Routing Selfish Unsplittable Traffic*

Vincenzo Auletta Roberto De Prisco Paolo Penna Giuseppe Persiano

August 23, 2006

Abstract

We consider general resource assignment games involving *selfish users/agents* in which users compete for resources and try to be assigned to resources which maximize their own benefits (e.g., try to route their traffic through links which minimize the latency of their own traffic). We propose and study a *mechanism design* approach in which an allocation mechanism assigns users to resources and charges the users for using the resources so to induce each user to *truthfully* report a private piece of information he/she holds (e.g., how much traffic he/she needs to transmit). This information is crucial for computing optimal (or close to the optimal) allocations and an agent could misreport his/her information so to induce the underlying allocation algorithm to output a solution which he/she likes more (e.g., which assigns better resources to him/her).

For our resource allocation problems, we give an *algorithmic characterization* of the solutions for which truth-telling is a Nash equilibrium. A natural application of these results is to a scheduling/routing problem which is the mechanism design "counterpart" of the selfish routing game of Koutsoupias and Papadimitriou [1999]: Each selfish user wants to route a piece of unsplittable traffic using one of m links of different speeds so to minimize his/her *own* latency. Our mechanism design counterpart can be seen as the problem of scheduling *selfish jobs* on parallel related machines and is the dual of the problem of scheduling (unselfish) jobs on parallel *selfish machines* studied by Archer and Tardos [2001].

Koutsoupias and Papadimitriou studied an "anarchic" scenario in which each user chooses his/her own link and this may produce Nash equilibria of cost $\Omega(\log m/\log\log m)$ times the optimum. Our mechanism design counterpart is a possible way of reducing the effect of selfish behaviour via suitable incentives to the agents (namely, taxes for using the links). We indeed show that, in the resulting game, it is possible to guarantee an approximation factor of 8 for any number of links/machines (this solution also works for online settings). However, it remains impossible to guarantee arbitrarily good approximate solutions even for two links/machines and even if the allocation algorithm is allowed super polynomial time. This result shows that our scheduling problem with selfish jobs is more difficult than the scheduling problem with selfish machines by Archer and Tardos (which admits exact solutions).

We study several generalizations of this basic problem, including (i) routing over arbitrary networks, (ii) cost functions other than the maximum link congestion, (iii) the case in which both machines and jobs are owned by selfish agents, (iv) agents owning more than one job, and (v) machines with different internal scheduling policies. These variants account for basic aspects of resource allocation problems and the corresponding results show how these aspects affect the quality of the solution that one can compute when selfish agents are involved.

Keywords: Scheduling, Selfish Routing, Nash Equilibrium, Algorithmic Mechanism Design.

*A preliminary version of this work appeared in the Proceedings of SPAA 2004. Work supported by the European Project IST-15964, Algorithmic Principles for Building Efficient Overlay Computers (AEOLUS).

Contents

1 Introduction **3**
1.1 Our contribution 4
1.2 Related work 7

2 Resource assignment games **8**
2.1 A mechanism design approach 10

3 A characterization of NE-truthful mechanisms **11**

4 Mechanisms for the MKP game **13**
4.1 Upper bounds for the MKP game 14
4.2 Lower bounds 15

5 Extensions of the MKP game **18**
5.1 Selfish machines 18
5.2 More games with quasi one-parameter agents 20

6 Agents owning more than one job **20**
6.1 Lower bounds 21
6.2 Upper bounds 22

7 Conclusions and open problems **29**

1 Introduction

Selfish routing games have been the subject of several studies because of their applications to situations, typical in the Internet, where different entities compete for shared resources and may act *selfishly* trying to increase their own benefits. The simplest example of such a game has been studied in the seminal paper by Koutsoupias and Papadimitriou [19]. Here, we have m parallel communication links and a set of n selfish agents, with agent i owning a piece of unsplittable traffic of weight t_i. Links can have different speeds and each agent chooses the link to use for routing his/her traffic so to minimize his/her expected latency even though this may lead to a globally inefficient solution. The Koutsoupias-Papadimitriou (KP) model does not assume central coordination (in the sense of a manager that centrally decides routes for the agents) nor agents are assumed to coordinate their routing strategies. A natural approach for analyzing this scenario comes from Microeconomics and Game Theory and it uses the well-known concept of *Nash equilibrium* (see e.g. [24]). Roughly speaking, in a Nash equilibrium an agent cannot benefit by unilaterally changing his/her "strategy" (in the KP model, to pick a different link). Koutsoupias and Papadimitriou [19] propose to study the *coordination ratio* as the measure of the loss of performance due to the lack of cooperation among the selfish agents: given a global optimization function (like minimizing the maximum link congestion), how bad is the *worst-case* Nash equilibria? In other words, the coordination ratio measures the "price of the anarchy." For this simple game it is possible to have a Nash equilibrium that costs $\Theta(\log m/\log\log\log m)$ times the optimum (see [8]). This result suggests to consider alternatives to the "anarchic policy" so to induce the agents to perform strategies which result in a better (possibly optimal) system performance. In this work, we propose a *mechanism design* approach in order to obtain a better resource allocation. We first consider the very same network of the KP model in which a *scheduler* assigns traffic to the links. The main difficulty is the fact that the scheduler must compute an allocation based on the amount of traffic that each agent *declares* to be willing to route; this value may be different from the true one and an agent could misreport this piece of information to "manipulate" the scheduler and reduce his/her latency. We thus augment the algorithm with suitable payment functions which charge users for using the links and whose purpose is to make convenient for the agents to be truthful (see Sect. 2 for formal definitions). The resulting problem is a mechanism design problem in which we want to schedule *selfish jobs* (i.e., traffic) on parallel related machines (i.e., links), as opposed to to the problem of scheduling (non-selfish) jobs on selfish machines investigated by Archer and Tardos [1]. In both cases, the objective is to minimize the *makespan*, i.e., the maximum link congestion.

The study of this simple setting (i.e., the networks in the KP model) allows for a direct comparison with other approaches to cope with selfish agents (e.g., allowing selfish routing [19], suggesting a good Nash equilibrium [10], inducing good Nash equilibria via scheduling policies "internal" to the machines [4]). Nevertheless, the relative simplicity of this model makes it possible to isolate fundamental aspects of resource allocation problems involving selfish users. We indeed derive a general technique for designing mechanisms for generalizations of this basic problem, including (i) routing over arbitrary networks, (ii) cost functions other than the maximum link congestion, (iii) the case in which both machines and jobs are owned by selfish agents, (iv) agents owning more than one job, and (v) machines with different internal scheduling policies.

1.1 Our contribution

In this paper we investigate a general *Resource Assignment* (RA) game (which includes the notable example of the selfish routing game of Koutsoupias and Papadimitriou) from the following prospective based on *mechanism design*. In the *Mechanism Resource Assignment* (MRA) game we consider an *allocation algorithm* A which computes an allocation of requests to resources. Agents cannot directly choose the resources nor refuse the allocation chosen by the allocation algorithm. However, they may still manipulate the system by reporting false information about their requests (see Section 2 for a formal description of our model). The allocation algorithm is also allowed to charge each agent for the use of the resources. We are interested in allocation algorithms for which there exists a payment function P such that the strategies of reporting the truth about their own requests constitute a Nash equilibrium for all the agents. In this case the pair (A, P) is called a *NE-truthful* mechanism (see Definition 1). Allocations computed by A are associated with a cost and we would like A to output an allocation of minimum cost. Since in most cases, computing the minimum cost allocation is computationally hard, we consider *approximate* mechanisms and measure the quality of a mechanism (A, P) by its approximation ratio; that is, the ratio between the cost of the allocation computed by A and the cost of an optimal allocation.

In Section 3, we start by characterizing the class of allocation algorithms that induce a NE-truthful mechanism for *quasi one-parameter agents* (see Definition 5). The class of quasi one-parameter agents is a generalization of the well-studied class of one-parameter agents [21, 1]. Interestingly, agents of MRA games are *quasi one-parameter* (see Definition 3), though not one-parameter. We show that monotone algorithms, that characterize truthful mechanisms for one-parameter agents [21, 1], also characterize NE-truthful mechanisms for quasi one-parameter agents. This result represents the "kernel" of several interesting problems which can be formulated in terms of MRA games: For these problems, the existence of (optimal/polynomial-time) NE-truthful mechanisms reduces to the existence of (optimal/polynomial-time) monotone algorithms. We also show that rather simple MRA games do not admit exact mechanisms with dominant strategies.

Mechanism design for the KP model (Section 4). We focus on a particular MRA game, which we call *MKP game*, that is a mechanism design version of the routing problem in the KP model [19]: In our game, agents are not allowed to pick the link where to route their traffic; a *scheduler* allocates the traffic (i.e., jobs) on the links (i.e., parallel related machines) and the agents cannot refuse the allocation chosen by the scheduler. However, they may still manipulate the system by reporting false information on the size of their own jobs (see Section 2 for a formal description of our model). We investigate the benefits of replacing the "anarchic" policy of having agents choosing their own route with a scheduling algorithm which, combined with a suitable payment function, yields a *mechanism* inducing the agents to report the correct information (see Section 2 for a formal definition of these concepts). We measure such benefits by considering how good the makespan of the solution computed by the mechanism can be with respect to the optimal one. That is, we investigate the approximation ratio that a mechanism for our game(s) can achieve. Since the problem is NP-hard even for two machines with identical speeds, we focus on both exponential-time and polynomial-time mechanisms. In particular, the negative results on exponential-time mechanisms show that the inapproximability of the problem does not arise because of its computational intractability but it comes from the "lack of altruism" of the selfish agents.

We characterize the existence of approximation mechanisms depending on (the combination of)

the following factors:

- The ratio r between the largest and the smallest machine speeds. In more general settings, r quantifies how much resources can differ (e.g., in the problem of routing in general networks, r is the ratio between the longest and the shortest path towards the destination).
- Whether we consider mechanism inducing Nash equilibria or stronger ones with dominant strategies (i.e., *NE-truthful* or *truthful* mechanisms, formally defined in Sect. 2.1).
- The number m of machines.

We *characterize* the class of NE-truthful mechanisms in terms of *job-monotone algorithms*: Roughly speaking, these algorithms assign jobs "monotonically", that is, if we increase the size of one job then this job cannot be moved to a slower machine (see Definition 3). This condition is also necessary for mechanisms of the stronger type (i.e., truthful with dominant strategies), though some of our results imply that it is not sufficient. Moreover, this requirement can be considered as the "dual" of the monotonicity for scheduling problems involving selfish machines [1]: there, slowing down one machine causes the increase of its assigned work.

From our characterization we derive upper and lower bounds showing that a crucial factor is the ratio r between the machine speeds. For $r = 1$, every algorithm is job-monotone and thus exact solutions can be implemented *in general*, that is, *for any cost function* (see Corollary 12). If we consider our setting where the makespan is the cost function, $(1 + \epsilon)$-approximate solutions can be obtained in polynomial time. By contrast, it is impossible to obtain truthful mechanisms (i.e., mechanism for which truth-telling is a dominant strategy) that achieve approximation better than $5/4$ even if we allow exponential running time and consider two machines with the same speed (see Theorem 19).

For every $r > 1$, no NE-truthful mechanism can guarantee $(1 + \epsilon)$-approximate makespan, for some $\epsilon > 0$, even if we consider exponential-time mechanisms for the case of two machines only. Our results show that optimal solutions can be obtained if and only if resources are all of the same type (i.e., $r = 1$). The negative results for $r > 1$ are complemented by a constant-ratio NE-truthful mechanism for *any* number of machines (even non-constant) having *arbitrary speeds*. This mechanism is *online* and achieves a competitive ratio of 8 (see Table 1 for a summary of our results for the MKP game). Payments satisfy the natural *no positive transfer* condition, that is, no agent receives money from the mechanism and thus users pay for routing their traffic.

Notice that truthful mechanisms are stronger than NE-truthful mechanisms. Indeed, dominant strategies guarantee that, even in presence of "irrational" agents that deviate, truth-telling remains the strategy maximizing the utility of every other agent. This is not the case for NE-truthful mechanisms where other Nash equilibria are possible (i.e., with some of the agents being *not* truth-telling). However, reaching such "alternative" Nash equilibria may be "difficult" for the agents since they will have to "coordinate" among themselves. Moreover, our results show that, if we want to obtain "good" approximate solutions, then we have to content ourselves with NE-mechanisms (see Table 1 for the case $r = 1$). Obviously, lower bounds for NE-truthful mechanisms apply to truthful ones.

More general settings (Section 5). We consider a scenario in which *both jobs and machines are owned by selfish agents*. This extension of the MKP game represents a situation in which some users compete for the resources while others own them. For this game, we present an online

mechanism achieving a competitive ratio of 12 for *any* number of machines with *verifiable arbitrary speeds* (see Corollary 22).

Our characterization for quasi one-parameter agents yields a general technique for designing mechanisms for any MRA game. A natural application of these results is to the problem of routing n pieces of unsplittable traffic between n pairs of nodes in an *arbitrary graph*. For this problem, NE-truthful mechanisms are characterized by routing algorithms which do not shorten the length of the path used for connecting a pair of nodes if the corresponding traffic demand increases. Our negative results for the MKP model when $r > 1$ imply that such algorithms cannot minimize the maximum link latency even when links are identical (a network of the KP model can be seen as a network with identical links connecting a source to a destination via disjoint paths of different lengths). In other words, even for identical links, exact solutions can be achieved only if we assume (a rather simple) combinatorial structure of the network (e.g., parallel identical links as in the KP model).

Agents owning several jobs (Section 6). We conclude with another extension of the KP model which cannot be formulated in terms of quasi one-parameter agents (and thus, as an RA game). This extension considers the case in which agents own more than one job and it is motivated by a scenario in which users' traffic is routed by n selfish providers and providers offering a better service (that is lower latency) can charge higher fees (see Sect. 6 for a more detailed discussion of the model). We first show that no $\frac{\sqrt{33}-1}{4}$-approximate solution can be achieved even when considering exponential-time truthful mechanisms for the case of two identical machines and at most two jobs per agent. This result contrasts with the $(1+\epsilon)$-approximate mechanism for identical speeds, when each agent owns one job. Motivated by this negative result, we turn our attention to truthful approximate mechanisms and give upper and lower bounds on the approximation ratio of such mechanisms (see Table 2). Some of our positive results are obtained via new polynomial-time approximation algorithms which can be combined with suitable payment functions so to obtain NE-truthful mechanisms achieving the same approximation ratio.

Speed ratio $r = s_{\max}/s_{\min}$	**Lower bound**	**Upper bound** (for any m and any r)
$r = 1$	$5/4 - \epsilon$ deterministic and truthful even exp. time [Thm 19]	exact deterministic and NE-truthful (non poly-time) [Cor 12] $1 + \epsilon$ (deterministic, poly-time) [Cor 12]
$1 < r < 2$	$\min\left\{r, \frac{1}{2} + \frac{1}{r}\right\}$ [Thm 17] (deterministic, NE-truthful)	8 [Thm 15] (deterministic, NE-truth., and online)
r integer	$1 + \frac{r-1}{2r^2-r}$ [Thm 16] (deterministic, NE-truthful)	8 [Thm 15] (deterministic, NE-truth., and online)

Table 1: Our Results: (in-)approximability via (NE-)truthful mechanisms for the MKP game (routing on networks of the KP model or scheduling selfish jobs on related machines).

k versus m		Lower bound		Upper bound	
$k = 1$		1		$1 + \epsilon$	[Cor 12]
$k \leq m$	$k = 2$	$\frac{\sqrt{33}-1}{4}$	[Thm 23]	$3/2 + \epsilon$	[Thm 35]
	$k > 2$	$\frac{\sqrt{33}-1}{4}$		$2 - 1/m$	[Thm 31]
$k > m$	$m = 2$, k_i even for all i	4/3	[Thm 24]	$3/2 + \epsilon$	[Thm 35]
	$m = 4$, k_i even for all i	4/3		$3 + \epsilon$	[Cor 36]
	$m > 2$, m even	4/3		3	[Thm 33]
	$m > 2$, m odd	4/3		$3(1 + \frac{2}{m-1})$	[Cor 34]

Table 2: Our Results: (in-)approximability via deterministic NE-truthful mechanisms for identical machines. k_i denotes the number of jobs owned by agent i and $k = \max_i k_i$; lower bounds apply to exponential-time mechanisms as well, while upper bounds are provided via polynomial-time mechanisms.

1.2 Related work

A number of papers for (variants of) the KP model [19] have studied the problem of characterizing, computing, and bounding the cost of Nash equilibria for the corresponding routing problem [20, 8, 12, 11, 9, 10] (see also [28, 26] for a different model considering splittable traffic on arbitrary networks). In the anarchic scenario in which agents decide by themselves, the final solution (a Nash equilibrium) can have a cost $\Theta(\log m / \log\log\log m)$ times the optimum [8]; this ratio is the *price of anarchy* for the case of arbitrary speeds, while the case of identical speeds has only a slightly better price of anarchy of $\Theta(\log m / \log\log m)$ [19, 8, 18].

Feldmann *et al.* [10] show how to compute in polynomial time a Nash equilibrium for the KP model whose cost is at most $(1 + \epsilon)$-times the optimum, for every $\epsilon > 0$. They prove this result via a *Nashification* technique which converts any given scheduling with makespan C into a scheduling which is a Nash equilibrium and whose makespan is at most C.

Christodoulou *et al.* [4] suggested a way of reducing the effect of selfishness by changing the *internal* scheduling policy of the machines. A remarkable fact here is that, for *identical* speeds, this approach reduces the effect of selfishness in the KP model from $\Theta(\log m / \log\log m)$ down to $4/3 - 1/m$ [4]. [1] The authors also conjecture that this ratio is the best possible with this kind of approach (termed *coordination mechanism*).

The price of anarchy [19], introduced in the context of the KP model, can be seen as the worst-case approximation ratio that agents will reach "by themselves". However, it is not clear how agents will "converge" to a Nash equilibrium and Even-Dar *et al.* [9] indeed show that, for some natural strategies, this may take an exponential number of "moves" in the KP model (at each step some agent moves his/her piece of traffic to the link which currently gives the minimum latency). The price of anarchy has been studied also in other atomic congestion games [7, 27]. In recent papers [16, 13] games with static coalitions have been considered.

In the attempt to cope with the negative effects of selfishly acting agents, the elegant theory of *mechanism design* (see the milestone papers by Vickrey [29], Clarke [5] and Groves [15]) has been recently applied to a number of optimization problems arising in the context of network optimization

[1] As Christodoulou *et al.* [4] point out, their approach can be seen as redesigning a system so to improve its performance when selfish users are involved. In particular, a change in the machines' internal scheduling policy yields a variant of the KP model in which, although the network is the same, the latency of a piece of traffic is different.

[22, 23, 1]. These problems are resource assignment problems, typically arising from the Internet, in which the agents can lie about the "type" of resources they hold (e.g., about the links/machines processing time). Nisan and Ronen [22] pointed out that classical mechanism design techniques (namely VCG mechanisms [29, 5, 15]) were not suitable for certain scheduling/routing problems. Archer and Tardos [1] considered the problem of scheduling jobs on parallel related machines (i.e., each agent corresponds to a link in the network of the KP model). They observed that this problem belongs to a wider class of problems involving *one-parameter* agents (see Def. 3) for which the design of a truthful mechanism reduces to the problem of designing a *monotone* algorithm (see Def. 3). This result is closely related to a certain type of auctions in which the auctioneer (i.e., the mechanism) offers identical items to a set of buyers (i.e., selfish agents) considered by Myerson [21]. In this work we exploit the results in [21, 1], although our problems are not one-parameter (indeed, our MKP game is "harder" then its dual in [1] since the former does not admit exact solutions, while the latter does).

Roadmap. In Section 2 we formally define our model and the basic concepts/definitions used throughout the paper; in particular, Section 2.1 presents the mechanism design approach. In Section 3 we characterize NE-truthful mechanisms for quasi one-parameter agents and for MRA games. We apply these results to the MKP game in Section 4 where we prove upper and lower bounds (Sections 4.1 and 4.2, respectively). We extend this model in Section 5. In particular, in Section 5.1 we consider machines that are owned by selfish agents; in Section 5.2 we consider the problem of routing in general networks. Section 6 deals with the case of agents owning more than one job. The effects of changing the internal scheduling policy are discussed in Section 7, together with a number of open problems and possible research directions.

2 Resource assignment games

In this paper we consider the following *Resource Assignment* game, which we call the *RA* game. We have a set R of n requests and a set S of m resources. A *feasible solution* X is an allocation of the resources to the requests. We consider the general setting in which the i-th request is associated with a *weight* t_i and, for a feasible solution X, the processing time of request i is equal to

$$\mathsf{res}^i(X) \cdot t_i + \mathsf{add}^i(X|t_{-i}),$$

where t_{-i} denotes the vector $(t_1, t_2, \ldots, t_{i-1}, t_{i+1}, \ldots t_n)$ of the weights of all requests other than i. That is, the completion time of request i consists of two parts: the first part, $\mathsf{res}^i(X) \cdot t_i$, depends on the solution X and on the weight t_i of the request; the second part, $\mathsf{add}^i(X|t_{-i})$, depends on the solution X and on the weights t_{-i} of the other requests. For example, the requests can be communication requests of different weights between a source u and a destination v in a network and the resources are the edges of the underlying communication graph. A solution X satisfies each of the requests by assigning a u-v path to each of them. In this case, $\mathsf{res}^i(X)$ can be seen as a cost per unit of traffic, given the path that X assigns to i, and $\mathsf{add}^i(X|t_{-i})$ as an additional cost due to the fact that these resources must be shared with other users.

Each request is owned by a *selfish agent* who, naturally, would prefer solutions where his/her *own* request is processed faster. Specifically, in the RA game, each agent i associates his/her request to some set of resources (chosen according to some probability distribution) and the set

of resources chosen by each agent defines a solution X. Given the vector of the request weights $t = (t_1, t_2, \ldots, t_n)$, agent i assigns to solution X value $v^i(X|t)$ defined as

$$v^i(X|t) := -(\mathsf{res}^i(X) \cdot t_i + \mathsf{add}^i(X|t_{-i})),$$

that is, the opposite of the processing time of his/her own request (e.g., the time required for having his/her own traffic being transmitted). Hence, the *valuation function*[2] $v^i(X|t)$ expresses how much agent i "likes" the assignment X, given his/her request of weight t_i and the agent prefers solutions for which he/she gives a high valuation. This may be in contrast with the main goal of computing solutions which are "globally" optimal; that is, solutions which allocate resources so to minimize some cost function $\mathsf{cost}(X|t)$.

The KP game. The KP game [19] is a special case of the RA game. Here the resources are m parallel links of different speeds $(s_1, \ldots, s_m)$, the i-th request consists of one unsplittable piece of traffic of weight t_i and each request is to be allocated to one of the m links. If a link of speed s is allocated to requests of total weight w then *all* these requests will be completed in time w/s. Each agent owns one request (the generalization to more than one request per agent is given in Sect. 6) and wishes to minimize his/her *own* latency. The global goal instead is to minimize the maximum latency. The problem is easily seen to be equivalent to the following scheduling problem. We have n jobs of weights $(t_1, \ldots, t_n)$ which need to be allocated to a set of m related machines of speeds $(s_1, \ldots, s_m)$; the completion time of machine j in allocation X is equal to $w_j(X|t)/s_j$, where $w_j(X|t)$ is the sum of the weights of jobs in X^j, that is the set of jobs assigned by X to machine j. Jobs are processed in a round-robin fashion and thus all jobs assigned to the same machine finish at the same time. Thus, if job i is assigned to machine j then the completion time of job i can be written as

$$\frac{w_j(X|t)}{s_j} = \frac{t_i}{s_j} + \frac{\sum_{k \in X^j, k \neq i} t_k}{s_j}$$

and agent's i valuation of X is $v^i(X|t) = -\frac{w_j(X|t)}{s_j}$. One can cast the KP game into the more general setting of RA games by letting

$$\mathsf{res}^i(X) = 1/s_j, \qquad \mathsf{add}^i(X|t_{-i}) = \frac{1}{s_j} \sum_{k \in X^j, k \neq i} t_k, \tag{1}$$

where j is the machine to which X assigns job i.

The global optimization function is the *makespan*, that is, the maximum machine completion time:

$$\mathsf{cost}(X|t) = \max_{1 \leq j \leq m} \frac{w_j(X|t)}{s_j}.$$

The KP game models the case in which agents perform selfish routing over parallel links; that is, each agent chooses a link according to some probability distribution which maximizes his/her own valuation (i.e., the agent picks the probability distribution that minimizes the expected completion time of his/her own jobs). Koutsoupias and Papadimitriou [19] introduce the concept of *price of*

[2] Actually the term "payoff function" is more standard in the setting of a strategic game but we choose to use the term "valuation" since, as it will be clear in the next sections, we are interested in a mechanism design approach to resource assignment games.

anarchy or *coordination ratio*, that is, the ratio between the cost (i.e., makespan) of the worst Nash equilibrium and the optimal cost. For this specific game it has been shown that the price of anarchy is $\Theta(\log m / \log\log\log m)$ [8].

2.1 A mechanism design approach

In this work, we consider a mechanism design approach to the RA game. In the resulting game, which we call the *MRA* game, an *allocation* algorithm A computes a feasible allocation of requests to resources instead of allowing each agent to choose which resources to allocate to his/her request. Since it is unreasonable to assume that algorithm A has knowledge of the weight of each request, we assume that A elicits this information from the agent owning the request and computes an allocation based on the weights of the requests as reported by the agents. Agent i can "influence" the allocation algorithm by misreporting the weight t_i of his/her request to the algorithm. Based on the *reported values* $b = (b_1, \ldots, b_n)$, algorithm A computes a solution $A(b)$. Hence, agent i may report $b_i \neq t_i$ so to induce algorithm A to compute a solution that i likes better, i.e., with higher valuation. In this case, there is no guarantee that a c-approximation algorithm A returns a solution $A(b)$ which is a c-approximate solution for the *true input* t. In order to incentivate the agents to report their *true* values t, we define suitable *payment functions* $p_A^i(b)$ which determine the amount of money that each agent i receives according to the declared values b. The payment functions, as well as the allocation algorithm A, are *known to the agents*. In the literature a pair $M = (A, p_A)$ is termed a *mechanism*. So, each agent is now willing to maximize his/her *utility* (or net profit) $u_M^i(\cdot)$ which is defined as follows:

$$u_M^i(b|t) := p_A^i(b) + v^i(A(b)|t). \tag{2}$$

We assume that agents are selfish but rational, i.e., they declare a false value only if they can obtain a strictly larger utility. To stress that agent i can only change the i-th value b_i of b, we distinguish the declared values of the other agents by introducing the notation $b_{-i} := (b_1, b_2, \ldots, b_{i-1}, b_{i+1}, \ldots, b_n)$. For any value x, we set

$$(x, b_{-i}) := (b_1, b_2, \ldots, b_{i-1}, x, b_{i+1}, \ldots, b_n).$$

We consider two solution concepts that have been studied for mechanisms: NE-truthful mechanisms and mechanisms truthful with respect to dominant strategies. Roughly speaking, in a NE-truthful mechanism, truth-telling is a Nash equilibrium and thus no agent has an incentive to *unilaterally* change his/her strategy if the other agents say the truth (see, for example, [24]).

Definition 1 *A mechanism $M = (A, p_A)$ is* NE-truthful *if, for every agent i, it holds that for all t and for all b_i*

$$u_M^i(t_i, t_{-i}|t) \geq u_M^i(b_i, t_{-i}|t).$$

NE-truthful mechanisms are commonly known as Bayesian-Nash implementations [24]. A stronger solution concept consists in requiring that truth-telling is the best strategy for a player regardless of the strategy adopted by other players:

Definition 2 *A mechanism $M = (A, p_A)$ is* truthful with dominant strategies *(in short,* truthful*) if, for every agent i, for every reported values b_{-i} of the other agents, and for every b_i*

$$u_M^i(t_i, b_{-i}|t) \geq u_M^i(b_i, b_{-i}|t). \tag{3}$$

Our setting leads to a variant of the KP game, which we call the *MKP game*, in which agents are not allowed to pick the link to use for routing their traffic but instead an allocation algorithm A, based on the reported weights of the requests, assigns each request to a link. Thus, agents may misreport the weight of their traffic so to "indirectly" pick better links. Observe that, in our variant, we assume that the mechanism is not able to *verify* whether the agent is reporting the real weight b_i or a different one. It can only compute the allocation and the payments but it cannot verify the real cost of the solution. Indeed, in practice, it may be too expensive to keep track of the actual amount of traffic sent by each user, thus preventing from the possibility of checking, for instance, whether $b_i < t_i$. Moreover, a user may report $b_i > t_i$ and actually send an amount of traffic equal to b_i, by just padding the original traffic with some "fake" traffic up to the declared value b_i.

Notation. For the MKP game introduced above, we adopt the following notation. We let $A^i(b)$ denote the machine to which job i is allocated according to the allocation $A(b)$ computed by A on input the vector of reported values b and by $s^i_A(b)$ its speed.[3] Also, we let $w^i_A(b|t_{-i})$ denote the sum of the weights of jobs assigned to $A^i(b)$ except for t_i. As in the KP game [19] we define agent's i valuation of solution $A(b)$ as

$$v^i(A(b)|t) := -\frac{t_i + w^i_A(b|t_{-i})}{s^i_A(b)}. \tag{4}$$

The cost $\mathsf{cost}(A(b)|t)$ of solution $A(b)$, is the makespan with respect to the true input t. According to our terminology, this is

$$\mathsf{cost}(A(b)|t) := \max_i \left\{ \frac{t_i + w^i_A(b|t_{-i})}{s^i_A(b)} \right\} = \max_i \left\{ -v^i(A(b)|t) \right\}.$$

In the sequel, for the sake of readability, we will sometimes omit b_{-i}, t_{-i}, A and M in the definitions above and simply use $p^i(b_i)$, and $u^i(b_i|t)$. We also denote $\mathsf{cost}(A(b)|t)$ simply as $\mathsf{cost}(A, b)$.

3 A characterization of NE-truthful mechanisms

In this section we introduce the notion of a *quasi one-parameter* agent and characterize NE-truthful mechanisms for quasi one-parameter agents as those mechanisms for which the allocation algorithm is *monotone*. Since MRA games involve quasi one-parameter agents, this result characterizes NE-truthful mechanisms for all MKP games.

We start by reviewing the notions of a *one-parameter* agent and of a *monotone* allocation algorithm.

Definition 3 (one-parameter agents and monotone algorithms) *An agent i is* one-parameter *if his/her valuation is of the form $v^i(X|t) = -\mathsf{res}^i(X) \cdot t_i$ for some publicly known function $\mathsf{res}^i(\cdot)$. An algorithm A is* monotone (for functions res^i) *if, for every agent i, for all b_{-i}, and for all b_i and $b'_i < b_i$, it holds that $\mathsf{res}^i(A(b_i, b_{-i})) \leq \mathsf{res}^i(A(b'_i, b_{-i}))$.*

[3] A more appropriate notation would be $s_{A^i(b)}$ but for the sake of readability we use $s^i_A(b)$.

Myerson [21] proved that monotone algorithms A characterize truthful mechanisms in the case of one-parameter agents in the sense that an algorithm A admits payment functions P such that (A, P) is a truthful mechanism if and only if A is monotone. Archer and Tardos [1] gave an alternative form for the payments which can be used for obtaining polynomial-time mechanisms.

Theorem 4 ([21, 1]) *An algorithm A admits payment P such that $M = (A, P)$ is truthful for one-parameter agents with respect to functions res^i if and only if, for all i, A is monotone for res^i. If A is monotone with respect to res^i, then the payment functions P^i are of the form*

$$P^i(b_i, b_{-i}) := h_i(b_{-i}) + b_i \cdot \mathsf{res}^i(A(b_i, b_{-i})) - \int_0^{b_i} \mathsf{res}^i(A(u, b_{-i}))\, du \tag{5}$$

where the h_i's are arbitrary scaling *functions.*[4]

Next we introduce the notion of a *quasi one-parameter* agent and give a necessary and sufficient condition on an algorithm A for the existence of payment functions Q such that (A, Q) is NE-truthful.

Definition 5 (quasi one-parameter agents) *An agent i is* quasi one-parameter *if his/her valuation is of the form $v^i(X|t) = -\mathsf{res}^i(X) \cdot t_i - \mathsf{add}^i(X|t_{-i})$, for publicly known functions $\mathsf{res}^i(\cdot)$ and $\mathsf{add}^i(\cdot)$.*

We have the following necessary condition.

Theorem 6 *A mechanism $M = (A, Q)$ for quasi one-parameter agents with functions res^i and add^i is NE-truthful only if A is monotone with respect to functions res^i.*

Proof. Fix agent i and vector t_{-i}. For $\tau > 0$, set $Q^i(\tau) = Q^i(\tau, t_{-i})$, $\mathsf{res}^i_A(\tau) = \mathsf{res}^i(A(\tau, t_{-i}))$ and $\mathsf{add}^i_A(\tau) = \mathsf{add}^i(A(\tau, t_{-i}|t_{-i}))$.

Pick $x < y$ and consider the two cases in which the true type of agent i is $t_i = x$ and the case $t_i = y$. Being NE-truthful requires the following two inequalities to hold

$$\begin{aligned} Q^i(x) - \mathsf{res}^i_A(x) \cdot x - \mathsf{add}^i_A(x) &\geq Q^i(y) - \mathsf{res}^i_A(y) \cdot x - \mathsf{add}^i_A(y); \\ Q^i(y) - \mathsf{res}^i_A(y) \cdot y - \mathsf{add}^i_A(y) &\geq Q^i(x) - \mathsf{res}^i_A(x) \cdot y - \mathsf{add}^i_A(x). \end{aligned}$$

By summing them up we obtain

$$\mathsf{res}^i_A(x) \cdot x + \mathsf{res}^i_A(y) \cdot y \leq \mathsf{res}^i_A(y) \cdot x + \mathsf{res}^i_A(x) \cdot y,$$

that is

$$\left(\mathsf{res}^i_A(x) - \mathsf{res}^i_A(y)\right)(x - y) \leq 0.$$

Since $x < y$, we obtain $\mathsf{res}^i_A(x) \geq \mathsf{res}^i_A(y)$, thus implying that A is monotone. □

Next we show that monotonicity of A is also sufficient for the existence of payment functions Q such that (A, Q) is NE-truthful.

[4] We stress that we have a different payment scheme for each choice of the scaling functions h_i and thus we should have used the more precise (and more cumbersome) notation $P^i_{h_i}$.

Theorem 7 *Let A be a monotone algorithm with respect to res^i and let P be payment functions such that (A, P) is truthful for one parameter agents with respect to res^i. Then, for all functions add^i, (A, Q) is NE-truthful for quasi one-parameter agents with respect to functions res^i and add^i where payments are defined as*

$$Q^i(b) := P^i(b) + \mathsf{add}^i(A(b)|b_{-i}). \tag{6}$$

Proof. Observe that, by Definition 5, the utility of a quasi one-parameter agent with payment Q^i satisfies

$$\begin{aligned} u^i(b_i, t_{-i}|t) &= -\mathsf{res}^i(A(b_i, t_{-i})) \cdot t_i - \mathsf{add}^i(A(b_i, t_{-i})|t_{-i}) + P^i(b_i, t_{-i}) + \mathsf{add}^i(A(b_i, t_{-i})|t_{-i}) \\ &= P^i(b_i, t_{-i}) - \mathsf{res}^i(A(b_i, t_{-i})) \cdot t_i. \end{aligned}$$

In other words, the utility of the quasi one-parameter agent i (for functions res^i and add^i) with payment function Q^i is equal to the utility of the one parameter agent (for function res^i) with respect to payment function P^i. Since (A, P) is truthful we have that $u^i(t_i, t_{-i}|t) \geq u^i(b_i, t_{-i}|t)$ for all b_i, t_i and t_{-i}. □

We observe that, since MRA games involve quasi one-parameter agents, Theorems 6-7 apply to any MRA game. Thus, we have the following corollary.

Corollary 8 *Let A be an allocation algorithm for a MRA game. Then there exist payment functions Q such that $M = (A, Q)$ is a NE-truthful mechanism for this MRA game if and only if A is monotone for work functions res^i.*

According to the definition of monotone algorithm (Def. 3), the above result states that $\mathsf{res}^i(A(b_i, b_{-i}))$ must be monotone non-increasing in b_i, for all i and b_{-i}. This requirement has a natural interpretation: the algorithm is not allowed to worsen the resources allocated to an agent if his/her communication request increases.

4 Mechanisms for the MKP game

In this section we apply our characterization for MRA games to the MKP game. We have the following definition.

Definition 9 (job-monotone algorithms) *An allocation algorithm A for the MKP game is* job-monotone *if for all i, for all b_{-i}, and for all b_i and $b'_i < b_i$, it holds that $s^i_A(b_i, b_{-i}) \geq s^i_A(b'_i, b_{-i})$.*

Observe that algorithm A is a job-monotone allocation algorithm for the MKP game if and only if A is monotone with respect to functions res^i defined in Eq. 1. Hence, Theorems 6 and 7 imply the following theorem.

Theorem 10 *Let A be an allocation algorithm for the MKP game. Then there exist payment functions Q such that $M = (A, Q)$ is a NE-truthful mechanism for the MKP game if and only if A is job-monotone.*

A natural requirement for the mechanism is the satisfaction of the *NPT* (*no positive transfer*) condition; that is, no agent should be paid by the system to use the resources.

Theorem 11 *Let A be a job-monotone allocation algorithm for the MKP game. Then there exists payment function q_A such that (A, q_A) is NE-truthful and satisfies the NPT condition.*

PROOF. We let P denote the payment functions defined in Eq. 5 (see Theorem 4), relative to the scaling functions $h^i(b_{-i}) = -\frac{1}{s_{\min}} \sum_{k \neq i} b_k$, where $s_{\min}$ is the minimum speed of the m machines. By Theorem 4, since A is monotone with respect to the functions res^i defined in Eq. 1, (A, P) is NE-truthful for one-parameter agents with functions res^i. Now, define payments $q_A^i(b)$ as

$$q_A^i(b) := P^i(b) + \mathsf{add}^i(A(b)|b_{-i}),$$

where add^i is defined as in Eq. 1. By Theorem 7, (A, q_A) is NE-truthful for the MKP game. We next prove that (A, q_A) satisfies the NPT condition. We have

$$\begin{aligned} q_A^i(b) &= P^i(b) + \mathsf{add}^i(A(b)|b_{-i}) \\ &= h^i(b_{-i}) + \mathsf{add}^i(A(b)|b_{-i}) + b_i \cdot \mathsf{res}^i(A(b)) - \int_0^{b_i} \mathsf{res}^i(A(u, b_{-i}))\, du \\ &\leq h^i(b_{-i}) + \mathsf{add}^i(A(b)|b_{-i}) \end{aligned}$$

where the last inequality holds because $\mathsf{res}^i(A(u, b_{-i}))$ is monotone with respect to u. Observe that

$$\begin{aligned} h^i(b_{-i}) + \mathsf{add}^i(A(b)|b_{-i}) &= \frac{w^i(A(b)|b_{-i})}{s_A^i(b)} - \frac{1}{s_{\min}} \sum_{k \neq i} b_k \\ &\leq \frac{1}{s_{\min}} \left(w^i(A(b)|b_{-i}) - \sum_{k \neq i} b_k \right) \\ &\leq 0. \end{aligned}$$

Hence we have that $q_A^i(b) \leq 0$ and thus the theorem holds. □

4.1 Upper bounds for the MKP game

In this section we give upper bounds on the approximation ratio achievable by NE-truthful mechanisms for the MKP game.

We first consider the case of machines with identical speeds and observe that *every* allocation algorithm A is job-monotone (see Definition 9). Theorem 11 implies the following result.

Corollary 12 *If all the machines have identical speeds, then for any (c-approximate) algorithm A, there exist payment functions q_A such that $M = (A, q_A)$ is a (c-approximate) NE-truthful mechanism for the MKP game and M satisfies the NPT condition.*

The above result applies to *any* cost function for which A is c-approximate. In particular, for the problem of minimizing the makespan there exists a polynomial-time NE-truthful $(1 + \epsilon)$-approximation mechanism, for every $\epsilon > 0$ (see [17]).

For the case of machines of different speeds we provide an upper bound on the approximation achievable by NE-truthful mechanism for the MKP game. We do so by giving a sufficient condition

for an algorithm to be job-monotone and then showing that the online algorithm of [2] satisfies such a condition.

In general, an online algorithm A can be seen as consisting of two functions Π and Γ. Function Π takes as input the current allocation X_i, the size of the current job t_i and the speed of a machine s_j. When a new job arrives, A evaluates Π for all machines. The job will be allocated to a machine s_j for which $\Pi(X_i, t_i, s_j) = 1$. If this is the case for more than one machine, then the actual machine receiving the job is determined by evaluating function Γ on the current allocation X_i and on the set of machines $S^\star$ for which $\Pi(X_i, t_i, s_j) = 1$.

In the next definition, we define *regular* online algorithms as algorithms for which Π enjoys a monotonicity property and Γ selects the slowest machine from $S^\star$.

Definition 13 *An online allocation algorithm A for the MKP game is* regular *if there exist functions Γ and Π such that, for all speed vectors $(s_1, \ldots, s_m)$:*

1. *If $\Pi(A(t_1, \ldots, t_{i-1}), s_j, t_i) = 0$ then for all $t'_i > t_i$ it holds that $\Pi(A(t_1, \ldots, t_{i-1}), s_j, t'_i) = 0$ as well.*

2. *Γ assigns the i-th job of weight t_i to the slowest machine j for which $\Pi(A(t_1, \ldots, t_{i-1}), s_j, t_i) = 1$.*

It can be easily seen that the following theorem follows directly from the definition of regular algorithms.

Theorem 14 *A regular allocation algorithm A is job-monotone.*

Next theorem shows that there exist NE-truthful mechanism that guarantee constant approximation ratio.

Theorem 15 *There exists an 8-approximate NE-truthful mechanism for the online MKP game. This mechanism satisfies the NPT condition.*

Proof. Consider the online 8-competitive algorithm Assign-R presented in [2]. Each job is assigned to the *least capable machine*; that is, the slowest machine such that the cost of the resulting assignment stays below $\Lambda := 2\ell$, where ℓ is the minimum makespan. If the minimum makespan is not known, then a simple doubling technique is used.

It is easy to see that algorithm Assign-R is regular from which the theorem follows. □

4.2 Lower bounds

In this section, we give lower bounds on the approximation ratio achievable by (NE-)truthful mechanisms for the MKP game for machines of different speeds. The idea is the following: given a set of m machines, we construct two sets of jobs of weights t and t', where t' differs from t only for the weight of job j which is one of the jobs allocated to the fastest machine on input t. By Theorem 10, the allocation algorithm must be job-monotone and thus job j has to be allocated to the same machine also for instance t'. By selecting appropriately the weight t'_j we obtain that any optimal algorithm allocates this job to a different machine. Therefore the optimal allocation cannot be used in a NE-truthful mechanism and the mechanism must be sub-optimal. By carefully picking t and t' we can bound the achievable approximation ratio from below.

We present our lower bounds as a function of the ratio between the largest and the smallest machine speed. We define $r := s_{\max}/s_{\min}$, where $s_{\max} := \max_{1\le i\le m}\{s_i\}$ and $s_{\min} := \min_{1\le i\le m}\{s_i\}$. Notice that our lower bounds approach 1 when r goes to infinity. This is not surprising: the approximation ratio of the algorithm that assigns all jobs to the fastest machine tends to 1 as r grows.

In the proof of the lower bounds we use the notation "$\mathsf{opt}(x \to s)$" to denote the minimum cost of all allocations that assign the job of weight x to the machine of speed s.

Theorem 16 *For any two machines for which $2r$ is an integer, no deterministic (NE-) truthful mechanism for the MKP game can guarantee c-approximate solutions, for $c < 1 + \frac{r-1}{2r^2-r}$.*

Proof. Consider two machines of speed 1 and $r \ge 1$ and a set of $2r$ jobs of weight 1. Clearly, the optimum has cost at most 2. Consider a NE-truthful mechanism $M = (A, p)$.

If A assigns no jobs to the faster machine then the cost of the solution is $2r$ and the approximation ratio is at least r. Since $r \ge 1$, we have that

$$r \ge 1 + \frac{r-1}{2r^2 - r}$$

and the theorem follows.

Suppose now that A assigns at least one job to the faster machine. Let j be the index of one such a job. We consider the set of jobs of weights $t' = (1, \ldots, x, 1, \ldots, 1)$, where the j^{th} job has weight $x = 2 - 1/r$, and we show that A has to compute a non-optimal allocation on t'. By Corollary 10, A must allocate job j to the faster machine and thus $\mathsf{cost}(A, t') \ge \mathsf{opt}(x \to r)$. Moreover, if $\mathsf{opt}(x \to r)$ assigns two or more jobs to machine 1, then $\mathsf{opt}(x \to r) \ge 2$ since otherwise the work of machine of speed r is at least $x + 2r - 2 = 2r - 1/r$, thus implying $\mathsf{opt}(x \to r) \ge 2 - 1/r^2$. Also, observe that $\mathsf{opt}(t') = \mathsf{opt}(x \to 1) = \max\{x, (2r-1)/r\} = 2 - 1/r$. Putting things together

$$\frac{\mathsf{cost}(A, t')}{\mathsf{opt}(t')} \quad \ge \quad \frac{\mathsf{opt}(x \to r)}{\mathsf{opt}(x \to 1)} \ge \frac{2 - 1/r^2}{2 - 1/r} = 1 + \frac{r-1}{2r^2 - r}.$$

□

Theorem 17 *For any $m \ge 2$ machines with $r < 2$, no deterministic (NE-) truthful mechanism for the MKP game can guarantee c-approximate solutions, for any $c < \min\left\{r, \frac{1}{2} + \frac{1}{r}\right\}$.*

Proof. Consider m machines with speeds $(1, 1, \ldots, 1, r)$ and $m + 1$ jobs of weight 1. Any (NE-) truthful mechanism $M = (A, p)$ assigning no job to the fastest machine incurs a cost of at least 2, while the optimum is $2/r$. In this case the approximation ratio is at least r.

Let us thus assume that A assigns at least one job to the fastest machine, and let j be the index of such a job. Let us consider a new job sequence $t' = (1, \ldots, 1, x, 1, \ldots, 1)$, where the weight of the j^{th} job has been increased from 1 to $x > 1$. By Theorem 10, algorithm A cannot allocate this job to a slower machine. Hence, this job must be allocated to the fastest machine and $\mathsf{cost}(A, t') \ge \mathsf{opt}(x \to r)$. For $x = 2/r > 1$, we have $\mathsf{opt}(x \to r) = \min\{2, (1 + 2/r)/r\}$ as shown in Fig. 1. In the same figure, we prove that $\mathsf{opt}(t') \le 2/r$, thus implying that the approximation ratio of A is bounded from below by

$$\frac{\mathsf{cost}(A, t')}{\mathsf{opt}(t')} \quad \ge \quad \frac{\mathsf{opt}(x \to r)}{\mathsf{opt}(x \to 1)} \ge \frac{\min\{2, (1 + 2/r)/r\}}{2/r} = \min\left\{r, \frac{1}{2} + \frac{1}{r}\right\}.$$

solutions	speed 1	$\cdots$	speed 1	$\cdots$	speed 1	speed r	cost
case 1	1	$\ldots$	1		1	1, x	$(1+2/r)/r$
case 2	1	$\ldots$	1, 1	$\ldots$	1	x	2

Figure 1: Assignments of jobs used in the proof of Theorem 17: we have m jobs of weight 1 and one job of size $x = 2/r$. For $r < 2$, the optimum solution has cost $2/r$. Any solution assigning the job of size x to machine of speed $r > 1$ has cost at least $\min\{2, (1 + 2/r)/r\}$.

□

Since $r \leq \frac{1+\sqrt{17}}{4}$ implies that $r \leq \frac{1}{2} + \frac{1}{r}$, we obtain the following result.

Corollary 18 *For $m \geq 2$ machines and for any $c \leq \frac{1+\sqrt{17}}{4}$, no deterministic (NE-) truthful mechanism for the MKP game can guarantee c-approximate solutions.*

We have seen that, for the case of machines of identical speeds, there exists a NE-truthful optimal (albeit exponential-time) mechanism for the makespan (see Corollary 12). Next, we show that no truthful mechanism with respect to dominant strategies can guarantee approximation ratio better than 5/4 even for the case of two machines with the same speed and even if we allow non-polynomial mechanism. Intuitively, this is due to the fact that the valuation assigned by an agent i to a solution depends on the types of all agents. Instead, in the "dual" problem of scheduling with selfish machines for which there exists an optimal mechanism (see [1]), the valuation of agent i depends only on his/her own type t_i and on some other public input (i.e., the size of the jobs).

Theorem 19 *For any $c < 5/4$, no deterministic truthful mechanism (with dominant strategies) for the MKP game can guarantee c-approximate solutions even for the case of two identical machines.*

Proof. Let $M = (A, p_A)$ be a truthful mechanism. We consider an instance consisting of three jobs and declarations $b' = (2, 1, 3)$ and $b'' = (4, 1, 3)$ with two machines with speed $s = 1$. We first show that, for at least one of b' and b'', M does not return the optimal allocation.

Suppose, by contradiction, that M is an exact mechanism and, since M is truthful, it must be the case that, for all t_2 and t_3, truth-telling is a dominant strategy for agent 1. Hence,

$$u_M^1(b'|(2, t_2, t_3)) \geq u_M^1(b''|(2, t_2, t_3))$$

and

$$u_M^1(b''|(4, t_2, t_3)) \geq u_M^1(b'|(4, t_2, t_3)).$$

By Equations (2) and (4), this is equivalent to

$$p^1(b') - 2 - w_A^1(b'|(t_2, t_3)) \geq p^1(b'') - 2 - w_A^1(b''|(t_2, t_3))$$

and

$$p^1(b'') - 4 - w_A^1(b''|(t_2, t_3)) \geq p^1(b') - 4 - w_A^1(b'|(t_2, t_3)).$$

The above two inequalities can be rewritten as

$$p_A^1(b') - p_A^1(b'') \geq w_A^1(b'|(t_2, t_3)) - w_A^1(b''|(t_2, t_3))$$

and

$$p_A^1(b') - p_A^1(b'') \leq w_A^1(b'|(t_2,t_3)) - w_A^1(b''|(t_2,t_3))$$

which imply that

$$p_A^1(b') - p_A^1(b'') = w_A^1(b'|(t_2,t_3)) - w_A^1(b''|(t_2,t_3)). \quad (7)$$

It is easy to see that b' and b'' admit only one optimal allocation each. The following table shows the two optimal allocations

reported weights	machine 1	machine 2
b'	1, **2**	3
b''	**4**	1, 3

from which we have $w_A^1(b'|(t_2,t_3)) = t_2$ and $w_A^1(b''|(t_2,t_3)) = 0$. Therefore, by Eq. 7 we have that $p_A^1(b') - p_A^1(b'') = t_2$. This condition cannot hold for all t_2 as otherwise it implies that M knows t_2. Therefore, for at least one of b' and b'', M computes a sub-optimal solution. We conclude the proof by observing that the second best allocation for b' has makespan 4 and that the second best allocation for b'' has makespan 5. □

5 Extensions of the MKP game

In this section we discuss some extensions of MKP game. In Section 5.1 we consider a version of the game where selfish agents own also the machines and in Section 5.2 we consider the problem of routing selfish unsplittable traffic on arbitrary networks.

5.1 Selfish machines

In this section, we consider the following extension to the MKP game. We have two types of agents: *job-agents* owning jobs and *machine-agents* owning machines. We remark that each machine-agent owns only one machine and he/she is the only one to know the real speed of his/her machine. The allocation algorithm elicits from each job-agent the weight of his/her job and from each machine-agent the speed of his/her machine and, based on the reported data, the algorithm computes an allocation of the jobs to the machines. The valuation of a job-agent is the same as the one we have used in the previous sections. Instead, machine-agent i corresponding to machine i of speed s_i has valuation of the form $v^i(X|t_i) = -W^i(X)/s_i$, where $W^i(X)$ is the work assigned to machine i by solution X. We further assume that the machine-agents are verifiable. Specifically, payments are provided after the jobs have been completed by the agent and the agent receives a payment only if his/her machine completed the jobs within a time corresponding to the reported speed b_i and the work $W^i(X)$ assigned to it, that is, after $W^i(X)/b_i$ time units.

In the new scenario, we have to design the payments and the allocation algorithm in such a way that job-agents and machine-agents have an incentive to reveal the true weight of their jobs and true speed of their machines, respectively. We use algorithm Monotone-Assign-R from [3] that, for sake of completeness, is described in Figure 2. We notice that algorithm Monotone-Assign-R receives as input an upper bound Λ on the makespan of the optimal solution. The following theorem holds.

Theorem 20 ([3]) *There exist payments r_A such that M = (Monotone-Assign-R, r_A) is a 12-competitive polynomial-time online truthful mechanism (with dominant strategies) for m selfish verifiable machines.*

Algorithm Monotone-Assign-R(s, Λ):
/* $s_1 \leq s_2 \cdots \leq s_m$; */
initialize $w'_j := 0$ and $w''_j := 0$ for $j = 1, 2, \ldots, m$;
set $w'_{m+1} = \infty$;

1. upon arrival of new job t_i do begin
2. let l be the slowest machine such that

$$((w''_l + t_i)/s_l \leq 2\Lambda) \wedge ((w'_l > 0) \vee (w'_{l+1} > 0));$$

3. assign t_i to machine l;
4. if $w'_l > 0$ then $w''_l := w''_l + t_i$ else $w'_l := t_i$; end.

Figure 2: An online weakly monotone algorithm for any number of machines.

We stress that in the above mechanism payments are computed online for each new job and at every time step the machines receive a *non-negative* payment.

Theorem 21 *Algorithm* Monotone-Assign-R *is regular.*

Proof. Let Π be defined as in Step 2 of Monotone-Assign-R (see Fig. 2) and for sake readability let A denote Monotone-Assign-R. We have

$$\Pi(A(t_1, \ldots, t_{i-1}), s_j, t_i) = ((w''_j + t_i)/s_j \leq 2\Lambda) \wedge ((w'_j > 0) \vee (w'_{j+1} > 0)).$$

From the above we have that if $\Pi(A(t_1, \ldots, t_{i-1}), s_j, t_i) = 0$ then for any $t'_i > t_i$ it holds that $\Pi(A(t_1, \ldots, t_{i-1}), s_j, t'_i) = 0$. Moreover Monotone-Assign-R allocates job t_i to the slowest machine j for which $\Pi(A(t_1, \ldots, t_{i-1}), s_j, t_i) = 1$. Thus Conditions 1 and 2 of Definition 13 are satisfied and Monotone-Assign-R is regular. □

Consider now mechanism (Monotone-Assign-R, p_A), where p_A are payment functions for both job-agents and machine-agents defined as follows: for each job-agent i, $p^i_A(b) := q^i_A(b)$, where q_A are the payment functions defined in Theorem 11; for each machine-agent j, $p^j_A(b) := r^j_A(b)$, where r_A are the payment functions defined in Theorem 20. Since Monotone-Assign-R is regular, from Theorems 14, 10 and 20, we obtain the following general result.

Corollary 22 *There exists an online* 12*-competitive polynomial-time mechanism for the problem of scheduling n selfish jobs on m selfish machines such that:*

1. *For the n selfish agents owning the jobs, the mechanism is NE-truthful and satisfies the NPT condition.*
2. *For the m agents owning the machines, if the machines are verifiable, then the mechanism is truthful (with dominant strategies), satisfies the NPT condition and the voluntary participation condition (i.e., truth-telling agents have non-negative utilities).*

5.2 More games with quasi one-parameter agents

Corollary 8 can be applied to a routing game on general graphs as opposed to graphs consisting of a collection of parallel edges as in the MKP game. In particular, we are given a network $G = (V, E, l)$, with $l_e = 1/s_e$ and s_e being the speed of link $e \in E$; moreover, we have n selfish users, each of them corresponding to a triple $(\sigma_i, \delta_i, t_i)$, with $\sigma_i, \delta_i \in V$ and $t_i > 0$. A feasible solution is a set $X = \{X^1, \ldots, X^n\}$ of n paths, one for each agent, such that path X^i connects σ_i to δ_i in G. User i sends an amount of traffic t_i trough the links in X^i, and traversing a link e takes time $(t_i + T^e(X|t_{-i}))/s_e$. The quantity $T^e(X|t_{-i})$ is due to the traffic that solution X sends on link e together with traffic t_i; e.g., the amount of work that traffic t_i finds on link e once entering on this link. The valuation of agent i is thus equal to

$$v^i(X|t) := -\sum_{e \in X^i} \frac{t_i + T^e(X|t_{-i})}{s_e},$$

that is, the opposite of the time required to transmit from σ_i to δ_i. This shows that in this general routing game, agents are quasi one-parameter and thus, by Theorem 6 and 7, a routing algorithm A admits payment functions P such that (A, P) is NE-truthful if and only if A is *length monotone*. More precisely, algorithm A is length monotone if, as the weight of request i grows and the weights of other requests do not change, algorithm A assigns paths of decreasing length to request i (the length of path X^i is $\sum_{e \in X^i} 1/s_e$).

6 Agents owning more than one job

In this section we investigate the version of the MKP game in which an agent may own more than one job and machine speeds are identical. We have m machines of speed s, l jobs of weight $(t_1, \ldots, t_l)$, and $n < l$ agents. Throughout this section we make use of the following notation:

J^i	the set of jobs owned by agent i
k_i	the number of jobs owned by agent i, that is, $k_i = \|J^i\|$
t_{-i}	the vector of the true weights of all jobs owned by agents other than i
b_{-i}	the vector of the declared weights of all jobs owned by agents other than i
own_j	the set of jobs with the same owner as job j, that is, $own_j = J^i$ for all $j \in J^i$
$A^j(b)$	the set of jobs that solution $A(b)$ assigns to the same machine as job j (including job j itself)
$w^j_A(b\|t)$	the sum of the real weights of the jobs of $A^j(b)$, that is, $w^j_A(b\|t) = \sum_{h \in A^j(b)} t_h$
$o^j_A(b\|t)$	the sum of the real weights of the jobs in $A^j(b)$ that do not belong to own_j, that is, $o^j_A(b\|t) = \sum_{h \in A^j(b) \setminus own_j} t_h$
$m^j_A(b)$	the *number* of jobs in $A^j(b)$ that belong to own_j, that is, $m^j_A(b) = \|A^j(b) \cap own_j\|$

The valuation of agent i is equal to minus the sum of the finish times of his/her jobs, that is,

$$v^i_A(b|t) := -\sum_{j \in J^i} \frac{w^j_A(b|t)}{s}. \tag{8}$$

This corresponds to the case in which each customer pays the agent controlling his/her piece of traffic a fixed amount minus the experienced latency of his/her traffic and each agent wants to maximize the amount of money received from the customers.

When algorithm A is clear from the context or immaterial, we will drop the subscript "A" and simply write $m^j(b), w^j(b|t)$, $v^i(b|t)$, and $o^j(b|t)$. Without loss of generality we assume that each machine has speed $s = 1$.

6.1 Lower bounds

In this section we prove lower bounds on the approximation ratio obtained by truthful mechanisms in the case where agents may own more than one job. Our proofs adopt the following strategy: we fix agent i and the vector t_{-i} of the weights of the jobs owned by agents than i and consider two possible declarations b' and b'' for agent i; any (NE-) truthful mechanism $M = (A, p)$ must guarantee that

$$p^i(b', t_{-i}) + v^i_A(b', t_{-i}|b') \geq p^i(b'', t_{-i}) + v^i_A(b'', t_{-i}|b') \tag{9}$$

and

$$p^i(b'', t_{-i}) + v^i_A(b'', t_{-i}|b'') \geq p^i(b', t_{-i}) + v^i_A(b', t_{-i}|b''). \tag{10}$$

From the above equations, we derive necessary conditions on the payment function and the allocation algorithm of a truthful mechanism which in turn imply a lower bound on the approximation ratio. As the next two theorems show, if agents are allowed to own more than one job and machines have the same speed, then, unlike the case studied in the previous section, not all algorithms A admit payments P so that (A, P) is NE-truthful.

We start with a lower bound on the achievable approximation ratio when agents are allowed to own at most 2 jobs.

Theorem 23 *For any $m \geq 2$ and for any $c < \frac{\sqrt{33}-1}{4}$, no (NE-)truthful mechanism can guarantee c-approximate solutions on m machines of equal speeds when there is at least one agent that owns 2 jobs.*

Proof. Let $M = (A, p)$ be a truthful mechanism for this problem. Let us consider an instance with $m = 2$ machines, $n = 2$ agents, $l = 3$ jobs and $J^1 = \{1, 2\}$ and $J^2 = \{3\}$. We consider declarations $b' = (x', y')$ and $b'' = (x'', y'')$ for agent 1, with $x' \leq y' < 1$ and $x'' \leq 1 < y''$, and set $t_{-1} = (1)$. Observe that, if M is an optimal mechanism, then the allocation algorithm A must allocate the two smallest jobs on the same machine and the largest one on the other machine. Thus, on input b' or b'', algorithm A should produce the following two allocations:

instance	machine 1	machine 2
b'	x', y'	1
b''	$x'', 1$	y''

Since M is (NE-) truthful, Eq.s 9-10 hold and payments must satisfy

$$p^1(b') - 2(x' + y') \geq p^1(b'') - x' - y' - 1, \tag{11}$$

$$p^1(b'') - x'' - y'' - 1 \geq p^1(b') - 2(x'' + y''). \tag{12}$$

These two inequalities hold both only when $x'' + y'' \geq x' + y'$. Hence, we conclude that, if M is (NE-) truthful and $x'' + y'' < x' + y'$, then A cannot give an optimal allocation on both inputs b' and b''.

We now give a lower bound on the approximation ratio of the solution given by A. Consider vectors $b' = \left(\frac{\sqrt{33}+3}{12}, \frac{\sqrt{33}+3}{12}, 1\right)$ and $b'' = \left(\frac{\sqrt{33}-3}{12} - \epsilon, \frac{\sqrt{33}+9}{12}, 1\right)$, for some arbitrary small $\varepsilon > 0$.

Observe that $x'' + y'' = x' + y' - \epsilon < x' + y'$ which implies that A cannot be optimal on both vectors. Observe that any sub-optimal allocation on b' must allocate the two jobs of agent 1 on different machines, while on b'' it must allocate them on the same machine. If A gives a sub-optimal allocation on input b', then the cost of $A(b')$ is at least $1 + x'$ and the approximation ratio is at least $\frac{1+x'}{2x'} = \frac{\sqrt{33}-1}{4}$. Instead, if A gives a sub-optimal allocation on b'', then the cost of $A(b'')$ is at least $x'' + y''$ and the approximation ratio is at least $\frac{x''+y''}{1+x''} = \frac{\sqrt{33}-1}{4} - \beta$, where β is arbitrary small (is equal to $\frac{\epsilon}{1+x''}$). Hence the theorem follows. □

The above theorem also applies to the case in which agents may own more than two jobs. However, in this case, we can obtain a better lower bound.

Theorem 24 *For any $m \geq 2$ and for any $c < 4/3$, no truthful mechanism can guarantee c-approximate solutions on m machines of equal speeds when each agent owns $k \geq m$ jobs. This holds also for instances with two machines and one agent owning four jobs.*

PROOF. We prove the theorem for $m = 2$ machines of unitary speed. Let $M = (A, p)$ be a truthful mechanism for this problem. Let us consider an instance with one agent and four jobs and consider two vectors of declared weights $b' = (1, 1, 1, 1)$ and $b'' = (x, x, x, 3x - 3)$ for some $x > 0$. Observe that if M is an exact mechanism, then for sufficiently large x, A computes the following allocations:

instance	machine 1	machine 2
b'	$1, 1$	$1, 1$
b''	x, x, x	$3x - 3$

Equations 9-10 imply that

$$p^1(b') - p^1(b'') \geq -2$$

and

$$p^1(b') - p^1(b'') \leq -3,$$

proving that A cannot compute an optimal allocation on both b' and b''. Thus, if A computes a sub-optimal solution on b' (assigning at least 3 jobs to a machine) then its solution costs at least 3 while the optimum is 2. Instead, if A computes a sub-optimal solution on b'' (assigning an even number of jobs to each machine), then its solution costs at least $4x - 3$ while the optimum is $3x$. Hence, for any $\epsilon > 0$, there exists a sufficiently large x such that the approximation ratio of A is at least $(4/3 - \epsilon)$. This completes the proof. □

6.2 Upper bounds

In this section we will provide a constant-approximation NE-truthful mechanism for the case of an arbitrary number of machines with unitary speed. The main idea is to develop new approximation algorithms for which the valuation functions can be rewritten as those of quasi one-parameter agents.

We start by observing the following fact.

Fact 25 *For each i, the valuation function of agent i can be rewritten as*

$$v_A^i(b|t) = -\sum_{j \in J^i} \left(o_A^j(b|t) + t_j \cdot m_A^j(b) \right). \tag{13}$$

PROOF. Each job $j \in J^i$ contributes $w_A^j(b|t)$ to the valuation $v_A^i(b|t)$. We split the quantity $w_A^j(b|t)$ in two parts: one due to the weights of the jobs owned by agent i that are assigned to the same machine as job j (we denote this quantity by $s_A^j(b|t)$) and a second part due to the weights of the jobs owned by oother agents and that are asigned to the same machine as job j (that is, $o_A^j(b|t)$). We can this write

$$v_A^i(b|t) = -\sum_{j \in J^i} \left(o_A^j(b|t) + s_A^j(b|t) \right).$$

The weight t_j of job $j \in J^i$ appears in the sum $\sum_{j \in J^i} s_A^j(b|t)$. once for each of the $m_A^j(b|t)$ jobs of J^i that are schedules on the same machine as job j. This concludes the proof. □

Let us consider the class of algorithms computing solutions for which the quantity $m^i(b)$ is constant, for any b.

Definition 26 (independent algorithms) *An algorithm A is* independent *if, for every b, b', it assigns jobs to machines in such a way that, for any task j, $m_A^j(b) = m_A^j(b')$. In this case, we will just write m_A^j instead of $m_A^j(b)$.*

The next theorem proves that independent algorithms can be used to design NE-truthful mechanisms for the case where agents own several jobs.

Theorem 27 *For any independent algorithm A, there exist payment functions p_A such that $M = (A, p_A)$ is NE-truthful. Moreover, if A is a polynomial-time algorithm then payments p_A are computable in polynomial-time.*

PROOF. For every agent i we define the payment function

$$p_A^i(b) := \sum_{j \in J^i} o_A^j(b|b). \tag{14}$$

Assume that $b_{-i} = t_{-i}$. Then, we have that $o_A^j(b|b) = o_A^j(b|t)$. Moreover, since algorithm A is independent, by Eq. 13 we have that

$$\begin{aligned} u^i(b|t) := p_A^i(b) + v_A^i(b|t_i) &= \sum_{j \in J^i} o_A^j(b|b) - \sum_{j \in J^i} \left(o_A^j(b|t) + m_A^j \cdot t_j \right) \\ &= \sum_{j \in J^i} \left(o_A^j(b|b) - o_A^j(b|t) \right) - \sum_{j \in J^i} m_A^j \cdot t_j \\ &= -\sum_{j \in J^i} m_A^j \cdot t_j. \end{aligned}$$

Hence, for $b_{-i} = t_{-i}$, the utility of agent i does not depend on his/her declarations and (A, p_A) is a NE-truthful mechanism.

To conclude the proof, observe that payments $p_A^i(\cdot)$ in Eq. 14 are computable in polynomial time if A runs in polynomial time. □

We now give two algorithms for allocating selfish jobs to identical machines when any agent may own several jobs. We start by considering the case in which, for each agent i, $k_i \leq m$ or k_i is

a multiple of m and give an independent algorithm `spread` for this simple case (see Fig. 3). Then, we show how this algorithm can be used as a subroutine to design a polynomial-time mechanism that is NE-truthful in the general case.

Algorithm `spread` considers one agent at a time and it allocates all jobs of agent i before considering jobs of agent $i+1$. The algorithm spreads the jobs of agent i evenly among the machines according to the following rule: the jobs of agent i are partitioned into subsets J_h^i each of cardinality m (with the possible exception of the last set which may have less than m jobs); then each of the m machines receives exactly (or at most, if the subset contains less than m jobs) one of the m jobs of subset J_h^i; each job of the set J_h^i is assigned to the least loaded machine.

Algorithm `spread`(b, m)

for $i = 1$ to n do
  set $c_i = \lceil k_i/m \rceil$;
  partition J^i into c_i sets $J_1^i, J_2^i, \ldots, J_{c_i}^i$, each of size m except, possibly, for $J_{c_i}^i$ that may have fewer jobs.
  for $h = 1$ to c_i
    for $j \in J_h^i$ in non-increasing order by declared weight do
      assign job j to the least loaded machine that has been assigned
      no other job from J_h^i;

Figure 3: The algorithm `spread`.

Lemma 28 *If for each agent i, $k_i \leq m$ or k_i is a multiple of m, then algorithm* `spread` *is independent.*

Proof. For any agent i, consider a job $j \in J^i$. If $k_i \leq m$, then for any input b, algorithm `spread` allocates at most one job from agent i on each machine and $m_A^j(b) = 1$. Instead, if $k_i = c_i \cdot m$, then for any input b, each machine receives exactly c_i jobs from agent i and $m_A^j(b) = c_i$. □

We now prove that `spread` leads to a polynomial-time NE-truthful 2-approximate mechanism for the case where, for each agent i, $k_i \leq m$ or k_i is a multiple of m.

We start by studying the approximation factor guaranteed by algorithm `spread`. For the case $k \leq m$, algorithm `spread` is a simple variant of the greedy algorithm in which no machine receives two jobs from the same agent. We prove that the difference between the maximum and the minimum load assigned to the machines by algorithm `spread` is bounded from above as for the greedy algorithm. Let L_h^i denote the load of machine h after algorithm `spread`, on input the vector b, has assigned all jobs of agent i. Let

$$L_{\max}^i := \max_{1 \leq h \leq m} L_h^i, \qquad L_{\min}^i := \min_{1 \leq h \leq m} L_h^i \tag{15}$$

and $b_{\max} := \max_{1 \leq j \leq l} b_j$. The following technical lemma holds.

Lemma 29 *If for each agent i, $k_i \leq m$, then we have that, for each agent i, $L_{\max}^i - L_{\min}^i \leq b_{\max}$.*

PROOF. We prove the lemma by induction on the agent index i. The lemma clearly holds after the algorithm examined all jobs of agent $i = 1$ (there is at most one job per machine). Let us assume that the lemma holds after the algorithm examined all jobs of agent i and consider how the algorithm allocates jobs of agent $i+1$. Let α and β be the indexes of two machines with maximum and minimum load, respectively, after the algorithm allocated all jobs of agent $i+1$. We remark that, since $k_{i+1} \leq m$, algorithm `spread` assigns at most one job of agent $i+1$ to each machine. Let x and y denote the weights of the jobs of agent $i+1$ assigned to machines α and β, respectively (if no such a job is assigned then we simply consider a "dummy" job of weight 0). Then, we have

$$L^{i+1}_{\max} - L^{i+1}_{\min} = L^i_\alpha + x - L^i_\beta - y.$$

We distinguish two possible cases:

($L^i_\alpha \leq L^i_\beta$). Since $y \geq 0$, we have that

$$L^{i+1}_{\max} - L^{i+1}_{\min} \leq L^i_\beta + x - L^i_\beta - y \leq x \leq b_{\max}.$$

($L^i_\alpha > L^i_\beta$). If $x \leq y$, then by the inductive hypothesis we have that

$$L^{i+1}_{\max} - L^{i+1}_{\min} \leq L^i_\alpha - L^i_\beta \leq b_{\max}.$$

On the other hand, if $x > y$ we get a contradiction. In fact, in this case algorithm `spread` allocates the job of weight x before the job of weight y. When the algorithm allocates the job of weight x machines α and β have load L^i_α and L^i_β, respectively. Since, by hypothesis, $L^i_\alpha > L^i_\beta$ the algorithm cannot allocate this job to machine α.

This completes the proof. □

Lemma 30 *If for each agent i, $k_i \leq m$ or k_i is a multiple of m, then algorithm* `spread` *is $(2-1/m)$-approximate.*

PROOF. Consider first the case in which, for each agent i, $k_i \leq m$. Let $L^i_{\max}$ and $L^i_{\min}$ be defined as in Eq. 15. Define $\Delta = L^n_{\max} - L^n_{\min}$. Observe that

$$\mathsf{opt}(b) \geq (\sum_{i=1}^{l} b_i)/m \geq (L^n_{\min} \cdot (m-1) + L^n_{\max})/m = L^n_{\min} + \Delta/m$$

and thus

$$L_{\min} \leq \mathsf{opt}(b) - \Delta/m.$$

Moreover

$$\mathsf{opt}(b) \geq b_{\max} \geq \Delta$$

where the second inequality follows from Lemma 29. Thus, we have that

$$\begin{aligned}
\mathsf{cost}(\texttt{spread}, b) &= L^n_{\max} \\
&= L^n_{\min} + \Delta \\
&\leq \mathsf{opt}(b) - \Delta/m + \Delta \text{ (by the above equations)} \\
&= \mathsf{opt}(b) + \Delta(1 - 1/m) \\
&\leq \mathsf{opt}(b) + \mathsf{opt}(b)(1 - 1/m) \text{ (by the above equations)} \\
&= \mathsf{opt}(b)(2 - 1/m).
\end{aligned}$$

Algorithm `split`(b, m)

01. partition the machines in two sets S_1 and S_2 of
    cardinality $\lceil m/2 \rceil$ and $\lfloor m/2 \rfloor$;
02. for $i = 1$ to n
03. write k_i as $k_i = c_i \cdot \lceil m/2 \rceil + q_i$ for some
    $0 \leq q_i < \lceil m/2 \rceil$;
04. partition J^i in $J_1^i(b_i)$ containing the $c_i \cdot \lceil m/2 \rceil$ smallest jobs
    and $J_2^i(b_i)$ containing the remaining q_i jobs;
05. let $\beta^1 = \cup_{i=1}^n J_1^i(b_i)$ and $\beta^2 = \cup_{i=1}^n J_2^i(b_i)$;
06. schedule jobs in β^1 on machines of S_1 using algorithm `spread`;
07. schedule jobs in β^2 on machines of S_2 using algorithm `spread`;

Figure 4: An algorithm for the case $k > m$.

Consider, now, the case in which some agent i has a number of jobs which is a multiple of m, say $c_i \cdot m$ jobs for an integer $c_i > 0$. The algorithm partitions the set J^i into c_i sets $J_1^i, J_2^i, \ldots, J_{c_i}^i$ of size m. Thus, it is easy to observe that if we replace agent i with c_i agents $i_1, i_2, \cdots, i_{c_i}$, where agent i_h owns m jobs in J_h^i, the allocation computed by the algorithm does not change. Then, by the previous case we have that $\mathsf{cost}(\texttt{spread}, b) \leq \mathsf{opt}(b)(2 - 1/m)$. □

We can now state the following theorem.

Theorem 31 *There exist payment functions $p_{\texttt{spread}}$ such that $M = (\texttt{spread}, p_{\texttt{spread}})$ is a polynomial-time NE-truthful $(2 - 1/m)$-approximate mechanism for allocating selfish jobs to m identical machines if, for each agent i, $k_i \leq m$ or k_i is a multiple of m.*

Proof. By Lemma 28 algorithm `spread` is independent in the particular case where, for each agent i, $k_i \leq m$ or k_i is a multiple of m. Thus, by Theorem 27 there exist payment functions $p_{\texttt{spread}}$ such that $(\texttt{spread}, p_{\texttt{spread}})$ is a polynomial-time NE-truthful mechanism. Moreover, by Lemma 30, algorithm `spread` is $(2 - 1/m)$-approximate with respect to the declared weights. However, since the mechanism is NE-truthful, the declared weights coincide with the real weights and thus the mechanism is $(2 - 1/m)$-approximate. □

We now show how algorithm `spread` can be used as a subroutine in a constant-approximation mechanism that is NE-truthful in the general case where each agent owns any number of jobs. This mechanism is based on algorithm `split`, shown in Fig. 4. The algorithm partitions the machines in two sets S_1 and S_2 of size $\lceil m/2 \rceil$ and $\lfloor m/2 \rfloor$, respectively. Moreover, for each agent i it partitions the job set J^i in J_1^i and J_2^i in such a way that J_1^i contains a number of jobs that is a multiple of $\lceil m/2 \rceil$ while J_2^i contains at most $\lfloor m/2 \rfloor$ jobs. Then, it uses algorithm `spread` to allocate the jobs in J_1^i to the machines in S_1 and the jobs in J_2^i to the machines in S_2.

Theorem 32 *There exist payment functions $p_{\texttt{split}}$ such that $M = (\texttt{split}, p_{\texttt{split}})$ is a polynomial-time NE-truthful mechanism for allocating jobs to m identical machines when any agent may own several jobs.*

PROOF. For each agent i we define the payment function

$$p^i_{\texttt{split}}(b) := \sum_{j \in J^i} o^j_{\texttt{split}}(b|b). \tag{16}$$

Observing that when $b_{-i} = t_{-i}$ it holds that $o^j_{\texttt{split}}(b|b) = o^j_{\texttt{split}}(b|t)$, we have that

$$\begin{aligned} u^i_{\texttt{split}}(b|t) &= v^i_{\texttt{split}}(b|t) + p^i_{\texttt{split}}(b|t) \text{ (by definition)} \\ &= -\sum_{j \in J^i} \left(o^j_{\texttt{split}}(b|b) + t_j m^j_{\texttt{split}}(b) \right) + \sum_{j \in J^i} o^j_{\texttt{split}}(b|b) \text{ (by Eq. 13)} \\ &= -\sum_{j \in J^i} t_j \cdot m^j_{\texttt{split}}(b). \end{aligned}$$

We now observe that each machine of S_1 receives exactly c_i jobs of agent i and thus for each $j \in J^i_1(b)$ we have $m^j_{\texttt{split}}(b) = c_i$. On the other hand, each machine of S_2 receives at most one job of agent i and thus for each $j \in J^i_2(b)$ we have $m^j_{\texttt{split}}(b) = 1$. We thus have

$$-\sum_{j \in J^i} t_j \cdot m^j_{\texttt{split}}(b) = -\sum_{j \in J^i_1(b_i)} c_i \cdot t_j - \sum_{j \in J^i_2(b_i)} t_j.$$

This quantity is maximized when $J^i_2(b)$ contains the q_i jobs of J^i with the largest true weight; that is, when $J^i_2(b) = J^i_2(t)$. In other words, if $b_{-i} = t_{-i}$, then agent i can maximize his utility by setting $b_i = t_i$. □

Theorem 33 *If m is even, then Algorithm* `split` *is 3-approximate for allocating jobs to m identical machines.*

PROOF. Observe that if m is even then the algorithm partitions machines in two sets S_1 and S_2 of size $m/2$, and jobs in two sets β^1 and β^2, where, for each agent i, β^1 contains a number of jobs of J^i that is a multiple of $m/2$, while β^2 contains at most $m/2$ jobs of J^i. Then, it uses algorithm `spread` to allocate jobs in β^1 to machines in S_1 and jobs in β^2 to machines in S_2. Denote by cost_i the cost of the allocation of jobs in β^i to machines in S_i, for $i = 1, 2$. The cost of the allocation computed by algorithm `split` is obviously equal to $\max\{\mathsf{cost}_1, \mathsf{cost}_2\}$. Let $B_1 = \sum_{i \in \beta^1} b_i$, $B_2 = \sum_{i \in \beta^2} b_i$, and $B = B_1 + B_2$. Let $b^1_{max} = \max_{i \in \beta^1}\{b_i\}$, $b^2_{max} = \max_{i \in \beta^2}\{b_i\}$ and $b_{max} = \max\{b^1_{max}, b^2_{max}\}$. Notice that, by Lemma 30, it follows that $\mathsf{cost}_1 \leq \frac{B_1}{m/2} + b^1_{max}$ and $\mathsf{cost}_2 \leq \frac{B_2}{m/2} + b^2_{max}$. Therefore,

$$\begin{aligned} \mathsf{cost}(\texttt{split}, b) &= \max\{\mathsf{cost}_1, \mathsf{cost}_2\} \\ &\leq \max\{\frac{B_1}{m/2} + b^1_{max}, \frac{B_2}{m/2} + b^2_{max}\} \\ &\leq \max\{\frac{B_1}{m/2}, \frac{B_2}{m/2}\} + \max\{b^1_{max}, b^2_{max}\} \\ &\leq \frac{B_1 + B_2}{m/2} + b_{max} \\ &\leq 2\frac{B}{m} + b_{max} \\ &\leq 3\mathsf{opt}_m(b). \end{aligned}$$

□

Corollary 34 *If m is odd, then there exists a polynomial-time NE-truthful $3(1+2/(m-1))$-approximate mechanism for allocating jobs to m identical machines when any agent may own several jobs.*

PROOF. Consider the mechanism $(\mathtt{split}, p_{\mathtt{split}})$ but run it on only $m-1$ machines. Then, by Theorem 32 the mechanism is polynomial-time and NE-truthful. Moreover, by (the proof of) Theorem 33 we have that

$$\mathsf{cost}(\mathtt{split}, b) \leq 2\frac{B}{m-1} + b_{max} = 2\frac{m}{m-1}\frac{B}{m} + b_{max} \leq \left(3 + \frac{2}{m-1}\right)\mathsf{opt}_{m-1}(b).$$

□

The above results can be improved when considering small values of m. In particular, we prove the following theorem.

Theorem 35 *For every $\varepsilon > 0$, there exists a polynomial-time NE-truthful $(\frac{3}{2}+\varepsilon)$-approximate mechanism for allocating jobs on two identical machines, when each agent owns either a single job or an even number of jobs.*

PROOF. Consider the following algorithm: first run the Graham's PTAS [14] for 2 machines to get a $(1+\varepsilon)$–approximate solution X; then transform X into a new solution X' in the following way. For every agent i such that $k_i > 1$ and X allocates $c_i > k_i/2$ jobs on machine j (where j is either 1 or 2), move the $c_i - k_i/2$ lightest jobs of i from machine j to the other machine.

It is easy to see that this algorithm is independent. In fact, for each agent i, if $k_i > 1$ then the algorithm assigns $k_i/2$ jobs to each machine. Then, by Theorem 27 there exists a payment scheme for turning this algorithm into a NE-truthful mechanism.

It remains to analyze the approximation guarantee of the mechanism. Observe that the algorithm computes first a solution X that is $(1+\varepsilon)$-approximate and then computes a solution X' from X by moving some of the jobs. In particular, for each agent i, the algorithm may move the $c_i - k_i/2$ lightest jobs of agent i. The total weight of the moved jobs is at most half of the total weight of the jobs of agent i. Summing over all the agents, we have that at most half of the load of a machine can be moved to the other machine. It follows that, the maximum load of X' is at most $3/2$ times the maximum load of X. Since solution X is $(1+\varepsilon)$-approximate, we have that the solution computed by the algorithm is $(3/2+\varepsilon)$-approximate. □

We notice that for $m = 4$ we can get a better approximation ratio by simply using the algorithm for $m = 2$ and ignoring two machines. Clearly in this way we lose a factor of 2 in the approximation given by Theorem 35, obtaining a $(3+\epsilon)$-approximation.

Corollary 36 *For every $\varepsilon > 0$, there exists a polynomial-time NE-truthful $(3+\varepsilon)$-approximate mechanism for allocating jobs on four identical machines, when each agent owns either a single job or an even number of jobs.*

7 Conclusions and open problems

In this work, we have investigated a general resource assignment game (which includes the notable example of the selfish routing game of Koutsoupias and Papadimitriou) from a *mechanism design*

prospective. In the resulting MRA game agents cannot directly choose the resources nor refuse the allocation chosen by the allocation algorithm. However, they may still manipulate the system by reporting false information about their requests. The allocation algorithm is also allowed to charge each agent for the use of the resources. We feel this is a very general and natural scenario and other approaches (e.g., suggesting and allocation for the KP model which is a good Nash equilibrium [10]) should consider this aspect of the problem (i.e., the fact that computations are based on the information reported by the agents).

Our characterization of NE-truthful mechanisms for Resource Assignment games is quite intuitive: if the "weight" of an agent request increases, then the algorithm should not worsen the set of resources that are assigned to this agent. Rather surprisingly, this natural requirement prevents from obtaining arbitrary good approximate solutions, even for the simple scenario of the MKP game. Since this negative result holds no matter what the running time of the algorithm is, it can be seen as the price that we have to pay (in terms of performance degradation) when selfish agents are involved in the use of the resources. Payments allow to reduce significantly the system degradation that occurs in the "anarchic" KP model, but we still have to pay something!

We have also generalized the MKP game in several directions which consider important aspects of the problem. The topology of the underlying network is an important factor and our characterization for routing on arbitrary graphs implies that the maximum link congestion cannot be minimized even when all links are the same (the network in the KP model is equivalent to a network connecting the source to the destination via disjoint paths of different lengths). Our characterization holds also for cost functions other than the maximum link congestion as the monotonicity condition (Theorems 6-7) does not consider the objective function. Similarly, changing the internal scheduling policy as done in [4] does not affect the mechanism. Indeed, if we change the order in which jobs are executed, then we only affect the "additive" factor $\mathsf{add}^i(X|t_{-i})$ in the definition of quasi one-parameter agent. As this term is irrelevant for the monotonicity of the allocation algorithm, Theorems 6-7 still hold. Therefore, all of our positive/negative results apply and, interestingly, better approximation factors cannot be obtained in this way. Another interesting aspect concern the extension of the MKP game with selfish machines (Section 5.1): this variant provides a first example of a resource allocation problem in which we have both agents competing and agent owning the resources.

These aspects are considered here separately. An interesting future research direction is to combine them and see whether this affects the approximation guarantee of the mechanisms. It would be also interesting to consider resource allocation games involving agents whose costs functions are not of the form of the MRA games (i.e., not quasi one-parameter). This requires the development of new mechanism design techniques which should depart significantly from the one-parameter setting [21, 1]. A first candidate might be the extension of the MKP game involving more jobs per agent. Our positive results are obtained via independent algorithms which essentially reduce this problem to the quasi one-parameter setting. However, it is not clear whether NE-mechanisms can use different algorithms so to achieve a better approximation factor.

References

[1] A. Archer and E. Tardos. Truthful mechanisms for one-parameter agents. In *Proc. of the IEEE Symposium on Foundations of Computer Science (**FOCS**)*, pages 482–491, 2001.

[2] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM*, 44(3):486–504, May 1997.

[3] V. Auletta, R. De Prisco, P. Penna, G. Persiano. On Designing Truthful Mechanisms for Online Scheduling. In *Proc. of the 12th International Colloquium on Structural Information and Communication Complexity (**SIROCCO**)*, volume 3499 of *LNCS*, pag. 3-17, 2005.

[4] G. Christodoulou, E. Koutsoupias and A. Nanavati Coordination Mechanisms. In *Proc. of the 31st International Colloquium on Automata, Languages, and Programming (**ICALP**)*, volume 3142 of *LNCS*, pag. 345-357, 2005.

[5] E.H. Clarke. Multipart Pricing of Public Goods. *Public Choice*, pages 17–33, 1971.

[6] R. Cole, Y. Dodis, and T. Roughgarden. Pricing network edges for heterogeneous selfish users. In *Proc. of the Annual ACM Symposium on Theory of Computing (**STOC**)*, pages 521–530. ACM Press, 2003.

[7] R. Cominetti, J.R.Correa, N.E. Stier-Moses. Network games with atomic players. In *Proc. of the 30th International Colloquium on Automata, Languages, and Programming (**ICALP**)*, volume 4516 of *LNCS*, pages 525–536, 2006.

[8] A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. In *Proc. of the 13th Annual ACM Symposium on Discrete Algorithms (**SODA**)*, pages 413–420, 2002.

[9] E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence time to Nash equilibria. In *Proc. of the 30th International Colloquium on Automata, Languages, and Programming (**ICALP**)*, volume 2719 of *LNCS*, 2003.

[10] R. Feldmann, M. Gairing, T. Luecking, B. Monien, and M. Rode. Nashification and the coordination ratio for a selfish routing game. In *Proc. of the 30th International Colloquium on Automata, Languages, and Programming (**ICALP**)*, volume 2719 of *LNCS*, 2003.

[11] A. Ferrante and M. Parente. Existence of Nash Equilibria in Selfish Routing problems. Technical Report, Università di Salerno, 2002.

[12] D. Fotakis, S. Kontongiannis, E. Koutsoupias, M. Mavronicolas, and P. Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. In *Proc. of the 29th International Colloquium on Automata, Languages, and Programming (**ICALP**)*, volume 2382 of *LNCS*, pages 123–134, 2002.

[13] D. Fotakis, S. Kontongiannis, and P. Spirakis. Atomic congestion games among coalitions. In *Proc. of the 30th International Colloquium on Automata, Languages, and Programming (**ICALP**)*, volume 4516 of *LNCS*, pages 572–583, 2006.

[14] R.L. Graham. Bound on multiprocessor timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416–429, 1969.

[15] T. Groves. Incentive in Teams. *Econometrica*, 41:617–631, 1973.

[16] A. Hayrapetyan, E. Tardos, and T. Wexler. The Effect of Collusion in Congestion Games. *ACM Symposium on Theory of Computing (STOC)*, 2006.

[17] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.

[18] E. Koutsoupias, M. Mavronicolas, and P. Spirakis. Approximate equilibria and ball fusion, *TOCS*, 36: 683–693, 2003

[19] E. Koutsoupias and C.H. Papadimitriou. Worst-case equilibria. In *Proc. of Annual Symposium on Theoretical Aspects of Computer Science (**STACS**)*, volume 1563 of *LNCS*, pages 404–413, 1999.

[20] M. Mavronicolas and P. Spirakis. The price of selfish routing. In *Proc. of the Annual ACM Symposium on Theory of Computing (**STOC**)*, pages 510–519, 2001.

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

[22] N. Nisan and A. Ronen. Algorithmic Mechanism Design. In *Proc. of the 31st Annual ACM Symposium on Theory of Computing (**STOC**)*, pages 129–140, 1999.

[23] N. Nisan and A. Ronen. Computationally Feasible VCG Mechanisms. In *Proceedings of the 2nd ACM Conference on Electronic Commerce (**EC**)*, pages 242–252, 2000.

[24] M.J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[25] C. H. Papadimitriou. Algorithms, Games, and the Internet. In *Proc. of the 33rd Annual ACM Symposium on Theory of Computing (**STOC**)*, pages 749–753, 2001.

[26] T. Roughgarden. Designing networks for selfish users is hard. In *Proc. of the 42nd IEEE Symposium on Foundations of Computer Science (**FOCS**)*, pages 472–481, 2001.

[27] T. Roughgarden. Selfish Routing with Atomic Players. In *Proc. of the Symposium on Discrete Algorithms (**SODA**) 2005*, pages 1184–1185, 2005.

[28] T. Roughgarden and E. Tardos. How bad is selfish routing? In *Proc. of the 41st IEEE Symposium on Foundations of Computer Science (**FOCS**)*, pages 93–102, 2000.

[29] W. Vickrey. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance*, pages 8–37, 1961.