Mechanism design for fractional scheduling on unrelated machines

George Christodoulou[1], Elias Koutsoupias[2], and Annamária Kovács[1]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany
{gchristo,panni}@mpi-inf.mpg.de
[2] Department of Informatics, University of Athens
elias@di.uoa.gr

Abstract. In this paper, we consider the mechanism design version of the fractional variant of the scheduling problem on unrelated machines. We give a lower bound of $2-1/n$ for any fractional truthful mechanism, while we propose a truthful mechanism that achieves approximation of $1+(n-1)/2$, for n machines. We also focus on an interesting family of allocation algorithms, the *task-independent* algorithms. We give a lower bound of $1+(n-1)/2$, that holds for every (not only monotone) allocation algorithm of this class. Under this consideration, our truthful independent mechanism is the best that we can hope from this family of algorithms.

1 Introduction

Mechanism design is an important branch of Microeconomics and in particular of Game Theory. The objective of a mechanism designer is to implement a goal, e.g., to sell an object to a set of potential buyers. The problem derives from the fact that the designer may not be informed about some parameters of the input. These values are controlled by selfish agents that may have incentive to misinform the designer, if this can serve their atomic interests. The mechanism design approach concerns the construction of a game, so that the outcome (equilibrium) of the game is the goal of the designer.

Task scheduling is one of the most important and well-studied problems in Computer Science, as it often arises, in numerous forms, as a subproblem in almost every subfield of Computer Science. One of its most classical and general variants is the scheduling on *unrelated* machines. In this setting, there are n machines[3] and m tasks, and the processing time needed by machine i to perform task j is determined by the t_{ij} entry of an $n \times m$ matrix t. A common objective is to assign the tasks to the machines in such a way, that the maximum load of the machines (i.e., the makespan) is minimized.

[3] In Game-theoretic settings n is used to denote the number of the players, while in scheduling literature, usually m is used to denote the cardinality of the machines set. In our case, the aforementioned sets coincide. We prefer to use the former notation, in order to be compatible with the original paper [20] by Nisan and Ronen.

Nisan and Ronen [20] initiated the study of the mechanism design version of scheduling on unrelated machines. In this form of the problem, the processing times that a machine i needs in order to execute the tasks (vector t_i), are *private* values, known only to the corresponding machine. The machines are controlled by selfish agents that aim at satisfying their own interests, and in the particular case they are unwilling to perform any task. In order to motivate them to reveal their actual values, the classical approach adopted by mechanism design, is to introduce side payments, i.e., to hire the machines. A mechanism for this problem consists of an allocation algorithm and a payment scheme. We are interested in bounding the approximation ratio of the mechanism's allocation algorithm.

In the classical version of the problem, each task must be assigned to exactly one machine. The LP-relaxation of the problem, also known as fractional scheduling, concerns the version where instead of being assigned to a single machine, each task can be splitted among the machines. Fractional variations of combinatorial problems have been studied extensively in network optimization, e.g., routing splittable traffic or flow problems.

The fractional scheduling problem can be formulated as a linear program and hence it can be solved in polynomial time. LP-relaxation turns out to be a useful tool in the design of approximation algorithms (both deterministic and randomized)[4]. Furthermore, it turned out to be a powerful technique to provide randomized truthful mechanisms (*see e.g. [16, 3]*). It is natural to ask how powerful LP-relaxation is in the mechanism design framework.

In this paper we consider the mechanism design version of the fractional scheduling on unrelated machines. An interesting fact is that while the offline problem is polynomially solvable, it turns out that in the mechanism design version of the problem it cannot be approximated within a constant factor, even by non-polynomial mechanisms (see Sec. 3). This means, that the additional properties that the allocation of a mechanism needs to satisfy in contrast to a simple algorithm (cf. Sec. 2), do not allow us to achieve an exact solution, even in non-polynomial time. Lower bounding fractional mechanisms is a nice approach to lower bound randomized (and deterministic) mechanisms of the integral case, as splitting a job is clearly a more radical solution than randomly assigning it.

We are particularly interested in a family of mechanisms that we call *task-independent*. A task-independent algorithm is any algorithm that in order to allocate task j, only considers the processing times t_{ij}, that concern the particular task. Such a consideration is motivated by the fact that (to the best of our knowledge) all the known positive results for this problem (e.g., see the mechanisms in [18, 20]), and in addition the mechanism that we propose in this paper, belong to this family of mechanisms. The question that we address here is: *how far can we go with task-independent algorithms?*

1.1 Related Work

Scheduling on unrelated machines is a classical NP-hard problem. Lenstra et al. [17] gave a 2-approximation polynomial time algorithm, while they also proved

[4] In fact, it has been used in order to obtain the 2-approximation algorithm in [17].

that the problem cannot be approximated (in polynomial time) within a factor less than 3/2. The mechanism design version of the problem originates in the seminal work of Nisan and Ronen [20]. They gave a n-approximation truthful mechanism and a lower bound of 2, while they conjectured the actual bound to be n. Christodoulou et al. [9] improved the lower bound to $1+\sqrt{2}$. Narrowing the gap between the lower and the upper bound still remains a big open question.

Randomization usually reduces the approximation ratio and that is also the case for this problem. Nisan and Ronen [20] proposed a randomized mechanism for 2 machines with approximation ratio 7/4. Recently, Mu'alem and Schapira [18] generalized this mechanism for n machines and achieved a $7n/8$ randomized truthful mechanism. In the same work, they also gave a lower bound of $2-1/n$ for randomized mechanisms. Notice that all the known lower bounds for this problem (both deterministic and randomized) follow due to the infrastructure of truthful mechanisms, and do not reside in any computational assumption; consequently they hold even for non polynomial time mechanisms.

Scheduling on related machines, from the mechanism design point of view, was first studied by Archer and Tardos [4]. In this variant of the problem, the private parameter for each machine, is a single value (its speed). Archer and Tardos [4] characterized the class of truthful mechanisms for this setting, in terms of a monotonicity condition of the mechanism's allocation algorithm. A similar characterization for one-parameter mechanism design problems (single item auction) can also be found in Myerson [19]. For this problem, it turns out that the optimal allocation algorithm can be modified to be a truthful mechanism. Archer and Tardos [4] gave a randomized truthful 3-approximation algorithm, which was later improved to a 2-approximation by Archer [2]. Andelman et al. [1] gave the first deterministic polynomial mechanism for the problem, with an approximation ratio of 5. Kovács [13] improved this by giving a 3-approximation deterministic truthful mechanism, while finally the ratio was reduced to 2.8 [14].

In the field of Combinatorial Auctions, a wide variety of combinatorial optimization problems has been considered from the mechanism design point of view (see for example [3, 6, 8, 10, 5, 11] and references within). In this context, Saks and Yu [21] characterized the class of truthful mechanisms for combinatorial auctions with convex valuations, generalizing results of [7, 12, 15].

1.2 Our Results

In this paper, we consider the mechanism design version of fractional scheduling on unrelated machines. We give a $2-1/n$ lower bound on the approximation ratio, that can be achieved by any truthful mechanism. This result shows that even in the case of such a problem, for which the offline version can be exactly solved in polynomial time, its mechanism design analog may turn out to be impossible to approximate, even by non-polynomial mechanisms. Notice that giving a lower bound for fractional mechanisms is another way to obtain lower bounds for randomized mechanisms for the integral case. Consequently, our $2-1/n$ lower bound extends the lower bounds in [18] to the class of fractional mechanisms. Note that a fractional mechanism is more powerful than a randomized mechanism for the integral case, since it has the flexibility to split a task into many

machines, while a randomized mechanism, finally, has to assign the whole task to a machine, and this affects its approximation ratio.

In the positive direction, we give a truthful mechanism with approximation ratio 3/2 for 2 machines, which matches our lower bound. This is the first new tight bound that we have for any variant of the problem, after the tight bound of 2 in the integral case, obtained for 2 machines in [20]. The generalization of our mechanism for n machines gives us an approximation ratio of $1 + (n-1)/2$.

Next we turn our attention to a family of mechanisms that we call *task-independent*. This family consists of mechanisms, where the decision for the assignment of a task, depends only on the processing times that concern the particular task *(time column w.r.t. the task)*. Considering task-independence is motivated by the fact that all known 'reasonable' deterministic and randomized mechanisms for this problem are task-independent. Furthermore, this sort of independence has attractive properties: easy to design by applying methods for one-parameter auctions, fits well with on-line settings, where tasks may appear one-by-one. It is natural to ask if there is room for improvement on the approximation ratio by use of such mechanisms. We extend this question for the class of task-independent *algorithms* that need not satisfy the additional properties imposed by truthfulness. We give a lower bound of $1 + (n-1)/2$ on the approximation ratio of any algorithm that belongs to this class. Our mechanism is also task-independent, and hence is optimal over this family of algorithms.

2 Problem Definition

In this section, we fix the notation that we will use throughout this paper, furthermore we give some preliminary definitions and cite relevant results.

There are n machines and m tasks. Each machine $i \in [n]$ needs t_{ij} units of time to perform task $j \in [m]$. We denote by t_i the row vector corresponding to machine i, and by t^j the column vector of the running times of task j. We assume that each machine $i \in [n]$ is controlled by a selfish agent that is unwilling to perform any operation, and vector t_i is private information known only to her. The vector t_i is also called the *type* of agent i.

Any mechanism defines for each player i a set A_i of available strategies, the player (agent) can choose from. We will consider *direct revelation* mechanisms, i.e., $A_i = T_i$ for all i, meaning that the players strategies are to simply report their types to the mechanism. In general, T_i consists of all possible vectors $b_i \in \mathbb{R}^m_+$, that is, a player may report a false vector $b_i \neq t_i$, if this serves his interests.

A mechanism $M = (x, p)$ consists of two parts:

An allocation algorithm: The allocation algorithm x, depends on the players' bids $b = (b_1, \ldots, b_n)$, with $0 \leq x_{ij} \leq 1$ denoting the fraction of task j that is assigned to the machine i. In the unsplittable case, these variables take only integral values $x_{ij} = \{0, 1\}$. Every task must be completely assigned to the machines' set, so $\sum_{i \in [n]} x_{ij} = 1, \quad \forall j \in [m]$.

A payment scheme: The payment scheme $p = (p_1, \ldots, p_n)$, also depends on the bid values b. The functions $p_1, \ldots, p_n$ stand for the payments that the mechanism hands to each agent.

The *utility* u_i of a player i is the payment that he gets minus the *actual* time that he needs in order to execute the set of tasks assigned to her, $u_i(b) = p_i(b) - \sum_{j\in[m]} t_{ij}x_{ij}(b)$. We are interested in *truthful* mechanisms. A mechanism is truthful, if for every player, reporting his true type is a *dominant strategy*. Formally,

$$u_i(t_i, b_{-i}) \geq u_i(t'_i, b_{-i}), \qquad \forall i \in [n], t'_i \in T_i, b_{-i} \in T_{-i}\,,$$

where T_{-i} denotes the possible types of all players disregarding i.

We remark here, that once we adopt the solution concept of dominant strategies, focusing on direct revelation and in particular on truthful mechanisms is not at all restrictive, due to the *Revelation Principle*. Roughly, the Revelation Principle states that any problem that can be implemented by a mechanism with dominant strategies, can also be implemented by a truthful mechanism (cf. [20]).

The objective function that we consider, in order to evaluate the performance of a mechanism's allocation algorithm x, is the maximum load of a machine (makespan), with respect to the real time matrix t. When we refer to the makespan of a mechanism, we mean the makespan of its allocation algorithm with respect to the input t, and we denote it by $Mech(t) = \max_{i\in[n]} \sum_{j\in[m]} t_{ij}x_{ij}$. Since we aim at minimizing the makespan, the optimum is $Opt(t) = \min_x \max_{i\in[n]} \sum_{j\in[m]} t_{ij}x_{ij}$. We are interested in the approximation ratio of the mechanism's allocation algorithm. A mechanism M is c-approximate, if $c \geq Mech(t)/Opt(t) \quad \forall t \in T$.

Although our mechanism is polynomially computable, we do not aim at minimizing the running time of the algorithm; we are looking for mechanisms with low approximation ratio. Our lower bounds also don't make use of any computational assumptions.

A useful characterization of truthful mechanisms in terms of the following monotonicity condition, helps us to get rid of the payments and focus on the properties of the allocation algorithm.

Definition 1. *An allocation algorithm is called* monotone[5] *if it satisfies the following property: for every two sets of tasks t and t' which differ only on machine i (i.e., on the i-th row) the associated allocations x and x' satisfy $(x_i - x'_i) \cdot (t_i - t'_i) \leq 0$, where $\cdot$ denotes the dot product of the vectors, that is, $\sum_{j\in[m]}(x_{ij} - x'_{ij})(t_{ij} - t'_{ij}) \leq 0$.*

The following theorem states that every truthful mechanism has to satisfy the monotonicity condition. It was used by Nisan and Ronen [20] in order to obtain their lower bounds.

Theorem 1. *Every truthful mechanism is monotone.*

Saks and Yu [21] proved that monotonicity is also a sufficient condition, for the combinatorial auctions setting with convex valuations (*i.e. there exist payments that can make a monotone algorithm into a truthful mechanism).*

[5] Also known as *weakly monotone*.

For the one-parameter case, i.e., when every agent has a single value to declare (e.g., the speed of her machine), Myerson [19] *(for auction setting)* and Archer and Tardos [4] *(for scheduling setting)*, showed that the monotonicity of the (allocation) algorithm is a necessary and sufficient condition for the existence of a truthful payment scheme. In this case they also provide an explicit formula for the payments. In their theorem cited below, the notion of a *decreasing output function*, corresponds to a monotone algorithm in the one-parameter setting.

Theorem 2. [19, 4] *The output function admits a truthful payment scheme if and only if it is decreasing. In this case the mechanism is truthful if and only if the payments $p_i(b_i, b_{-i})$ are of the form*

$$h_i(b_{-i}) + b_i x_i(b_i, b_{-i}) - \int_0^{b_i} x_i(u, b_{-i})\, du$$

where the h_i are arbitrary functions.

3 Lower Bound for Truthful Mechanisms

Here we will give a lower bound on the approximation ratio of any fractional truthful mechanism.

Theorem 3. *There is no deterministic truthful mechanism that can achieve an approximation ratio better than $2 - \frac{1}{n}$, where n is the number of the machines.*

Proof. Let t be the actual time matrix of the players as below

$$t_{ij} = \begin{cases} 0, & j = i \\ 1, & j = n+1 \\ A, & \text{otherwise} \end{cases}$$

and $x = x(t)$ be the corresponding allocation that a truthful mechanism $M = (x, p)$ gives with respect to t. For significantly large values of A, player i gets substantially the whole portion of task i, otherwise the approximation ratio is high, e.g., for $A = \frac{2}{\delta}$, every player i should get a portion greater than $1-(n-1)\delta$, otherwise the approximation ratio is at least 2.

Clearly, there is a player $k \in [n]$, with $x_{kn+1} \geq \frac{1}{n}$. Now let's consider how the allocation algorithm of the mechanism behaves if the following time matrix is given as input

$$t'_{ij} = \begin{cases} \frac{1}{n-1}, & i = k, j = i \\ 1 - \epsilon, & i = k, j = n+1 \\ t_{ij}, & \text{otherwise} \end{cases}$$

The following claim states that due to monotonicity, the mechanism cannot assign to player k a substantially smaller portion of the $n+1^{\text{st}}$ task than $\frac{1}{n}$.

Claim. If $x_{kn+1} \geq \frac{1}{n}$, then for the allocation $x' = x(t')$ on input t' it holds that $x'_{kn+1} \geq \frac{1}{n} - \epsilon$.

Proof. Due to Theorem 1 we have that for every player $i \in [n]$, it holds that

$$\sum_{j\in[m]} (t_{ij} - t'_{ij})(x_{ij} - x'_{ij}) \leq 0$$

and by applying this to the k-th player we get

$$\left(0 - \frac{1}{n-1}\right)(x_{kk} - x'_{kk}) + (1 - 1 + \epsilon)(x_{kn+1} - x'_{kn+1}) \leq 0,$$

from which we get

$$x'_{kn+1} \geq x_{kn+1} + \frac{x'_{kk} - x_{kk}}{\epsilon(n-1)} \geq x_{kn+1} - \frac{\delta}{\epsilon} \geq \frac{1}{n} - \frac{\delta}{\epsilon}$$

and for $\delta = \epsilon^2$ we finally obtain

$$x'_{kn+1} \geq \frac{1}{n} - \epsilon$$

□

On the other hand, an optimal allocation x^* for t' is

$$x^*_{ij} = \begin{cases} 1, & j = i \\ 0, & i = k, j = n+1 \\ \frac{1}{n-1}, & i \neq k, j = n+1 \\ 0, & \text{otherwise} \end{cases}$$

with makespan $1/(n-1)$, while the mechanism gives player k a total load of at least

$$(1 - (n-1)\delta)\frac{1}{n-1} + \left(\frac{1}{n} - \epsilon\right)(1 - \epsilon) > \frac{1}{n-1} + \frac{1}{n} - \delta - \epsilon\left(\frac{n+1}{n}\right).$$

For arbitrary small ϵ, this finally gives an approximation ratio of at least $2 - \frac{1}{n}$.
□

4 The Truthful Mechanism

We describe a truthful mechanism, called SQUARE, for the fractional scheduling problem, with approximation ratio $1 + \frac{n-1}{2}$. On two machines this ratio becomes $3/2$, so in this case SQUARE has the best possible worst case ratio w.r.t. truthful mechanisms. Furthermore, in Section 5 we will show that for arbitrary number of machines, our mechanism is optimal among the so called *task-independent* algorithms.

Next, we define the mechanism SQUARE$= (x^{Sq}, p^{Sq})$[6]. Recall that b_{ij} is the reported value for t_{ij}, the actual execution time of task j on machine i.

[6] In most of the section we will omit the superscripts ^{Sq}.

Definition 2 (The mechanism SQUARE= (x^{Sq}, p^{Sq})).

Allocation algorithm: *Let $b^j = (b_{1j}, b_{2j}, \ldots, b_{nj})^T$ be the jth column-vector of the input matrix. If b^j has at least one zero coordinate, then* SQUARE *distributes the jth task among machines having zero execution time arbitrarily. If $b_{ij} \neq 0$ ($i \in [n]$), then the fraction of the jth task allocated to machine i is*

$$x_{ij}^{Sq}(b) = x_{ij}(b) = \frac{\prod_{k \neq i} b_{kj}^2}{\sum_{l=1}^{n} \prod_{k \neq l} b_{kj}^2}\,. \tag{1}$$

Payment scheme: *Let the constants c_{ij} be defined as*

$$c_{ij} = \frac{\prod_{k \neq i} b_{kj}}{\sqrt{\sum_{l \neq i} \prod_{k \neq l,i} b_{kj}^2}}\,,$$

then the payments $p^{Sq} = (p_1, \ldots, p_n)$ to the agents are

$$p_i(b) = \sum_{j=1}^{m} \left(b_{ij} \cdot \frac{c_{ij}^2}{b_{ij}^2 + c_{ij}^2} + c_{ij} \cdot \frac{\pi}{2} - c_{ij} \arctan \frac{b_{ij}}{c_{ij}} \right)\,.$$

The algorithm x^{SQ} of SQUARE allocates the tasks individually (independently), and so that the assigned fractions of a task are inversely proportional to the squares of (declared) execution times w.r.t. each machine. For instance, for two machines (1) boils down to

$$x_{1j} = \frac{b_{2j}^2}{b_{1j}^2 + b_{2j}^2}; \qquad\qquad x_{2j} = \frac{b_{1j}^2}{b_{1j}^2 + b_{2j}^2}\,.$$

For arbitrary n it is obvious that $0 \leq x_{ij} \leq 1$, and $\sum_{i=1}^{n} x_{ij} = 1$. It is easy to see that SQUARE is monotone: Let the input matrix b be changed only on the ith row, that is, for any fixed task j, just the entry b_{ij} may change. Assume first that in the column-vector b^j all execution times are nonzero. Observe that the variable b_{ij} appears only in the denominator of the expression (1), namely as b_{ij}^2, having a positive coefficient. Thus, x_{ij} does not increase when b_{ij} increases, and vice versa. It is easy to see that the same holds if in b^j there are zero entries other than b_{ij}, and similarly, if b_{ij} was, or just became the only zero entry. Thus, we obtained that for every single one-parameter problem b^j, the assignment is monotone, and this, in turn, implies weak monotonicity *(see Definition 1)* for x^{Sq}.

Now consider p^{Sq}. For two machines, the constant c_{ij} is simply the bid of the other machine for this job, that is, $c_{1j} = b_{2j}$ and $c_{2j} = b_{1j}$. For more machines, c_{ij} would be the 'bid' of a single other machine, if we replaced the machines $[n]\backslash\{i\}$ with one machine. The payment $p_i(b)$ is simply defined to be the sum of the payments that agent i would get for performing each (fractional) task independently, as determined for truthful mechanisms for one-parameter agents by Theorem 2:

$$p_i(b_i, b_{-i}) = h_i(b_{-i}) + b_i x_i(b_i, b_{-i}) - \int_0^{b_i} x_i(u, b_{-i})\, du\,.$$

Here the $h_i(b_{-i})$ are arbitrary constants. If we want that the so called *voluntary participation* [4] of the players is ensured (i.e., it is worth taking part in the game), then h_i can be chosen to be $h_i = \int_0^\infty x_i(u, b_{-i})\, du$, so that finally we get

$$p_i(b_i, b_{-i}) = b_i x_i(b_i, b_{-i}) + \int_{b_i}^{\infty} x_i(u, b_{-i})\, du\,,$$

for the one-parameter case. Applying this formula for each task, we obtain the above definition of the payments.

Theorem 4. *The mechanism* SQUARE *is truthful.*

Proof. To put it short, the theorem follows from the fact that SQUARE is the sum of m truthful mechanisms for the one-parameter problem (note that the total execution time (load), the total payment, thus also the total utility is the sum of the respective amounts for the single tasks). For each j, the mechanism (x^j, p^j) is truthful, since x^j is a monotone algorithm, and we defined $p^j = (p_{1j}, \ldots, p_{nj})^T$ according to Theorem 2. We do not include an elementary proof here. □

Approximation Ratio. Let $Squ(t)$ be the makespan of the schedule produced by SQUARE on input t, and $Opt(t)$ denote the optimum makespan. In what follows, we show that $Squ(t)/Opt(t) \leq 1 + \frac{n-1}{2}$ for any matrix t. The next lemma will largely simplify the upper-bound proof. The proof of the lemma is omitted.

Lemma 1. *If there exists an input instance t, such that $Squ(t)/Opt(t) = \alpha$, then there also exists an instance t^*, for which $Squ(t^*)/Opt(t^*) = \alpha$, moreover there is an optimal allocation of t^* that does not split any job.*

Theorem 5. *For the approximation ratio of* SQUARE, $\frac{Squ(t)}{Opt(t)} \leq 1 + \frac{n-1}{2}$ *holds, where n denotes the number of machines, and t is an arbitrary set of input tasks.*

Proof. Consider the input t. Due to Lemma 1, we can assume that the (indices of) tasks are partitioned into the sets $J_1,\ J_2,\ \ldots, J_n$, so that there is an optimal allocation OPT where job t^j is allocated completely to machine i, if and only if $j \in J_i$. We can also assume that $t_{ij} > 0$ for all i and j. Otherwise we would have a job that adds zero execution time to the makespan in both the allocation of SQUARE, and of OPT, and removing this job from the input would not affect the approximation ratio. For the optimum makespan it holds that

$$Opt(t) = \max_{i \in [n]} \sum_{j \in J_i} t_{ij}\,. \tag{2}$$

For the running time of an arbitrary machine i in SQUARE, we have

$$Squ_i(t) = \sum_{r=1}^{n} \sum_{j \in J_r} x_{ij}(t) t_{ij}\,,$$

where the $x_{ij}(t)$ are defined by (1). We decompose the above expression as follows:

$$Squ_i(t) = \sum_{j \in J_i} x_{ij} t_{ij} + \sum_{r \neq i} \sum_{j \in J_r} x_{ij} t_{ij} \,.$$

We can upper bound the first sum using (2), and the fact that $x_{ij} \leq 1$:

$$\sum_{j \in J_i} x_{ij} t_{ij} \leq \sum_{j \in J_i} 1 \cdot t_{ij} \leq Opt(t) \,.$$

Next we upper bound every sum of the form $\sum_{j \in J_r} x_{ij} t_{ij} \quad (r \neq i)$, by $\frac{1}{2} \cdot Opt(t)$. Since there are $n-1$ such sums, this will prove that

$$Squ_i(t) \leq Opt(t) + (n-1) \cdot \frac{1}{2} \cdot Opt(t) = (1 + \frac{n-1}{2}) \cdot Opt(t) \,.$$

Since i was an arbitrary machine, eventually this implies

$$Squ(t) = \max_{i \in [n]} Squ_i(t) \leq (1 + \frac{n-1}{2}) \cdot Opt(t) \,.$$

The bound $\sum_{j \in J_r} x_{ij} t_{ij} \leq \frac{1}{2} \cdot Opt(t)$ can be shown as follows:

$$\sum_{j \in J_r} x_{ij} t_{ij} = \sum_{j \in J_r} \frac{\prod_{k \neq i} t_{kj}^2}{\sum_{l=1}^{n} \prod_{k \neq l} t_{kj}^2} \cdot t_{ij} = \sum_{j \in J_r} \frac{t_{ij} t_{rj} \prod_{k \neq i,r} t_{kj}^2}{\sum_{l=1}^{n} \prod_{k \neq l} t_{kj}^2} \cdot t_{rj} =$$

$$= \sum_{j \in J_r} \frac{t_{ij} t_{rj}}{t_{ij}^2 + t_{rj}^2 + \sum_{l \neq i,r} t_{ij}^2 t_{rj}^2 / t_{lj}^2} t_{rj} \leq \sum_{j \in J_r} \frac{t_{ij} t_{rj}}{t_{ij}^2 + t_{rj}^2} t_{rj} \leq \sum_{j \in J_r} \frac{1}{2} t_{rj} \leq \frac{1}{2} Opt(t),$$

where the inequality $\frac{t_{ij} t_{rj}}{t_{ij}^2 + t_{rj}^2} \leq \frac{1}{2}$ holds for any two nonzero real numbers; the last inequality is implied by (2). □

Corollary 1. *For two machines the truthful mechanism* Square *has approximation ratio* 3/2*, which is the best worst case ratio we can expect from any truthful mechanism for the fractional scheduling problem.*

5 Lower Bound for Independent Algorithms

In this section we prove a lower bound of $1 + \frac{n-1}{2}$ for the worst case ratio of independent fractional algorithms. An algorithm is independent, if it allocates the tasks independently of each-other, or formally:

Definition 3. *An allocation algorithm x is called* task-independent*, or simply* independent*, if the following holds: If t and t' are two $n \times m$ input matrices, such that for the jth task $t_{ij} = t'_{ij}$ $(\forall i \in [n])$, then for this task it also holds that $x_{ij} = x'_{ij}$ $(\forall i \in [n])$.*

It is remarkable, that the currently known best mechanisms (in fact, any 'reasonable' mechanism we know of) are all independent, in the integral, the randomized, and the fractional case. It is not difficult to come up with independent (suboptimal) algorithms, which are also weakly monotone. However it seems to be an intriguing question, whether there exist non-inependent, and still monotone algorithms having better approximation ratio than the best independent ones. We note that in the integral case it is easy to construct an instance with n machines and n^2 tasks, that proves a lower bound of n (i.e., tight bound) for independent algorithms.

Theorem 6. *If x is an independent fractional allocation algorithm for the unrelated machines problem, then it has approximation ratio of at least $1 + \frac{n-1}{2}$, where n denotes the number of machines.*

Proof. In order to obtain the lower bound, consider the following input matrix with $n \geq 2$ machines and $m = 1 + \binom{n}{2}$ tasks. The first task is numbered by $j = 0$; furthermore, for all $\binom{n}{2}$ possible pairs of machines $g < h$ there is a task j_{gh} :

$$t_{ij} = \begin{cases} 0, & j = 0 \\ 1, & j = j_{gh},\ i \in \{g, h\} \\ A, & \text{otherwise} \end{cases}$$

Obviously, by setting A large enough, we can make it sure – like in the proof of Theorem 3 – that the corresponding share of a player of a certain task is arbitrarily small, otherwise the approximation ratio gets too large. That is, we can assume that the bulk of any job is allocated to the machines having execution time 1 for this job.

Let us consider an arbitrary independent algorithm x. Observe that no matter how x allocates the above tasks, the total running time of all the jobs cannot be less than $\binom{n}{2}$. Thus, there exists a machine, say machine k, with running time at least $\binom{n}{2}/n = \frac{n-1}{2}$. Now we modify the instance t to t' : we keep the original execution times of tasks that had running time 1 on machine k, and zero out all other t_{ij}; furthermore, task 0 will now have execution time 1 on machine k, and A on other machines.

$$t'_{ij} = \begin{cases} 1, & j = 0,\ i = k \\ A, & j = 0,\ i \neq k, \\ t_{ij}, & j = j_{gh} \text{ and } g = k \text{ or } h = k \\ 0, & \text{otherwise} \end{cases}$$

As noted above, on instance t at least $\frac{n-1}{2} - \epsilon$ running time on machine k was due to jobs that have execution time 1 on this machine. Since x is independent, on instance t' the machine gets the same allocation over these jobs, and also gets a $(1-\epsilon)$ fraction of job 0, achieving a running time of at least $1 + (n-1)/2 - 2\epsilon$, for any $\epsilon > 0$. On the other hand, it is clear that the optimal allocation has makespan 1. □

Corollary 2. *The mechanism* SQUARE *has optimal approximation ratio among all independent mechanisms.*

One can show that among all allocations where the distribution of task j is inversely proportional to $(t_{1j}^{\alpha}, t_{2j}^{\alpha}, \ldots, t_{nj}^{\alpha})$ for some $\alpha > 0$, the above optimal approximation ratio is obtained if and only if $\alpha = 2$.

References

1. N. Andelman, Y. Azar, and M. Sorani. Truthful approximation mechanisms for scheduling selfish related machines. In *STACS*, pages 69–82, 2005.
2. A. Archer. *Mechanisms for Discrete Optimization with Rational Agents.* PhD thesis, Cornell University, January 2004.
3. A. Archer, C. H. Papadimitriou, K. Talwar, and É. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *SODA*, pages 205–214, 2003.
4. A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *FOCS*, pages 482–491, 2001.
5. M. Babaioff, R. Lavi, and E. Pavlov. Mechanism design for single-value domains. In *AAAI*, pages 241–247, 2005.
6. Y. Bartal, R. Gonen, and N. Nisan. Incentive compatible multi unit combinatorial auctions. In *TARK*, pages 72–87, 2003.
7. S. Bikhchandani, S. Chatterji, R. Lavi, A. Mu'alem, N. Nisan, and A. Sen. Weak monotonicity characterizes deterministic dominant strategy implementation. *Econometrica*, 74(4):1109–1132, 2006.
8. P. Briest, P. Krysta, and B. Vöcking. Approximation techniques for utilitarian mechanism design. In *STOC*, pages 39–48, 2005.
9. G. Christodoulou, E. Koutsoupias, and A. Vidali. A lower bound for scheduling mechanisms. In *SODA*, pages 1163–1170, 2007.
10. S. Dobzinski, N. Nisan, and M. Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *STOC*, pages 610–618, 2005.
11. S. Dobzinski, N. Nisan, and M. Schapira. Truthful randomized mechanisms for combinatorial auctions. In STOC, pages 644–652, 2006.
12. H. Gui, R. Müller, and R. V. Vohra. Dominant strategy mechanisms with multidimensional types. In *Computing and Markets*, 2005.
13. A. Kovács. Fast monotone 3-approximation algorithm for scheduling related machines. In *ESA*, pages 616–627, 2005.
14. A. Kovács. *Fast Algorithms for Two Scheduling Problems.* PhD thesis, Universität des Saarlandes, 2007.
15. R. Lavi, A. Mu'alem, and N. Nisan. Towards a characterization of truthful combinatorial auctions. In *FOCS*, pages 574–583, 2003.
16. R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. In *FOCS*, pages 595–604, 2005.
17. J.K. Lenstra, D.B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(1):259–271, 1990.
18. A. Mu'alem and M. Schapira. Setting lower bounds on truthfulness. In *SODA*, pages 1143–1152, 2007.
19. R. B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.
20. N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.
21. M. E. Saks and L. Yu. Weak monotonicity suffices for truthfulness on convex domains. In *EC*, pages 286–293, 2005.