

[4] V. Auletta, R. De-Prisco, P. Penna, and G. Persiano. Deterministic truthful approximation mechanisms for scheduling related machines. In *Proc. 21st STACS*, pages 608–619, 2004.

[5] I. Bezáková and V. Dani. Allocating indivisible goods. In *ACM SIGecom Exchanges*, 2005.

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

[7] P. Briest, P. Krysta, and B. Vocking. Approximation techniques for utilitarian mechanism design. In *Proc. 37th STOC*, pages 39–48, 2005.

[8] G. Christodoulou, E. Koutsoupias, and A. Vidali. A lower bound for scheduling mechanisms. In *Proc. 18th SODA*, pages 1163–1170, 2007.

[9] E. Clarke. Multipart pricing of public goods. *Public Choice*, 8:17–33, 1971.

[10] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[11] H. Gui, R. Muller, and R. V. Vohra. Characterizing dominant strategy mechanisms with multi-dimensional types, 2004. Working paper.

[12] L. A. Hall. Approximation algorithms for scheduling. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems.* PWS Publishing, MA, 1996.

[13] A. Kovács. Fast monotone 3-approximation algorithm for scheduling related machines. In *Proc. 13th ESA*, pages 616–627, 2005.

[14] V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. Approximation algorithms for scheduling on multiple machines. In *Proc. 46th FOCS*, pages 254–263, 2005.

[15] R. Lavi, A. Mu'alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *Proc. 44th FOCS*, pages 574–583, 2003.

[16] R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. In *Proc. 46th FOCS*, pages 595–604, 2005.

[17] D. Lehmann, L. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49:577–602, 2002.

[18] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Prog.*, 46:259–271, 1990.

[19] R. J. Lipton, E. Markakis, E. Mossel, and A. Saberi. On approximately fair allocations of indivisible goods. In *Proc. 5th EC*, pages 125–131, 2004.

[20] A. Mu'alem and M. Schapira. Setting lower bounds on truthfulness. In *Proc. 18th SODA*, 1143–1152, 2007.

[21] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6:58–73, 1981.

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

[23] J. C. Rochet. A necessary and sufficient condition for rationalizability in a quasilinear context. *Journal of Mathematical Economics*, 16:191–200, 1987.

[24] M. Saks and L. Yu. Weak monotonicity suffices for truthfulness on convex domains. In *Proc. 6th EC*, pages 286–293, 2005.

[25] D. B. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–474, 1993.

[26] W. Vickrey. Counterspeculations, auctions, and competitive sealed tenders. *J. Finance*, 16:8–37, 1961.