Truthful Approximation Mechanisms for Scheduling Selfish Related Machines *

Nir Andelman † Yossi Azar ‡ Motti Sorani §

March 22, 2005

Abstract

We consider the problem of scheduling jobs on related machines owned by selfish agents. Previously, Archer and Tardos showed a 2-approximation randomized mechanism which is truthful in expectation only (a weaker notion of truthfulness). We provide a 5-approximation deterministic truthful mechanism, the first deterministic truthful result for the problem.

In case the number of machines is constant, we provide a deterministic Fully Polynomial Time Approximation Scheme (FPTAS) algorithm, and a suitable payment scheme that yields a truthful mechanism for the problem. This result, which is based on converting FPTAS to monotone FPTAS, improves a previous result of Auletta et al, who showed a $(4+\varepsilon)$-approximation truthful mechanism.

1 Introduction

The emergence of the Internet as the platform for distributed computation changed the point of view of the algorithm designer [14, 15]. The old implicit assumption that the participating machines (agents) act as instructed can no longer be taken for granted. As the machines over the Internet are controlled by different users, they are likely to do what is most beneficial to their owners, manipulate the system and lie when it is possible to maximize their own profit. Where optimization problems are concerned, results can be severe and unexpected when false information is introduced to the classic optimization algorithms, due to the selfish behavior of the agents.

In this paper we deal with the problem *Minimum Makespan* for scheduling jobs on related machines, (also known as $Q||C_{max}$). The *system* allocates jobs with arbitrary sizes to the machines, where each machine is owned by an *agent*. The machines have different speeds, known to their

*A preliminary version of this paper appears in the proceedings of the 22nd International Symposium on Theoretical Aspects of Computer Science (STACS), 2005, pp. 69-82.

†andelman@cs.tau.ac.il. School of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel. Research supported in part by the Israel Science Foundation.

‡azar@tau.ac.il. School of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel. Research supported in part by the German Israeli Foundation.

§mottis@tau.ac.il. School of Computer Science, Tel-Aviv University, Tel-Aviv, 69978, Israel.

owner only. At first phase, the agents declare their speeds, then given these bids the system allocates the jobs to the machines. The objective of the system is to minimize the makespan. The classic scheduling problem (when information is complete and accurate) is known to be NP-Complete. Hence research focused on obtaining a polynomial time approximation algorithms for this problem.

The field of *Mechanism Design* provides a scheme to overcome the selfishness of the agents, mainly by paying the agents in order to force them to declare their true properties, thus helping the system to solve the optimization problem correctly. The most famous result in this area is the Vickrey-Clarke-Groves (VCG) Mechanism [6, 16, 9] which applies to *utilitarian* objective only (the sum of the agent's valuations). The Minimum Makespan problem is not utilitarian as we are seeking to minimize the maximum load, not the sum of the loads.

Several optimization problems were re-considered in the context of selfish agents [12]. Even in cases where truthful tools such as VCG are at hand, it turned out that applying them to combinatorial optimization problems is computationally intractable. Ronen and Nisan [13] showed that if the optimal outcome is replaced by the result of a computationally tractable approximation algorithm then the resulting mechanism is no longer necessarily truthful. New attitudes are required to achieve approximation which still retain truthfulness.

Archer and Tardos introduced a framework for designing a truthful mechanism for one-parameter agents [3]. In particular they considered the fundamental problem of scheduling on related machines, and showed a randomized 3-approximation truthful mechanism, later improved to a 2-approximation [4]. Their mechanism utilizes a weaker notion of truthfulness, as it is truthful in expectation only.

1.1 Results in this paper

Our main results are the following

- We show a deterministic 5-approximation truthful mechanism for scheduling jobs on arbitrary number of related machines.
- We show a deterministic truthful FPTAS for scheduling jobs on a fixed number of machines.

All results follow the framework of Archer and Tardos, introducing monotone algorithms together with a *payments scheme* computable in polynomial time.

Our truthful mechanisms are deterministic. Hence truth-telling is a dominant strategy over all possible strategies of an agent. This truthfulness, analogous to universal truthfulness for randomized mechanisms, is stronger than the one use in the 3-approximation randomized mechanism in [3], as the latter is truthful in expectation only.

We also show the existence of truthful deterministic PTAS and FPTAS mechanisms for any fixed number of related machines. Our mechanisms improve the result of the $(4+\varepsilon)$-approximation truthful mechanism for constant number of machines introduced in [5, 2]. We present both mechanisms since our PTAS is fairly simple to implement and may be more efficient than the FPTAS if the required approximation ratio $1+\varepsilon$ is moderate.

1.2 Previous Work

The classic problem of scheduling jobs on parallel related machines was dealt by several approximation approaches. The known basic result of an LPT algorithm which sorts the jobs in non-increasing order of job size, then allocates a job to the machine which will be least busy afterwards is 2-approximation [8]. An FPTAS was first introduced by Horowitz and Sahni for the case where the number of machines is constant. Their approach was based on rounding an exact solution by dynamic programming [11]. Finally, in the late 80's Hochbaum and Shmoys introduced a PTAS for the general case of an arbitrary number of machines [10, 7]. Since the problem is strongly NP-Complete, no FPTAS is possible for the general case, and their result remains the best possible, unless P=NP.

Scheduling with selfish agents was first analyzed by Ronen and Nisan. Their results mainly concern scheduling on unrelated machines, known also as $R||C_{max}$. Our problem was first tackled by Archer and Tardos [3] who showed that the former known approximation algorithms for the problem are not truthful. They introduced a truthful randomized mechanism which achieves a 3-approximation to the problem. This approach achieves truthfulness with respect to the *expected profit* only. Thus it possible that even though the expected profit is maximized when telling the truth, there might exist a better (untruthful) strategy.

The first deterministic result is due to Auletta et al [5]. They show a deterministic truthful mechanism which is $(4+\varepsilon)$-approximation for any *fixed* number of machines. The case of arbitrary number of machines remained open previous to our paper.

A different approach by Nisan and Ronen introduces another model in which the mechanism is allowed to observe the machines process their jobs and compute the payments afterwards. Using these *mechanisms with verification* [12] allows application of penalty on lying agents, and was shown to cope well with the existing known approximation algorithms.

2 Preliminaries

We consider the problem of scheduling jobs on related machines. We are given a number of machines, m, and a job sequence with sizes $\sigma = (p_1, p_2, ..., p_n)$. Each machine, owned by an agent, has a *speed* s_i known only to its agent. Alternatively, the secret (sometimes called *type*) of each agent is $t_i = 1/s_i$ which is the number of time units required to process one unit of work (or the *cost per unit of work*). Thus the processing time of job p_j on machine i is $p_j t_j$. The *work* of machine i, denoted by w_i is given by the sum of the processing time of jobs assigned to it (the total work assigned to it). We assume a machine incurs a cost proportional to the total processing time spent. The output is an assignment of jobs to machines. The mechanism's goal is to minimize the maximum completion time over all machines.

A *mechanism* for this problem is a pair $M = (A, P)$, where A is an algorithm to allocate jobs to the machines (agents) and P is a payment scheme. The mechanism asks each agent to report its bid (their cost per unit of work). Based on these reports, the mechanism uses A to construct an allocation, and pays according to P.

A *strategy* for agent i is to declare a value b_i as its cost per unit of work (which in principle can be larger or smaller than t_i). Let b to be the vector of bids, and b_{-i} denote the vector of bids not

including b_i, i.e. $(b_1, b_2, .., b_{i-1}, b_{i+1}, ..., b_m)$. Denote by (b_{-i}, α) the vector generated by inserting the value α to the vector b_{-i}. Notice that if we view b as a sorted vector, then (b_{-i}, α) corresponds also to a sorted vector (thus the index of α might be different than i).

The output of the algorithm $o(b)$ is an allocation of the jobs to the agents, and the profit of agent i is defined as $profit_i(t_i, b) = P_i(b) - t_i w_i(o(b))$. A strategy b_i is *(weakly) dominant* for agent i if bidding b_i always maximizes his profit, i.e. $profit_i(t_i, (b_{-i}, b_i)) \geq profit_i(t_i, (b_{-i}, b'_i))$ for all b_{-i}, b'_i. A mechanism is *truthful* if truth-telling is a dominant strategy for each agent (i.e. t_i is a dominant strategy for all i)

We assume w.l.o.g that the vector of bids b is sorted in non-increasing order of speed (non-decreasing order of cost per unit of work), breaking ties by the original index.

An algorithm A is a c-approximation algorithm if for every instance (σ, t) , $cost(A, \sigma, t) \leq c \cdot opt(\sigma, t)$. For our problem the cost is the maximum completion time. A c-approximation mechanism is one whose output algorithm is an c-approximation. A PTAS (Polynomial-time approximation scheme) is a family of algorithms such that for every $\varepsilon > 0$ there exists a $(1 + \varepsilon)$-approximation algorithm. If the running time is also polynomial in $1/\varepsilon$, the family of algorithms is an FPTAS.

A vector $(v_1, v_2, .., v_n)$ is *lexicographically smaller* than $(u_1, u_2, .., u_n)$ if for some i, $v_i < u_i$ and $v_k = u_k$ for all $1 \leq k < i$.

2.1 Monotone Algorithms

Archer and Tardos showed necessary and sufficient conditions to obtain a truthful mechanism [3].

Definition 2.1 *Fixing the other agents bids b_{-i}, we define the work-curve for agent i as $w_i(b_{-i}, b_i)$, namely a single-variable function of b_i. A work-curve is decreasing if $w_i(b_{-i}, b_i)$ is a decreasing function of b_i for all b_{-i}.*

A decreasing work-curve means that when an agent bids lower (saying he is faster) more or equal amount of work should be allocated to his machine, given that the other agents' bids are fixed. A *monotone algorithm* is an algorithm that produces an assignment which preserves the decreasing work-curve property for all agents. Several classic approximation algorithms fail to keep this monotonicity, among them the LPT algorithm, and the classic PTAS of Horowitz and Sahni [3, 5].

Definition 2.2 *A mechanism satisfies the voluntary participation condition if agents who bid truthfully never incur a net loss, i.e. $profit_i(t_i, (b_{-i}, t_i)) \geq 0$ for all agents i, true values t_i and other agents' bids b_{-i}.*

Theorem 2.1 *[3] A mechanism is truthful and admits a voluntary participation if and only if the work-curve of each agent is decreasing, $\int_0^\infty w_i(b_{-i}, u)du < \infty$ for all i, b_{-i} and the payments in this case should be*

$$P_i(b_{-i,}b_i) = b_i w_i(b_{-i}, b_i) + \int_{b_i}^{\infty} w_i(b_{-i}, u)du \tag{1}$$

Input: a job sequence σ , and a non-decreasing sorted bids vector $b = (b_1, b_2, .., b_m)$

Output: An allocation of jobs to the m machines

1. Set $d_1 = \frac{5}{8} b_1$
2. For $j \geq 2$, round up the bids to the closest value of $b_1 \cdot 2.5^i$, which is larger than the original bid, i.e. $d_j = b_1 \cdot 2.5^i$, where $i = \left\lfloor log_{2.5} \frac{b_j}{b_1} \right\rfloor + 1$
3. Sort the job sequence in non-increasing order
4. Calculate the valid fractional assignment for the job sequence σ given the new bids-vector d. Let T^f be the value of the fractional solution.
5. For each machine $j = 1 \ldots m$, assign jobs in non-increasing order of job-size to machine j (using bid d_j), until machine j exceeds threshold T^f (or equals it)
6. Return the assignment

Figure 1: Monotone-RF

Therefore in order to achieve a truthful mechanism we need to design a monotone algorithm, and use the payment scheme as in (1). Since truthfulness is reduced to designing a monotone algorithm, we may assume, for the sake of the monotonicity proof, that the bids are equal to the real speeds.

3 Truthful Approximation for arbitrary number of machines

A classic approximation algorithm for the problem, forms a "valid" fractional allocation of the jobs to the machines, and then uses a simple rounding to get a 2-approximation for the problem. In [3] it has been shown that this simple algorithm is not monotone, thus not truthful.

The main result of this section is a deterministic monotone algorithm which is based on the fractional assignment. Algorithm *Monotone-RF (Monotone Rounded Fractional)*, shown in figure 1, is shown to be a 5-approximation algorithm.

Given a threshold T, we can treat the machines as bins of size T/b_i. A *fractional assignment* of the jobs to the machines, is a partition of each job j into pieces whose sizes sum to p_j and an assignment of these pieces to the machines (bins). A fractional assignment is *valid* if each bin is large enough to contain the sum of all pieces assigned to it, and for every piece assigned to it, it is capable of containing the entire job the piece belongs to. The smallest threshold for which there exist a valid fractional assignment, T^f is a lower bound to the optimal solution, and can be calculated exactly in the following manner (see [3]):

$$T^f = \max_j \min_i \max \{b_i p_j, \frac{\sum_{k=1}^{j} p_k}{\sum_{l=1}^{i} \frac{1}{b_l}}\} \tag{2}$$

This valid fractional assignment with respect to this threshold is obtained by sorting the jobs in non-increasing order, and allocating them to the machines (ordered in non-increasing order of their speeds). Some jobs are sliced between two machines when the threshold is exceeded in the middle of a job.

An important property of the valid fractional assignment is the monotonicity of T^f: as we increase the speed of a machine, T^f is not increased. Let $T^f(b_{-i}, b_i)$ be the the smallest threshold for which there exist a valid fractional assignment, given the bids vector (b_{-i}, b_i).

Observation 3.1 *[3]* $T^f(b_{-i}, \alpha b_i) \leq \alpha T^f(b_{-i}, b_i)$ *for all* $\alpha > 1$ *and* i.

Lemma 3.1 *For any machine* i *which is not the fastest* $(i > 1)$*, and for any rounded bids vector* d*:*

$$T^f(d_{-i}, d_i) \leq T^f(d_{-i}, \beta) \leq \frac{5}{4} T^f(d_{-i}, d_i)$$

for all $\beta \geq d_i$

Proof: The first inequality is straight-forward as the allocation for (d_{-i}, β) is also a valid fractional assignment for (d_{-i}, d_i), given any fixed threshold T.

As for the second inequality, we generate a valid fractional assignment which allocates zero work to machine i. This allocation would use a threshold at most $\frac{5}{4} T^f(d_{-i}, d_i)$. Since this allocation is a valid fractional assignment for (d_{-i}, β), the minimal threshold for (d_{-i}, β) might only be smaller than the generated one.

To form an allocation which does not use machine i, for every $2 \leq j \leq i$ take all the pieces previously assigned to machine j and assign them to machine $(j-1)$. The first machine is now allocated the pieces originally assigned to the second machine, along with its original assignment. Since the algorithm assures that $4d_1 \leq d_2$, the assignment is clearly valid, with a threshold which is at most $\frac{5}{4} T^f(d_{-i}, d_i)$. ■

We note that Lemma 3.1 holds for rounded bids vectors created by Monotone-RF, but does not hold in general. The following lemmata consider several scenarios in which machine i slows down. We denote by d' the rounded bid vector obtained after the machine's slowdown. Let i' be the index of the slowing-down machine in d'; Notice that i' might be different than i. We denote by w'_j the total work allocated to machine j given the new bids vector d'. We denote by v_i the rounded speed of machine i, i.e. $v_i = 1/d_i$.

Lemma 3.2 *Using Algorithm Monotone-RF, when machine* i *which is not the fastest slows down, the total amount of work assigned to the machines faster than it can not decrease. i.e.*

$$\sum_{k=1}^{i-1} w_k \leq \sum_{k=1}^{i-1} w'_k \leq \sum_{k=1}^{i'-1} w'_k$$

Proof: If the rounded bid of machine i is the same as before the slowdown, the assignment is not changed. Thus we consider the case where the new rounded bid is different than the one before

the slowdown. Let β be the rounded bid of machine i where $\beta > d_i$. Let i' be the new index of the slowing machine in d'. Clearly $i \leq i'$.

By Lemma 3.1, $T^f(d_{-i}, \beta) \geq T^f(d_{-i}, d_i)$, i.e. the new threshold used by algorithm Monotone-RF can only increase due to the slowdown. By induction the index of the last job assigned to each of the first $i-1$ machines can be equal, or higher after the slowdown. Thus the total amount of work assigned to the first $i-1$ machines is the same or higher, and the amount of work assigned to the first $i'-1$ machines can only be higher than that. ■

Lemma 3.3 *If the fastest machine slows down yet remains the fastest, the amount of work assigned to it can only decrease.*

Proof: We observe how the bin size of the first machine changes as its speed decreases gradually. As long as the value of $\lfloor \log_{2.5} \frac{b_j}{b_1} \rfloor$ does not change for all $j \geq 2$, all rounded speeds change proportionally, i.e. there is some constant $c > 1$ such that $d' = c \cdot d$. Therefore, the same fractional assignment is calculated (with a new threshold of cT^f) with the same sizes for bins. In the breakpoints where the rounded bid of at least one machine is cut down by 2.5, by Observation 3.1 the threshold cannot increase, and therefore the bin size of the first machine can only decrease.

Since the fastest machine is always assigned the first jobs, a decrease in its bin size can only decrease the number of jobs assigned to it, and therefore the amount of work assigned to it in the integral assignment also decreases. ■

Definition 3.1 *Given an assignment of jobs by algorithm Monotone-RF, we classify the machines in the following way:*

Full machine *a machine (bin) which the total processing time of the jobs assigned to it is at least its threshold.*

Empty machine *a machine with no jobs allocated to it*

Partially-full machine *a non-empty machine (bin) which is not full. (There is at most one partially-full machine)*

Lemma 3.4 *When machine i decreases its speed (increases its bid) the total work allocated to it by algorithm Monotone-RF can not increase.*

Proof: Lemma 3.3 proves the claim when machine i is the fastest machine and remains the fastest. If machine i is not the fastest machine but its rounded bid d_i does not change, then the slowdown has no effect since the same assignment is generated. It remains to prove the claim for the breakpoints, which are when the fastest machine becomes the non-fastest, and when the rounded bid is multiplied by 2.5. We prove the claim for each step, thus the claim holds for the entire slowdown.

Consider the following cases for the class of machine i before the slowdown:

1. Machine i is the fastest machine ($i = 1$), but after the slowdown another machine j becomes the fastest - we observe the breakpoint where both machines have the same speed and add an artificial stage to the slowdown where the title of the fastest machine passes from i to j

(without having the speeds change). The same threshold is calculated in both cases, only the order of the machines changes. The amount of work allocated to machine i when it is considered the fastest is at least $\frac{8}{5} \cdot v_1 \cdot T^f$, while after machine j becomes the fastest it is at most $2 \cdot \frac{v_1}{2.5} \cdot T^f = \frac{4}{5} \cdot v_1 \cdot T^f$, and therefore decreases.

2. Machine i is a full machine which is not the fastest - the threshold used for assigning jobs to the machine is T^f. Due to Lemma 3.1, $T^f(d) \leq T^f(d') \leq \frac{5}{4}T^f(d)$, where d is the rounded bids vector, and d' is the rounded vector after the slowdown. Before the slowdown the amount of work allocated to it was at least $T^f \cdot v_i$, whereas after slowing down it can become at most $2 \cdot (\frac{5}{4} \cdot T^f) \cdot \frac{v_i}{2.5} = T^f \cdot v_i$. If the machine became partially-full or empty after slowing down, the amount of work allocated to it can only be smaller.

3. Machine i is partially-full - if it becomes empty then the claim is trivial, otherwise some jobs are allocated to machine i. Let $i' \geq i$ be the new index of the machine in the sorted order. Due to Lemma 3.2 the amount of work allocated to machines with a lower index than i' can be no less than the amount before the slowdown (i.e. $\sum_{k=1}^{i-1} w_k \leq \sum_{k=1}^{i'-1} w'_k$), thus leaving less work to be allocated to machine i.

4. Machine i is empty - The machine stays empty due to Lemma 3.2.

■

Lemma 3.4 shows that the work-curve of agent i is non-increasing. Hence the following theorem is immediate:

Theorem 3.5 *Algorithm Monotone-RF provides monotone assignment. Hence A Mechanism Design based on Algorithm Monotone-RF and payment scheme as in (1) is truthful.*

We now analyze the approximation provided by algorithm *Monotone-RF*.

Denote by $k^f(i)$ the index of the last job (or a fraction of a job) assigned to machine i in the fractional assignment. Respectively let $k(i)$ be the index of the last job assigned to machine i by *Monotone-RF*.

Lemma 3.6 *for all i, $k^f(i) \leq k(i)$*

Proof: By induction. The claim clearly holds for $i = 1$ since $T_1 \geq T^f$. Assume the argument is true for machine i. By induction hypothesis $k^f(i) \leq k(i)$. Since allocation is done in non-increasing order of job size, the first job to be allocated to $i+1$ by our algorithm might be only smaller than the one allocated by the fractional assignment. Moreover, since the allocation exceeds the fractional threshold, at least the same number of jobs will be assigned to machine i. Thus $k^f(i+1) \leq k(i+1)$.
■

Theorem 3.7 *Algorithm Monotone-RF is a 5-approximation.*

Proof: Lemma 3.6 assures that at the end of the run of algorithm *Monotone-RF* all jobs are allocated. Since the speeds were decreased by at most a factor of 2.5, the threshold $T^f(d)$ may be at most 2.5 times the value of the optimal allocation using the unrounded speeds. Since the speed of the fastest machine is increased by a factor of 1.6, the amount of work assigned to the fastest machine in the fractional solution may be at most $1.6 \cdot 2.5 = 4$ times the value of the optimal allocation.

In the integral solution, since the amount of work assigned to the first machine can exceed the bin's size by at most the size of the second bin, and since the first bin is at least 4 times larger than the second bin, the load on the fastest machine can be at most $1.25 \cdot T^f(d)$, and therefore the load on this machine is at most $1.25 \cdot 4 = 5$ times the optimal. For any other machine, the last job can exceed the threshold by at most $T^f(d)$, and therefore the load on any other machine is at most $2 \cdot T^f(d)$, which is at most $2 \cdot 2.5 = 5$ times the optimal. Therefore, a 5-approximation is achieved. ■

To conclude, we need to show that calculating the payments given by (1) can be done in polynomial time. We analyze the number of breakpoints of the integral in that expression. According to Lemma 3.6 the work curve for machine i is zeroed furthermost when the valid fractional assignment does not use machine i. There is no use in assigning jobs to a machine when its bid β is too high even for the smallest job, i.e. $\beta p_n > T^f$. Using the higher bound $T^f \leq np_1d_1 < np_1b_1$, we get a zero assignment for $\beta \geq n\frac{p1}{pn}b_1$. The only exception is when the integral is calculated for the fastest machine, where we get a higher bound of $\beta \geq n\frac{p1}{pn}b_2$. While $\beta \leq b_2$, there is a breakpoint whenever $b_j = 2.5^i\beta$, for some i and for any machine $j > 1$. Therefore, for each factor of 2.5, there are at most $m-1$ breakpoints (one for each of the other machines), while for $\beta > b_2$, there is one breakpoint for each step.

Thus the number of iterations will be $O(\log_{2.5} n\frac{p1}{pn} + m\log_{2.5}\frac{b_2}{b_1})$ for the fastest machine, $O(m\log_{2.5} n\frac{p1}{pn}\frac{b_2}{b_1})$ in total, which is polynomial in the input size.

4 Truthful PTAS-Mechanism for any fixed number of machines

We now show a truthful mechanism for any fixed number of machines. Due to simplicity of presentation, we normalize the sizes of the jobs such that the total size of all jobs is one, $\sum_{j=1}^{n} p_j = 1$ (as they are known to the mechanism).

Based on the payments scheme as in (1), it is enough to show a monotone PTAS algorithm. The algorithm, *Monotone-PTAS*, is shown in figure 2. This algorithm shares similar ideas of the PTAS variant of Alon et al [1].

Theorem 4.1 *A Mechanism Design based on Algorithm Monotone-PTAS and payment scheme as in (1) is truthful.*

Proof: It is suffice to show that Algorithm Monotone-PTAS is monotone. Notice that the job sizes in the instance $\sigma^\sharp$ were generated independently from the bids of the agents. It was shown in [3] that the optimal minimum-lexicographic solution is monotone. ■

Theorem 4.2 *The algorithm Monotone-PTAS achieves a Polynomial Time Approximation Scheme (PTAS)*

Proof: We first show that the algorithm is polynomial. The construction of $\sigma^\sharp$ takes a linear time. As for the rest - the construction of $\sigma^\sharp$ ensures that the minimal job size is $\frac{1}{2} \cdot \frac{\varepsilon^2}{m^2}$. Thus the total number of jobs is no more than $\frac{2m^2}{\varepsilon^2}$, a constant. Solving $\sigma^\sharp$ exactly on m machines while enumerating all the possible allocations takes a constant time.

We now analyze the quality of the approximation. First assume, for the purpose of analysis , that both Opt and our algorithm are not using "slow" machines, i.e. machines whose speed is less than $s_{max} \cdot \frac{\varepsilon}{m}$, where s_{max} is the maximal speed. Let T'_{opt} be the optimal solution for this instance, and T' our solution. Since we solve for chunks whose size is no more than $\frac{\varepsilon^2}{m^2}$, unlike Opt who solves for the original sizes, we can suffer an addition in processing time of maximum $\frac{\frac{\varepsilon^2}{m^2}}{s_{max} \cdot \frac{\varepsilon}{m}} = \frac{\varepsilon}{m \cdot s_{max}}$ (i.e. an additional chunk on the slowest machine used). A lower bound on the optimal solution is $T'_{opt} \geq \frac{1}{m \cdot s_{max}}$, Thus $T' \leq (1+\varepsilon)T'_{opt}$.

We now compare T'_{opt} to performance of Opt when the "slow" machines restriction is removed, namely T_{opt}. The total work done by the "slow" machines in opt is bounded above by $s_{max} \cdot \varepsilon T_{opt}$. If we move this amount of work to the fastest machine we pay maximum extra processing time of εT_{opt}, thus $T'_{opt} \leq (1+\varepsilon)T_{opt}$. Combining these two lower bounds we get that $T \leq T' \leq (1+\varepsilon)T'_{opt} \leq (1+\varepsilon)^2 \cdot T_{opt} \leq (1+3\varepsilon)T_{opt}$ for any $\varepsilon < \frac{1}{2}$, a PTAS. ■

To conclude, we need to show that calculating the payments given by (1) can be done in polynomial time. Notice that the integral in this expression has a constant number of breakpoints

Input: a job sequence σ, a bids vector $b = (b_1, b_2, .., b_m)$ and parameter ε

Output: An allocation of jobs to the m machines that achieves a $(1+3\epsilon)$-approximation

1. Construct a new instance $\sigma^\sharp$ based on the original job instance, as follows:
   (a) sort the job sequence in non-increasing order
   (b) $\sigma^\sharp = \{p_j \in \sigma | p_j \geq \frac{\varepsilon^2}{m^2}\}$
   (c) merge the rest of jobs in a greedy manner to chunks of size in the range $[\frac{1}{2} \cdot \frac{\varepsilon^2}{m^2}, \frac{\varepsilon^2}{m^2}]$ and add them to $\sigma^\sharp$
2. Solve Minimum Makespan *exactly* with the instance $(\sigma^\sharp, b)$ to obtain the optimal solution. If several optimal allocations exist, choose the one with the lexicographically minimum schedule (where the machines are ordered according to some external machine-id)
3. Return the same allocation achieved on $\sigma^\sharp$. A small job is allocated to the same machine which his chunk has been allocated.

Figure 2: Monotone PTAS

(the number of all possible allocations to agent i) thus calculating the payments can be done in constant time.

5 Truthful FPTAS-Mechanism for any fixed number of machines

We now show another truthful mechanism for any fixed number of machines. The mechanism uses a c-approximation algorithm as a black box, to generate a $c(1+\epsilon)$-approximation monotone algorithm. Using an FPTAS as the black box (for example, the FPTAS of Horowitz and Sahni [11]) outputs a monotone FPTAS. Adding a payments scheme as in (1), ensures truthful mechanism. The algorithm, *Monotone-Black-Box*, is shown in figure 3.

Input: a non decreasing sorted job sequence σ, a bids vector $b = (b_1, b_2, .., b_m)$, a parameter ε and a black box, which is a polynomial time c-approximation.

Output: An allocation of jobs to the m machines that achieves a $c(1+\epsilon)$-approximation

1. Construct a new bid vector $d = (d_1, d_2, \ldots, d_m)$, in the following way:
   (a) round up each bid to the closest value of $(1+\epsilon)^i$
   (b) normalize the bids such that $d_1 = 1$
   (c) for each bid $d_j = (1+\epsilon)^i$, if $i > l+1$ where $l = \lceil \log_{1+\epsilon} cn \cdot \frac{p_1}{p_n} \rceil$ then set $d_j = (1+\epsilon)^{l+1}$
2. Enumerate over all possible vectors $d' = ((1+\epsilon)^{i_1}, (1+\epsilon)^{i_2}, \ldots, (1+\epsilon)^{i_m})$, where $i_j \in \{0, 1, \ldots, l+1\}$. For each vector:
   (a) apply the black box algorithm to d'
   (b) sort the output assignment such that the i-th fastest machine in d' will get the i-th largest amount of work
3. Test all the sorted assignments on d, and return the one with the minimal makespan. In case of a tie, choose the assignment with the lexicographically maximum schedule (i.e. allocating more to the faster machines)

Figure 3: Monotone Black Box

Theorem 5.1 *Algorithm Monotone Black Box is a $c(1+\epsilon)$ approximation algorithm.*

Proof: The output assignment is a c-approximation for the vector d, since d is tested in the enumeration, and since sorting the assignment can only improve the makespan. As for the original bid vector b, rounding the bids adds a multiplicative factor of $1+\epsilon$ to the approximation ratio. Normalizing the vector has no effect, as well as trimming the largest bids, since any non zero assignment to a machine with a bid of at least $(1+\epsilon)^l$ cannot be a c-approximation, since the load on that machine will be more than c times the load of assigning all the jobs to the fastest machine. ■

Theorem 5.2 *If the black box in Algorithm Monotone Black Box is an FPTAS then the algorithm itself is also an FPTAS.*

Proof: By Theorem 5.1, the algorithm is a $(1+\epsilon)^2$ approximation. It remains to prove that the running time is polynomial in the input, including $\frac{1}{\epsilon}$. In each iteration of the enumeration, applying the black box, sorting the output assignment and testing it on the vector d can be completed in polynomial time, by the assumption that the black box is an FPTAS. The size of the enumeration is $O(l^m)$, where m is a constant and l is polynomial in the input. ■

Theorem 5.3 *Algorithm Monotone Black Box is monotone.*

Proof: We prove that if a machine j raises its bid (lowers its speed) then the amount of work assigned to it cannot increase. We increment the bid in steps, such that in each step the power of $1+\epsilon$ that equals the rounded bid rises by one. We prove the claim for a single step, and therefore, the claim also holds for the entire increment.

We first assume that d_j is not the unique fastest machine (i.e., there is a machine $k \neq j$ such that $d_k = 1$). If $d_j \geq (1+\epsilon)^l$ then by the proof of Theorem 5.1, the assignment to machine j must be null, otherwise the approximation ratio is not achieved. Clearly, by raising the bid the assignment will remain null, and the claim holds. Therefore, we assume that the normalized rounded bid rises from d_j to $d_j(1+\epsilon)$, the assignment changes from W to W', and the amount of work allocated to machine j changes from w_j to $w'_j > w_j$.

We use $T(W,d)$ to denote the makespan of assignment W on bid vector d. Since the algorithm chooses the optimal assignment among a set that contains both W and W', we have that $T(W,d) \leq T(W',d)$ and $T(W',d') \leq T(W,d')$. Additionally, since the bids in d are smaller than the bids in d', we have that $T(W,d) \leq T(W,d')$ and $T(W',d) \leq T(W',d')$.

Suppose that machine j is the bottleneck in $T(W,d')$, meaning that the load on machine j is the highest. Since $w'_j > w_j$, we have $T(W,d') < T(W',d')$, as the load on machine j increases even more. This is a contradiction to $T(W',d') \leq T(W,d')$, and therefore machine j cannot be the bottleneck in $T(W,d')$. Therefore, if machine j is not the bottleneck in $T(W,d')$, we have that $T(W,d) = T(W,d')$. Since $T(W,d) \leq T(W',d) \leq T(W',d') \leq T(W,d')$, we actually have that $T(W,d) = T(W',d)$ and $T(W,d') = T(W',d')$. Therefore, we have a tie between W and W' for both d and d'. Since in each case the tie is broken differently, it must be that $W = W'$. Since the assignment is sorted (the faster machine is assigned more work), if a machine decreases its speed then the amount of work assigned to it (by the same assignment) cannot increase, which is a contradiction to $w'_j > w_j$.

If machine j is the unique fastest, then due to the normalization of the rounded bids and trimming of high bids, after it raises its bid by one step the new bid vector d' will be as follows: d_j remains 1, bids between $1+\epsilon$ and $(1+\epsilon)^l$ decrease by one step, and bids equal to $(1+\epsilon)^{l+1}$ can either decrease to $(1+\epsilon)^l$ or remain $(1+\epsilon)^{l+1}$.

Let $\hat{d}$ be the same bid vector as d', with all the bids of $(1+\epsilon)^{l+1}$ replaced with $(1+\epsilon)^l$. Since machines that bid $(1+\epsilon)^l$ or more must get a null assignment, then the optimal assignment (among all assignments that are considered by the algorithm) for $\hat{d}$ is the same as d'. The same assignment remains the optimum for vector $\hat{d}(1+\epsilon)$, where all bids are incremented by one step. The bid vector $\hat{d}(1+\epsilon)$ is exactly the bid vector d, with d_j replaced with $1+\epsilon$ (instead of 1). By the same

argument from the case where machine j is not the unique fastest, the work assigned to machine j in $\hat{d}(1+\epsilon)$ is at most the same as the work assigned in d, and therefore the algorithm is monotone for the unique fastest machine as well. ■

To conclude, we claim that the payments for each agent can be calculated in polynomial time, since the number of breakpoints in the integral is bounded by the number of possible allocations considered by the algorithm, which is polynomial in the input size (including $\frac{1}{\epsilon}$) .

6 Conclusions and Open Problems

We have shown a *deterministic* constant-approximation truthful mechanism for assigning jobs on uniformly related machines, and an FPTAS truthful mechanism for the special case where the number of machines is fixed. The main open question left is whether a truthful PTAS mechanism exists in the case of an arbitrary number of machines.

References

[1] N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

[2] P. Ambrosio and E. Auletta. Deterministic monotone algorithms for scheduling on related machines. In *Proceeding of WAOA*, 2004.

[3] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, page 482. IEEE Computer Society, 2001.

[4] Aaron Archer. *Mechanisms for Discrete Optimization with Rational Agents*. PhD thesis, Cornell University, 2004.

[5] Vincenzo Auletta, Roberto De Prisco, Paolo Penna, and Pino Persiano. Deterministic truthful approximation mechanisms for scheduling related machines. In *Proceedings of 21th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2996 of *Lecture Notes in Computer Science*, pages 608–619. Springer Verlag, 2004.

[6] E. Clarke. Multipart pricing of public goods. *Public Choice*, 1971.

[7] Leah Epstein and Jiri Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. In *Proceedings of the 7th Annual European Symposium on Algorithms*, pages 151–162. Springer-Verlag, 1999.

[8] Teofilo Gonzalez, Oscar H. Ibarra, and Sartaj Sahni. Bounds for LPT schedules on uniform processors. *SIAM Journal on Computing*, 6(1):155–166, March 1977.

[9] T. Groves. Incentives in teams. *Econemetrica*, 1973.

[10] D. Hochbaum and D. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

[11] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling non-identical processors. *Journal of the Association for Computing Machinery*, 23:317–327, 1976.

[12] N. Nisan and A. Ronen. Algorithmic mechanism design. In *31st ACM Symp. on Theory of Computing*, pages 129–140, 1999.

[13] Noam Nisan and Amir Ronen. Computationally feasible vcg mechanisms. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 242–252. ACM Press, 2000.

[14] Christos Papadimitriou. Algorithms, games, and the internet. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 749–753. ACM Press, 2001.

[15] Eva Tardos. Network games. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2004.

[16] W. Vickery. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 1961.