Cost-Sharing Scheduling Games on Restricted Unrelated Machines[*]

Guy Avni [†] Tami Tamir [‡]

May 31, 2016

Abstract

We study a very general cost-sharing scheduling game. An instance consists of k jobs and m machines and an arbitrary weighted bipartite graph denoting the jobs strategies. An edge connecting a job and a machine specifies that the job may choose the machine; edge weights correspond to processing times. Each machine has an activation cost that needs to be covered by the job assigned to it. Jobs assigned to a particular machine share its cost proportionally to the load they generate.

Our game generalizes singleton cost-sharing games with weighted players. We provide a complete analysis of the game with respect to equilibrium existence, computation, convergence and quality – with respect to the total cost. We study both unilateral and coordinated deviations.

We show that the main factor in determining the stability of an instance and the quality of a stable assignment is the machines' activation-cost. Games with unit-cost machines are generalized ordinal potential games, and every instance has an optimal solution which is also a pure Nash equilibrium (PNE). On the other hand, with arbitrary-cost machines, a PNE is guaranteed to exist only for very limited instances, and the price of stability is linear in the number of players. Also, the problem of deciding whether a given game instance has a PNE is NP-complete.

In our analysis of coordinated deviations, we characterize instances for which a strong equilibrium exists and can be calculated efficiently, and show tight bounds for the strong price of anarchy and the price of stability.

1 Introduction

In job-scheduling applications, jobs are assigned to machines to be processed. Many interesting combinatorial optimization problems arise in this setting, which is a major discipline in operations research. A centralized scheduler should assign the jobs in a way that achieves load balancing, an effective use of the system's resources, or a target quality of service [20]. Many modern systems provide service to multiple strategic users, whose individual payoff is affected by the decisions made by others. As a result, non-cooperative game theory has become an essential tool in the analysis of job-scheduling applications. We assume that each job is controlled by a player which has strategic considerations and acts to minimize his own cost, rather than to optimize any global objective. Practically, this means that the jobs

[*]A preliminary version of this paper appears in the proceedings of the 8th International Symposium on Algorithmic Game Theory (SAGT'15).

[†]School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel.

[‡]School of Computer Science, The Interdisciplinary Center, Herzliya, Israel.

choose a machine instead of being assigned to one by a centralized scheduler. In this paper we study the corresponding cost-sharing scheduling game (CSSG, for short) on restricted unrelated parallel machines.

An instance of CSSG is given by an arbitrary weighted bipartite graph whose vertex set consists of job-vertices and machine-vertices. The scheduling is restricted in a sense that not all machines are feasible to all jobs: each job is connected by edges to the machines that are able to process it. Edge weights specify the processing times, reflecting the load generated by the job on the machine. Scheduling on restricted unrelated machines is the most general model of scheduling on parallel machines.

In the corresponding game, the strategy space of a job is the set of machines that can process it. Each machine has an activation cost that needs to be covered by the jobs assigned to it. Cost-sharing games, in which players' strategies are subsets of resources and the resource's activation cost is covered by its users, arise in many applications, and are well-studied. Our game is different from previously studied games in several ways, each giving rise to new challenges. Previous work on cost-sharing scheduling games assume that either the activation-cost of a resource is shared uniformly by its users, or that players are weighted. To the best of our knowledge, this is the first time that scheduling on unrelated machines is analyzed as a non-cooperative cost-sharing game.

1.1 Preliminaries

An instance of CSSG is given by an arbitrary weighted bipartite graph whose vertex set is $\mathcal{J} \cup \mathcal{M}$, where $\mathcal{J}$ is a set of k jobs, and $\mathcal{M}$ is a set of m machines. Not all machines are feasible to all jobs: each $i \in \mathcal{J}$, has a set $M_i \subseteq \mathcal{M}$ of machines that may process it. For every job i and machine $j \in M_i$, it is known what the processing time $p_{i,j}$ of i on machine j is. The feasible sets and the processing times are given by the edges of the bipartite graph. Specifically, there is an edge (i, j) whose weight is $p_{i,j}$ for every $j \in M_i$.

Job i is controlled by Player i whose strategy space is the set of machines in M_i. Each machine $j \in \mathcal{M}$ has an activation cost, $c(j)$, which is shared by the jobs assigned to it, where the share is proportional to the load generated by the job.

A profile of a CSSG game is a vector $P = \langle s_1, s_2, \ldots, s_k \rangle \in (M_1 \times M_2 \times \ldots \times M_k)$ describing the machines selected by the players. For a machine $j \in \mathcal{M}$, we define the *load* on j in P, denoted $L_j(P)$, as the total processing times of the jobs assigned to machine j in P, that is, $L_j(P) = \sum_{\{i|s_i=j\}} p_{i,j}$. When P is clear from the context we omit it. The cost of Player i in the profile P is $cost_i(P) = \frac{p_{i,s_i}}{L_{s_i}(P)} \cdot c(s_i)$ and the cost of the profile P is $cost(P) = \sum_{1 \le i \le k} cost_i(P)$. Note that $cost(P)$ also equals the total activation-cost of non-idle machines, that is, $cost(P) = \sum_{j \in \cup_i s_i} c(j)$. Note that for a given profile, our cost-sharing scheme fits the proportional cost-sharing rule for weighted players, where the processing-times give the players' weights. This rule is commonly used (e.g., [22, 2, 12]) when the cost of a resource should split among its users proportional to their demand.

Consider a game G. For a profile P, a job $i \in \mathcal{J}$, and a strategy $s'_i \in M_i$, let (P_{-i}, s'_i) denote the profile obtained from P by replacing the strategy of Player i by s'_i. That is, the profile resulting from a migration of job i from machine s_i to machine s'_i. A profile P is a *pure Nash equilibrium* (NE) if no job i can benefit from unilaterally deviating from his strategy in P to another strategy; i.e., for every player i and every strategy $s'_i \in M_i$ it holds that $cost_i((P_{-i}, s'_i)) \ge cost_i(P)$.

Best-Response Dynamics (BRD) is a local-search method where in each step some player is chosen and plays its best improving deviation (if one exists), given the strategies of the other players. Since BRD corresponds to actual dynamics in real life applications, the question of BRD convergence and the quality of possible BRD outcomes are major issues in our study.

It is well known that decentralized decision-making may lead to sub-optimal solutions from the point of view of the society as a whole. We denote by OPT the cost of a social-optimal (SO) solution; i.e., $OPT = \min_P cost(P)$. We quantify the inefficiency incurred due to self-interested behavior according to the *price of anarchy* (PoA) [16, 19] and *price of stability* (PoS) [2, 24] measures. The PoA is the worst-case inefficiency of a pure Nash equilibrium, while the PoS measures the best-case inefficiency of a pure Nash equilibrium. Formally,

Definition 1.1 *Let $\mathcal{G}$ be a family of games, and let G be a game in $\mathcal{G}$. Let $\Upsilon(G)$ be the set of pure Nash equilibria of the game G. Assume that $\Upsilon(G) \neq \emptyset$.*

- *The* price of anarchy *of G is the ratio between the* maximal *cost of a PNE and the social optimum of G. That is, $PoA(G) = \max_{P\in\Upsilon(G)} cost(P)/OPT(G)$. The* price of anarchy *of the family of games $\mathcal{G}$ is $PoA(\mathcal{G}) = sup_{G\in\mathcal{G}} PoA(G)$.*
- *The* price of stability *of G is the ratio between the* minimal *cost of a PNE and the social optimum of G. That is, $PoS(G) = \min_{P\in\Upsilon(G)} cost(P)/OPT(G)$. The* price of stability *of the family of games $\mathcal{G}$ is $PoS(\mathcal{G}) = sup_{G\in\mathcal{G}} PoS(G)$.*

A firmer notion of stability requires that a profile is stable against *coordinated deviations*. A set of players $\Gamma \subseteq \mathcal{J}$ forms a *coalition* if there exists a joint move where each job $i \in \Gamma$ strictly reduces its cost. A profile P is a *Strong Equilibrium* (SE) if there is no coalition $\Gamma \subseteq \mathcal{J}$ that has a beneficial joint move from P [4]. The *strong price of anarchy* (SPoA) and the *strong price of stability* (SPoS) introduced in [1] are defined similarly, where $\Upsilon(G)$ refers to the set of strong equilibria.

In our study of CSSGs, we distinguish between unit-cost instances, in which all machines have the same activation cost, say $c(j) = 1$ for all $j \in \mathcal{M}$, and the general case, where $c(j)$ is arbitrary. We say that an instance has *machine-independent* processing-times if for every job i there is $p_i > 0$ such that $p_{i,j} = p_i$ for all $j \in M_i$.

1.2 Related Work

Game-theoretic analysis became an important tool for analyzing huge systems that are controlled by users with strategic consideration. In particular, systems in which a set of resources is shared by selfish users.

Congestion games [21] consist of a set of resources and a set of players who need to use these resources. Players' strategies are subsets of resources. Each resource has a latency function which, given the load generated by the players on the resource, returns the cost of the resource. We refer to the setting in which the latency functions are increasing as *congestion games* (the more congested the resource, the higher the waiting time), and we focus on *cost-sharing games* in which each resource has an activation cost that is shared by the players using it according to some sharing mechanism. For example, in network formation games, players have reachability objectives and strategies are subsets of edges, each inducing a simple path from the source to the target [2]. Players that use an edge uniformly share its cost. Such games always have a PNE and the PoS is logarithmic in the number of players.

Weighted cost-sharing games are cost-sharing games in which each player i has a *weight* $w_i \in \mathbb{N}$, and his contribution to the load of the resources he uses as well as his payments are multiplied by w_i. In [2] the authors study the counterpart of network formation games in the weighted cost-sharing setting. They show that every two-player game admits a PNE and that the PoS is an order of the number of players. Later, [7] closed the problem of PNE existence in these games by showing an example of a three-player game with no PNE. The authors also analyzed the inefficiency of weighted cost-sharing games with respect to the players' weights. In particular, for w_{max}, the maximal weight of a player, and W, the total players' weight, they show that for $\alpha = \Omega(\log w_{max})$, an α-approximate NE exists and the PoS is $O((\log W)/\alpha)$ with respect to $O(\alpha)$-approximate NE – outcomes from which no player can deviate and decrease its cost by more than an α-multiplicative factor.

Variants of cost-sharing games have been the subject of extensive research. In [17], games with player specific decreasing functions are studied. A different model, that studies cost-sharing games with non-anonymous cost functions is presented in [23]. The paper studies network cost-sharing games where the cost of each edge is a submodular function of its users, and this cost is shared using the *Shapley value*. It is well-known that proportional cost-sharing in weighted congestion games induces games in which PNEs need not exist [7, 10, 13]. On the other hand, a cost sharing mechanism based on the Shapley value, suggested and analyzed in [15], is shown to guaranteed the existence of PNE. We elaborate on other sharing mechanisms in Appendix A and show that CSSGs may not have a PNE even with this cost-sharing mechanism. A model of a job scheduling game, in which a job's cost is composed of both the load on its machine and its share in the machine's activation cost was introduced in [9]. Equilibrium inefficiency for this model was analyzed with respect to both the maximal cost of a player [9, 29, 30] and the total players' cost [6].

In a more general setting, players' strategies are multisets of resources. Thus, a player may need multiple uses of the same resource and his cost for using the resource depends on the number of times he uses the resource [5]. Such *multiset cost-sharing games* are less stable than classical cost-sharing games. Even very simple instances may not have a PNE and an equilibrium may be extremely inefficient (the PoS may equal the number of players) [5]. CSSGs can be viewed as a special case of multiset cost-sharing games in which in each multiset, a unique resource appears (possibly multiple times). Instances with machine-independent costs are a special case of weighted cost-sharing games in a similar manner.

A lot of attention has been given to scheduling congestion games (for a survey, see [27]), which can be thought of as a special case of congestion games in which the players' strategies are singletons. Most previous work assumes that the cost of a player is simply the load on the machine, and is thus independent of the job's length. Scheduling congestion games that do take the length into an account, were defined and studied in [18] (there, defined and studied as weighted congestion games with separable preferences) and [28]. The paper [11] provide bounds on the PoS for singleton congestion games, with weighted and unweighted players.

The SPoA and SPoS measures were introduced by [1], which study a similar game to ours only with congestion effects rather than cost-sharing, and with a different definition of the social optimum; namely the cost of the highest paying player (which is the *makespan* in their setting). The SPoA and SPoS were studied in [8] for (unweighted) network formation games in the cost-sharing setting. They show that SPoA is at most $O(\log k)$ (assuming that an SE exists).

The paper [26] studies the complexity of equilibria in a wide range of cost sharing games. Their results on singleton cost sharing games correspond to our model with unit-length jobs (and therefore also fair cost-sharing).

1.3 Our Results

In this paper we provide a complete analysis of the game with respect to equilibrium existence, computation, convergence, and quality. We study both unilateral and coordinated deviations, distinguishing between instances having unit and arbitrary machine-activation costs.

In Section 2 we analyze instances with unit-cost machines. The existence of a PNE in every CSSG is proved using a non-standard potential function. We also show that an optimal stable assignment exists, thus, PoS $= 1$. On the other hand, a schedule may be stable but inefficient, and the PoA is essentially $\min\{k, m\}$.

Section 3 considers the extended model of instances with arbitrary-cost machines. We prove that an CSSG is guaranteed to have a PNE if and only if the number m of machines is at most 2 or the number k of players is at most 3. Thus, for $m > 2$ and $k > 3$, a game with no PNE exists. We show that in general, the problem of deciding whether a given game instance has a PNE is NP-complete, and we present an efficient algorithm for computing a PNE for instances with machine-independent processing times. We then analyze the equilibrium inefficiency and show that even the best stable profile can be extremely expensive. Formally, we have PoS$= k$. This is in contrast to the known bound of $O(\log k)$ in classical cost-sharing games and 1 in games with unit-cost machines.

Section 4 studies coordinated deviations. We show that an SE is guaranteed to exist and can be found efficiently, only in instances with machine-independent processing times. We provide tight bounds for the SPoA and SPoS, that depends on both the number of players and (for unit-cost machines) the number of machines. The inefficiency of SE in CSSGs with arbitrary-cost machines follows from the analysis of unilateral deviations, and the fact that PoA $\geq$ SPoA $\geq$ SPoS $\geq$ PoS.

Our results are detailed in Table 1. The table lists our results where the inefficiency bounds refer to the objective of minimizing the total players' cost. That is, the cost of a profile P is $cost(P) = \sum_{1 \leq i \leq k} cost_i(P)$. An alternative objective, which is also common in the study of jobs scheduling games (e.g., [27, 9]) is to minimize the *maximal* cost of a player. In Section 5 we provide tight bounds for the equilibrium inefficiency with respect to the max-cost objective, and show that stability may be extremely inefficient with respect to this objective. Specifically, for CSSGs with unit-cost machines PoA $=$ SPoA $= k$ and SPoS $=$ PoS $= k/2$. For arbitrary-cost machines, even the PoS is k.

Activation costs	Processing times	Pure Nash Equilibrium			Strong Equilibrium		
		$\exists$	PoA	PoS	$\exists$	SPoA	SPoS
Unit	arbitrary	yes	$\min\{m, k\}$	1	no	$\min\{m, \frac{k}{2} + \frac{1}{2}\}$	$\min\{\frac{m}{2}, \frac{k}{4} + \frac{1}{2}\}$
	machine-indp.	yes	$\min\{m, k\}$	1	yes	$\min\{\frac{m}{2}, \frac{k}{4} + \frac{1}{2}\}$	$\min\{\frac{m}{2}, \frac{k}{4} + \frac{1}{2}\}$
Arbitrary	arbitrary	no†	k	k	no	k	k
	machine-indp.	yes	$k^{\ddagger}$	k	yes§	k	k

Table 1: Summary of our results. (†) Deciding whether a PNE exists is NP-complete. (‡) Adapted to our model from [2]. (§) Extension of [26].

It is interesting to compare our results with known results on equilibrium inefficiency on other cost-sharing games. For classical network formation games, it is known that the PoA=k and the PoS=$O(\log k)$ [2]. Thus, the fact that the PoA of CSSG equals the number of players could be expected. On the other hand, the significant distinction between instances with unit-cost and arbitrary-cost machines, is surprising, as it may seem that by adding dummy jobs that are limited to go only to a specific machine, one can 'simulate' arbitrary costs using a unit-cost instance. Therefore, we find this distinction interesting and unexpected. We note that the lower bound of PoS=k for CSSGs with arbitrary-cost machines is achieved with machines having almost-unit costs (see Theorem 3.6, describing an instance in which a single machine has unit-cost, and all other machines have cost $1 - \varepsilon$ for an arbitrarily small ε). The high PoS agrees with the known bound for weighted network formation games [2].

In [3], the author studies (among other questions) the efficiency of strong NE in weighted network formation games. It is shown that the SPoA is $1 + \log W$. These bounds are different from ours as we study the inefficiency with respect to the number of machines and jobs rather than the processing times. In fact, these bounds apply also to our setting and shed more light on the inefficiency picture. The upper bound follows since these models are more general. The lower bound in [3] follows since the network used to prove it can be described as a CSSG with machine-independent processing times. The bounds in [7] are less relevant to our work, as they refer to the players' weights, and to approximate-NE.

Weighted cost-sharing games with singleton strategies correspond to CSSGs with machine-independent processing-times, in which the input is given by *a complete* bipartite. The fact that CSSGs are played on restricted scheduling environment distinguishes between our work and previous work on weighted games. With machine-independent processing time, for every job i and machine j it holds that $p_{i,j} \in \{p_i, \infty\}$. Weighted games correspond to the case $p_{i,j} = p_i$. Note that most of our proofs for the PoS lower bounds involve restricted schedules, and are therefore not valid for weighted games.

Remark: While the focus here is on pure Nash equilibria, our results have direct implications on the price of anarchy and the price of stability with respect to mixed Nash equilibria. First, a mixed NE exists in any CSSG since the number of pure strategies of each player is finite. Also, it is easy to see that our upper bounds of PoA $\leq \min\{m, k\}$ (for unit-cost machines) and PoA $\leq k$ (for arbitrary activation costs) carry over to mixed strategies. In particular, the upper bound analysis in Theorem 2.2 and Theorem 3.5 is valid also when referring to mixed strategies. The matching lower bounds trivially carry over to mixed NE, since by definition, the worst mixed NE is at least as bad as the worst pure NE. Analyzing the inefficiency of mixed strong equilibrium is left as an open problem.

2 Instances with Unit-Cost Machines

In this section we study game instances in which all machines have the same activation cost, say $c(j) = 1$ for all $j \in \mathcal{M}$. We suggest a non-standard potential function to show that a CSSG with unit costs is a a generalized ordinal potential game. Hence, a PNE exists, and every sequence of improvement steps is guaranteed to converge to a PNE. We also provide tight bounds for the PoA and PoS. Let P be a profile of a CSSG with unit costs. Recall that with unit-cost machines, $cost(P)$ gives the number of active machines in P, that is, $cost(P) = |\{j \in \mathcal{M} | L_j > 0\}|$.

Theorem 2.1 *A CSSG with unit-cost machines is a generalized ordinal potential game.*

Proof: Let G be a CSSG with unit-cost machines. Let P be a profile of G. Consider the function

$$\Phi(P) = (cost(P), \Pi_{\{j \in \mathcal{M} | L_j > 0\}} L_j),$$

that maps a profile to a 2-dim vector. The first entry in the vector specifies the number of active machines in P; The second entry is the product of these machines' loads.

We show that Φ is a generalized ordinal potential function for the game. Specifically, we show that every migration of a job in best response dynamics reduces the lexicographic order of the potential. Consider a profile P and assume, w.l.o.g, that Player 1 migrates from machine u to machine w, and the resulting profile is P'. Denote by L_u, L'_u, L_w, and L'_w the loads on machines u and w before and after the deviation of Player 1, respectively, that is, $L_u = \sum_{\{i | s_i = u\}} p_{i,u}$ and $L'_u = L_u - p_{1,u}$, $L_w = \sum_{\{i | s_i = w\}} p_{i,w}$ and $L'_w = L_w + p_{1,w}$.

Clearly, the migration is beneficial only if $L_w > 0$. Thus, $cost(P') \leq cost(P)$. If $L'_u = 0$, then $cost(P') = cost(P) - 1$ and $\Phi(P) \succ \Phi(P')$. Otherwise, $cost(P') = cost(P)$. We show that the second entry in the potential vector strictly decreases by showing that $\Phi(P)_2 / \Phi(P')_2 > 1$.

Since the loads on machines other than u, w do not change, we have

$$\frac{\Phi(P)_2}{\Phi(P')_2} = \frac{L_u \cdot L_w}{L'_u \cdot L'_w}.$$

Note that the above fraction is well-defined as $L'_w > p_{1,w} > 0$ and $L'_u > 0$ since we analyze the case $cost(P') = cost(P)$.

Multiply both numerator and denominator by $p_{1,u}$ and $p_{1,w}$ and rearrange to get

$$\frac{\Phi(P)_2}{\Phi(P')_2} = \frac{L_u}{p_{1,u}} \cdot \frac{p_{1,w}}{L'_w} \cdot \frac{L_w}{p_{1,w}} \cdot \frac{p_{1,u}}{L'_u}.$$

Note that

$$\frac{L_u}{p_{1,u}} = \frac{1}{cost_1(P)} \quad \text{and} \quad \frac{p_{1,w}}{L'_w} = cost_1(P'). \tag{1}$$

Also,

$$\frac{1}{cost_1(P)} = \frac{L_u}{p_{1,u}} = \frac{L'_u}{p_{1,u}} + 1 \quad \text{and} \quad \frac{1}{cost_1(P')} = \frac{L'_w}{p_{1,w}} = \frac{L_w}{p_{1,w}} + 1.$$

Thus,

$$\frac{p_{1,u}}{L'_u} = \frac{cost_1(P)}{1 - cost_1(P)} \quad \text{and} \quad \frac{L_w}{p_{1,w}} = \frac{1 - cost_1(P')}{cost_1(P')}. \tag{2}$$

Combining (1) and (2), we have

$$\frac{\Phi(P)_2}{\Phi(P')_2} = \frac{cost_1(P')}{cost_1(P)} \cdot \frac{cost_1(P)}{1 - cost_1(P)} \cdot \frac{1 - cost_1(P')}{cost_1(P')} = \frac{1 - cost_1(P')}{1 - cost_1(P)}.$$

Since the migration is beneficial, $cost_1(P) > cost_1(P')$. Since both costs are positive and strictly lower than 1, we conclude that $\Phi(P)_2/\Phi(P')_2 > 1$. Thus, $\Phi(P) \succ \Phi(P')$, as required. ■

We turn to study the equilibrium inefficiency. Recall that our measurement for a profile P is the total players' cost, which is equal to the number of active machines.

Theorem 2.2 *Every CSSG instance with unit-cost machines has PoS $= 1$. If $k < m$, then PoA $= k$. If $m \leq k < 2m - 1$, then PoA $= m - 1$. If $k \geq 2m - 1$, then PoA $= m$.*

Proof: Consider a BRD sequence that starts from the social optimum profile (SO). By Theorem 2.1, the sequence reaches a PNE. Note that the maximal cost of a player in the SO is 1. Therefore, during the BRD process, when a player deviates, he will never activate a new machine (at cost 1). It follows that the number of active machines in the resulting PNE is at most the social optimum. Thus, PoS $= 1$.

We turn to analyze the PoA. Assume first that $k < m$. We describe a family of game instances for which PoA $= k$. Let $\mathcal{M} = \{0, 1, \ldots, k, k+1 \ldots, m-1\}$. For $1 \leq i \leq k$, the capable machines for Player i are $\{0, i\}$. Thus, machines $k+1, \ldots, m-1$ are dummy machines and not capable for any player. The social optimum is 1 and it is attained when all players are assigned to machine 0. The worst PNE is when for all $1 \leq i \leq k$, Player i is assigned to machine i. This is indeed a PNE since no player can reduce his payment by deviating to machine 0 as this machine is not used by any player in this profile and the machine costs are equal. Thus, PoA $= \frac{k}{1} = k$. Clearly, this bound is tight as PoA $\leq k$ trivially holds.

Assume next that $m \leq k < 2m - 1$. We prove that PoA $= m - 1$. The lower bound is very similar to the case in which $k < m$. For $1 \leq i \leq m - 1$, the capable machines for Player i are $\{0, i\}$, and for

$m \leq i \leq k$, the capable machines are $\{0, m-1\}$. The social optimum is 1 and it is attained when all players choose machine 0. The worst PNE is when all players use their "second" machine, and its cost is $m-1$. For the upper bound, assume by contradiction that there is a game instance with PoA $> m-1$. Since the SO uses at least one machine and any profile uses at most m machines, in an instance with PoA $> m-1$ it must be that $OPT = 1$ and the worst PNE costs m. Since the social optimum profile uses a single machine, that machine is capable for all players. We denote this machine by u. Let N be the profile of the worst PNE. Since its cost is m, every machine is used by at least one player. Since $k < 2m-1$, by the pigeonhole principle, there are at least two machines that are used by exactly one player. Let $v \neq u$ be a machine that is used only by Player i. Since all machines are used in N, there is at least one player that uses machine u. By the above, machine u is capable for Player i. Since he does not share the cost of machine v, by deviating to machine u, he reduces his payment. Thus, we reach a contradiction to the fact that N is a PNE.

Assume $k \geq 2m-1$. We show a family of instances in which the SO is 1 and the worst PNE uses m machines, and thus PoA $= m$. This is clearly a tight bound as the SO is at least 1 and any schedule uses at most m machines. We continue to describe the family. The only capable machines for Player 1 is machine 0. For $i = 2, 4, \ldots, 2m-2$, Players i and $i+1$ have $M_i = M_{i+1} = \{0, \frac{i}{2}\}$. For $2m-1 < i \leq k$, we have $M_i = \{0, m-1\}$. The processing times of the players on all the machines is equal. The SO is clearly 1 and it is achieved when all players choose machine 0. We claim that the profile in which all players (except for Player 1) choose their "second" machine is a PNE. Indeed, note that in this profile there are at least two players using machines $1, \ldots, m-1$ and the share of the machines' cost is divided equally. Since only Player 1 uses machine 0, a player cannot reduce his payment by deviating to that machine. ■

3 Instances with Arbitrary Cost Machines

In this section we extend the model and consider instances with arbitrary cost machines. As we show, a PNE may not exist even in very small instances. Moreover, it is NP-hard to decide whether a given instance has a PNE. On the other hand, a PNE is guaranteed to exist and can be calculated efficiently for instances with machine-independent processing times.

Theorem 3.1 *A PNE is guaranteed to exist in every CSSG in which $m \leq 2$ or $k \leq 3$. There is a CSSG with $m = 3$ and $k = 4$ with no PNE.*

Proof: We begin with the positive cases in which a PNE exists. Assume $m = 2$, and denote the two machines by u and v. We prove that a PNE exists. The proof is by induction on the number of players. For a single player, the PNE is simply an assignment of the job to a cheapest machine in M_1. Given a PNE for $k-1$ jobs, add the k-th job greedily to a machine minimizing his cost, say u. If the resulting assignment is not a PNE, then some jobs might migrate from v to u. Since the addition of Job k and the other migrating jobs is beneficial for the jobs on u no job migrates to v and a PNE is reached.

For $k \leq 3$, assign first Job 1 to the cheapest machine $u \in M_1$. Add Job 2 greedily to the machine, v, minimizing his cost, breaking ties in favor of u. If Job 2 joins Job 1, that is, if $u = v$, then advance to Job 3. Otherwise, if $c(v) < c(u)$, we conclude that $v \notin M_1$, and if $c(v) \geq c(u)$ and $v \in M_1$ we let Job 1 migrate to v if this is beneficial. We now have a stable schedule of Jobs $1, 2$ and we advance to Job 3. Job 3 is added greedily to the machine, w, minimizing his cost, breaking ties in favor of already open machines.

	m_1	m_2	m_3
a	2	$--$	13
b	1	1	$-$
c	$--$	2	1
d	2	$--$	$--$

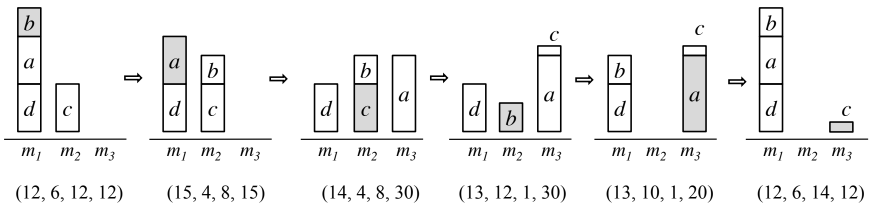

Figure 1: For $c(m_1) = 30, c(m_2) = 12$, and $c(m_3) = 14$, the instance has no PNE.

Assume first that Jobs $1, 2$ are on different machines u, v. W.l.o.g., $c(u) \le c(v)$ and thus $u \notin M_2$. If Job 3 joins u, we get a PNE. If Job 3 joins v, then if $v \in M_1$ we let Job 1 migrate to v if this is beneficial. The result is a PNE. If Job 3 opens a new machine, w, such that $c(w) < c(u)$, then w is not feasible to $1, 2$ and we have a PNE. If $c(u) \le c(w) < c(v)$ then w is not feasible for Job 2. If $w \in M_1$ we let Job 1 migrate to w if this is beneficial. The result is a PNE. Finally, if $c(v) \le c(w)$ we let both Jobs $1, 2$ migrate to w if this is beneficial and possible. The result is a PNE.

Otherwise, Jobs $1, 2$ are on one machine u. If Job 3 joins u, we get a PNE. If Job 3 opens a new machine, w, such that $c(w) < c(u)$, then w might be feasible to one of $1, 2$ (not to both, since otherwise $u = w$). We let this job migrate if this is beneficial, and reach a PNE. If $c(u) \le c(w)$ we let both Jobs $1, 2$ migrate to w if this is beneficial and possible. The result is a PNE. Thus, for three players a PNE exists.

On the other hand, we show that there exists an instance with $m = 3$ machines and $k = 4$ players that has no PNE. Consider an instance, I_{noNE}, with three machines having activation costs $30, 12$ and 14, and four jobs having processing times as given in the table. Note that Job d must be assigned to m_1 and each of the other jobs has two feasible machines. Fig. 1 presents a loop of beneficial moves that covers six out of the eight possible configurations. The payment vector is given below each configuration. The job that has a beneficial move is darker and it deviates to the next configuration (the leftmost configuration follows the rightmost one). It is easy to see that the two other configurations (in which no machine accommodates two jobs from a, b, c) are not stable either. ∎

The next natural question is whether it is possible to decide efficiently whether a given instance has a PNE. We show that this is an NP-complete problem.

Theorem 3.2 *Deciding whether a game instance with arbitrary-cost machines has a PNE is NP-complete.*

Proof: Checking stability of a given profile can be done efficiently, therefore the problem is clearly in NP. We prove hardness by showing a reduction from the 3-dimensional matching problem (3DM), which is known to be NP-hard [14]. The input to the 3DM problem is a set of triplets $T \subseteq X_1 \times X_2 \times X_3$, where $|X_1| = |X_2| = |X_3| = n$. The number of triplets is $|T| \ge n$. The desired output is a 3-dim matching $T' \subseteq T$ such that $|T'| = n$ and any element in $X_1 \cup X_2 \cup X_3$ appears exactly once in T'.

Given an instance of 3DM, we construct a game G with $|T| + 9n$ machines and $12n$ jobs. The first $|T|$ machines, denoted triplet-machines, correspond to the 3DM triplets. The additional $9n$ machines form $3n$ copies of machines m_1, m_2, m_3 introduced in the instance I_{noNE} in the proof of Theorem 3.1. Each copy is associated with one element of $X_1 \cup X_2 \cup X_3$. Formally, the set of machines is $m_1^t, \ldots, m_{|T|}^t \cup \{m_1^{h,i}, m_2^{h,i}, m_3^{h,i} | \forall h \in \{1, 2, 3\}, 1 \le i \le n\}$. The cost of a triplet-machine $m_1^t, \ldots, m_{|T|}^t$ is 66. The cost of every $m_1^{h,i}$-machine is 30, the cost of every $m_2^{h,i}$-machine is 12, and the cost of every $m_3^{h,i}$-machine is 14.

We turn to define the jobs and their processing times. For each element in $X_1 \cup X_2 \cup X_3$, there are four jobs. The first job corresponds to the element itself, and three additional jobs are copies of jobs a, b

and c from I_{noNE}. Formally, the set of jobs is $X_1 \cup X_2 \cup X_3 \cup \{a^{h,i}, b^{h,i}, c^{h,i} | \forall h \in \{1,2,3\}, 1 \leq i \leq n\}$.

For all $h \in \{1,2,3\}$ and $1 \leq i \leq n$, the job $x_{h,i}$ can be processed on every triplet-machine to which the corresponding element belongs, or on machine $m_1^{h,i}$. Its processing time on both machines is 2. For all $h \in \{1,2,3\}$ and $1 \leq i \leq n$, the jobs $a^{h,i}$, $b^{h,i}$ and $c^{h,i}$ can be processed on $m_1^{h,i}, m_2^{h,i}$ and $m_3^{h,i}$ with allowed machines and processing times as described in the Table in Fig. 1.

We show that a matching of size n exists if and only if G has a PNE. The main idea is that a 3DM corresponds to a schedule in which the element-jobs are assigned in triplets to triplet-machines – each paying one third of a triplet-machine cost. On the other hand, if a 3DM does not exist, then every unmatched element pays at least half of a triplet-machine cost, and prefers migrating to a corresponding copy of m_1, generating an instance of I_{noNE} as a sub-instance that has no stable-assignment.

Formally, we claim that a 3DM matching exists if and only if G has a PNE schedule. Let T' be a matching of size n. The following is a PNE schedule: For every triplet $e = \{x_{1,i}, x_{2,j}, x_{3,k}\}$ in T', assign the three element-jobs on the triplet-machine m_e^t. In addition, assign Jobs $a^{h,i}$ and $c^{h,i}$ on $m_3^{h,i}$, and Job $b^{h,i}$ on $m_2^{h,i}$ for all $h \in \{1,2,3\}, 1 \leq i \leq n\}$. The above assignment is a PNE. Every element-job pays one third of the cost of a triplet-machina, that is, $66/3 = 22$. Migrating to a different triplet machine is not-beneficial, since all feasible triplet-machines are currently empty – by the matching constraint, T' includes only one triplet that includes $x_{h,i}$. It is easy to verify that a migration is not beneficial also for jobs of type $a^{h,i}, b^{h,i}$ and $c^{h,i}$.

Assume that a matching of the 3DM instance does not exist. This implies that in every schedule of the corresponding instance, more than n triplet-machines are active. Therefore, at least one triplet machine is assigned only one or two element-jobs. The cost for such jobs is 66 or 33, which is more than 30 – the activation cost of an m_1-machine. This implies that the element-job $x_{h,i}$ can benefit from migrating to $m_1^{h,i}$. However, once $x_{h,i}$ migrates to $m_1^{h,i}$, then together with the jobs $a^{h,i}, b^{h,i}$ and $c^{h,i}$, we get the instance I_{noNE} on $\{m_1^{h,i}, m_2^{h,i}, m_3^{h,i}\}$, where Job $x_{h,i}$ plays the role of Job d (its processing time on $m_1^{h,i}$ is 2). This sub-instance has no stable assignment. We conclude that if a matching does not exist, then every unmatched element induces a sub-instance that has no stable-assignment. ■

3.1 Machine-independent processing times

We show that when the processing times are machine independent, a PNE is guaranteed to exist, can be found efficiently, and BRD converges from any initial configuration. The BRD convergence proof builds on the proof for weighted symmetric cost-sharing games [2]. Our game is different since in the setting of [2], all machines are feasible to all jobs, that is, for all i, we have $M_i = \mathcal{M}$.

Theorem 3.3 *If the processing times are independent of the machines, then any application of BRD converges to a PNE.*

Proof: For each job i let p_i be such that $p_{i,j} = p_i$ for all $j \in M_i$. We tune the proof in [2]. Given a profile P, let $c_P(j) = c(j)/L_j(P)$ denote the marginal cost of M_j under P. Note that if $P(i) = j$ then i' payment is $p_i c_P(j)$. If $L_j(P) = 0$ then $c_P(j) = \infty$. Let $w(P)$ be the vector of marginal costs sorted in non-decreasing order, that is, $c_P(1) \leq c_P(2) \leq \ldots \leq c_P(m)$. We show that any beneficial move of a player in P results in a configuration for which the vector of marginal costs is strictly lexicographically smaller than $w(P)$. Assume that player i migrates from machine u to machine v. Clearly, $\{u, v\} \subseteq M_i$. For any $\ell \in M_i$, let P_ℓ denote the configuration resulting from P when i migrates to machine ℓ. To show that $w(P_v)$ is strictly lexicographically smaller than $w(P)$, it suffices to show that

$$\min_{j \in M_i} c_{P_v}(j) < \min_{j \in M_i} c_P(j). \tag{3}$$

Define $j' = \arg\min_{j \in M_i} c_P(j)$. Since M_v was i's best response, $c_{P_v}(b) \leq c_{P_j}(j)$ for all $j \in M_i$. In particular, $c_{P_v}(b) \leq c_{P_{j'}}(j')$. If $j' \neq u$, then $c_{P_{j'}}(j') < c_P(j')$ since a migration into a machine increases its load and thus decreases its marginal cost. Therefore, $c_{P_v}(v) < c_P(j')$ which proves inequality (3). If $j' = u$, then since the migration is beneficial for i, it holds that $c_{P_v}(v) < c_P(u) = c_P(j')$, which again proves inequality (3). ∎

We do not bound the number of BRD iterations performed till convergence to a PNE. However, we show that a PNE (in fact, a SE) can be calculated efficiently. The following algorithm computes a SE schedule for instances with machine-independent processing times, and arbitrary activation costs. It generalizes an algorithm from [26] for unit-length jobs (and thus, uniform cost-sharing).

Algorithm 1 - An algorithm for finding a SE for machine-independent processing times

```
1: for $t = 1$ to $m$ do
2:     For every machine $j$, let $A_t(j) = \sum_{\{i|j \in M_i\}} p_i$.
3:     Let $j_t = \arg\min_j c(j)/A_t(j)$.
4:     Assign all jobs $i$ for which $j_t \in M_i$ to machine $j_t$.
5:     Remove the machine $j_t$ and the jobs assigned to it from the instance.
6: end for
```

In words, in every iteration, the algorithm calculates for every machine j the total length, $A_t(j)$, of unassigned jobs it is capable to precess. It then selects the machine, j_t, that will achieve the minimal marginal cost per one unit of load if it is assigned all the jobs it can process. This marginal cost is given by $c(j)/A_t(j)$. Then, the algorithm assigns to machine j_t all these jobs and advance to the next iteration with fewer unassigned jobs.

Theorem 3.4 *Algorithm 1 outputs a SE.*

Proof: Assume that the resulting assignment is not a SE, and let Γ be a coalition. Let M_Γ be the set of machines involved in Γ's deviation. Let j_0 be the first machine in M_Γ to be activated in Algorithm 1. By the algorithm, j_0 is not feasible for any job assigned to later machines, so the coalition does not include a migration into j_0. Thus, some job $i \in \Gamma$ benefits by migrating from j_0 to some machine $j' \in M_\Gamma$. Assume that i was assigned to j_0 at iteration t_0, that is, $j_0 = j_{t_0}$. By the choice of j_0, $j' = j_{t'}$ for some $t' > t_0$. Let $S_{j'}$ denote the set of jobs in Γ for which $M_{j'}$ is feasible. By the algorithm, and the definition of j_0, these jobs were assigned in iterations $\{t_0, \ldots, t'\}$. Thus, the maximal possible load on j' after the deviation is $A_{t_0}(j')$. We show that the migration of i cannot be beneficial even if all the jobs in $S_{j'}$ migrate to $M_{j'}$ in Γ's deviation.

In the current assignment, the cost of job i is $c(j_0)\frac{p_i}{A_{t_0}(j_0)}$. By migrating, job i will join load at most $A_{t_0}(j')$ on machine j', thus, its cost would be at least $c(j')\frac{p_i}{A_{t_0}(j')}$. Since j_0 was selected at t_0 it holds that $c(j_0)/A_{t_0}(j_0) \leq c(j')/A_{t_0}(j')$. Thus, i's migration cannot be beneficial. ∎

Remark 3.1 *A different restricted class of instances assumes job-independent processing times. That is, for every machine j there exists a $p_j > 0$ be such that for all jobs for which $j \in M_i$, we have $p_{i,j} = p_j$. The resulting game corresponds to the model of uniform cost-sharing. As we show in Appendix A, the resulting game is a potential game.*

3.2 Equilibrium inefficiency

We show that stability might lead to an extremely inefficient outcome with respect to the total players' cost. Similar to classic congestion games, the PoA equals the number of players. On the other hand, the PoS might also be linear in the number of players (compared to O(log k) in classical cost-sharing games and 1 in games with unit-cost machines). Specifically,

Theorem 3.5 *The PoA of CSSGs equals the number of players.*

Proof: Let P be some PNE profile. First, PoA $\leq k$ since otherwise, some job pays more than OPT and migrating from P to its machine in the optimal profile is beneficial, contradicting the assumption that P is a PNE. This analysis is tight already for an instance with unit-length jobs and two machines, having costs k and 1, both are feasible to all jobs. The schedule in which all jobs are assigned to the expensive machine is a PNE. ∎

Theorem 3.6 *CSSGs with $m > 3$ machines and $k \leq m$ players have PoS $= k$.*

Proof: Since PoS $\leq$ PoA, Theorem 3.5 implies that PoS $\leq k$. For the lower bound, consider the following game in which the unique PNE has cost $k - \varepsilon'$ while the social optimum has cost 1. The jobs have lengths that are independent of the machine they are assigned to. The lengths are $1 \ll \ell_1 \ll \ldots \ll \ell_k$, where for $1 < i \leq k$, we have $\frac{\ell_i}{\sum_{1 \leq j \leq i} \ell_j} > 1 - \varepsilon$. Assume that a single machine, having cost 1 is feasible to all jobs. There are $k-1$ additional machines each having cost $1-\varepsilon$. Each of these machines is feasible to a single job among the $k-1$ longer jobs. The unique PNE is when each machine accommodates a single job. If two or more jobs are assigned together to the first machine, then the longer will escape to its dedicated machine. The PNE's cost is $k - \varepsilon'$, thus PoS $= k$, and we are done. ∎

For some special cases of CSSGs the PoS can be bounded as follows.

Theorem 3.7 *CSSGs with $m \in \{2, 3\}$ machines have PoS $= m$. CSSGs with k players and $m < k$ machines have PoS $= \Theta(k)$.*

Proof: Assume first that $m = 2$. Denote the two machines by u and v. If the social optimum costs $c(u) + c(v)$ then this is also the cost of any PNE and PoS $= 1$. If the SO assigns all the jobs on one machine, say u, then it is not a PNE only if some job pays more than $c(v)$. Since any job pays at most $c(u)$, we have that $c(v) < c(u)$ and therefore the cost of a PNE is at most $c(u) + c(v) < 2OPT$.

For $m = 3$, let w denote the third machine. If all three machines are activated in the SO, then this is also the cost of any PNE and PoS $= 1$. If the SO activates two machines, say u and v, and it is not a PNE then we can find a PNE under the assumption that machine w is not available at all, by Theorem 3.1, a PNE exists for $m = 2$. This assignment has cost= OPT. If it is not a PNE, then some job would migrate to machine w. Thus, $c(w) < OPT$, and since any assignment, and in particular any PNE, has cost at most $c(u) + c(v) + c(w)$, we get that for any PNE P, $cost(P) < 2OPT$. Finally, if the SO uses one machine and it is not a PNE then by the same argument the cost of any other machine is less than OPT and PoS < 3.

The above analysis is tight: an instance with $k = m$ players for which the unique PNE has cost $m - \varepsilon$ while the SO has cost 1, is described in Theorem 3.6.

We continue to the case in which $3 < m < k$. By Theorem 3.6, we have PoS $\leq k$, and in particular PoS $= O(k)$. For the lower bound, we describe a family of instances with PoS $= \Omega(k)$. Given k, we extend the no-NE example from the proof of Theorem 3.1. The activation costs of the machines M_a, M_b, and M_c remain $c(a) = 30, c(b) = 12$, and $c(c) = 14$, respectively. The activation cost of

M_d is $c(d) = (k-3) \cdot c(a) + 3 \cdot c(b)$. The first four jobs are identical to jobs $1, 2, 3$, and 4 in the example, only that they are allowed to go also to machine M_d. We add $(k-4)$ dummy jobs that are restricted to go only to M_a and M_d. The total processing time of all these jobs on machine M_a is ε. Thus, even when all the dummy jobs are assigned to machine M_a, there is a BRD cycle involving players $1, 2$, and 3, on machines M_a, M_b, and M_c, as in the original example. Let P be the profile in which all jobs are assigned to M_d. The processing times for Jobs $1, 2$, and 3 on M_d are such that $cost_1(P) = cost_2(P) = cost_3(P) = c(b)$. The processing times for Jobs $4, \ldots, k$ are such that for every such job i, we have $cost_i(P) = c(a)$. It is not hard to see that P is the only PNE in this game. Thus, PoS $= \frac{c(d)}{c(a)+c(b)} = 0.714 \cdot (k-3) + 0.857 = \Omega(k)$. By using $c(a) = 69$ and $\ell_4^1 = 7.7$, we slightly increase the bound to PoS $= 0.852 \cdot (k-3) + 0.148$. ■

4 Coordinated Deviations

Recall that a strong equilibrium (SE) is a configuration in which no *coalition* of players can deviate in a way that benefits all its members. We show that for machine-independent processing times, a SE is guaranteed to exist, and we present a poly-time algorithm to find one. We also prove that SPoS $=$ SPoA $= \frac{m}{2}$. On the other hand, we show that a SE may not exist when jobs have arbitrary processing times. In fact, even with unit-cost machines and if just a single job is allowed to have two variable processing times, there exists an instance, with $m = 3$ machines and $k = 5$ jobs that has no SE. The inefficiency of the general case decreases; we show that SPoS $= \frac{m}{2}$ and SPoA $= m$.

We start with the simpler class of machine-independent processing times. Recall that for every job i there is $p_i > 0$ such that $p_{i,j} = p_i$ for all $j \in M_i$. We show that any sequence of beneficial coordinated deviations converges to a SE. Moreover, a simple greedy algorithm for finding a SE exists – even for instances with arbitrary cost machines.

Theorem 4.1 *For any instance with unit-cost machines and machine-independent processing times, any sequence of beneficial coordinated deviations converges to a SE.*

Proof: For a profile P, let $w(P)$ be the vector of loads sorted in non-increasing order, that is, $L_1(P) \geq L_2(P) \geq \ldots \geq L_m(P)$. Let Γ be a coalition active at profile P. Let P' be the profile resulting from Γ's deviation. We show that $w(P')$ is strictly lexicographically larger than $w(P)$. Let M_Γ be the set of machines involved in Γ's deviation. Let $j_0 = \arg\max_{j \in M_\Gamma} L_j(P)$ and $j_0' = \arg\max_{j \in M_\Gamma} L_j(P')$. We show that $L_{j_0'}(P') > L_{j_0}(P)$. First, if a job i migrates out of m_{j_0} to some machine m_u, then, in order for the deviation to be beneficial for i, it must be that $p_i/L_{j_0}(P) > p_i/L_u(P)$. Since $L_{j_0'}(P') \geq L_u(P')$, we conclude $L_{j_0'}(P') \geq L_u(P') > L_{j_0}(P)$. Otherwise, no job migrates out of m_{j_0}. Being part of M_Γ, it must be that at least one job joins m_{j_0}. Therefore, $L_{j_0}(P') > L_{j_0}(P)$ and thus $L_{j_0'}(P') \geq L_{j_0}(P)$.

Given that every coordinated deviation results in strictly lexicographically larger load vector, we conclude that any sequence of beneficial coordinated deviations must converge. ■

We turn to study the inefficiency of a strong equilibrium. We show that even with machine-independent processing times, an optimal solution may be significantly better than any stable one. Thus, in systems where coordinated deviations are allowed, we may end-up with an extremely poor outcome.

Theorem 4.2 *CSSGs with unit-cost machines and machine-independent processing times have SPoS $=$ SPoA $= \min\{\frac{m}{2}, \frac{k}{4} + \frac{1}{2}\}$.*

Proof: We first show the bounds w.r.t. the number of machines. We show that SPoS $\geq \frac{m}{2}$ and SPoA $\leq \frac{m}{2}$. The statement will follow since SPoS $\leq$ SPoA. We start with the upper bound and show

SPoA $\leq \frac{m}{2}$. For any profile P it clearly holds that $cost(P) \leq m$. If the SO assigns the jobs on two or more machines, then SPoA $\leq m/2$ as required. Assume that the SO assigns all the jobs on a single machine m_0. We show that only one machine is active in any SE, implying that in this case, SPoA $= 1$. Consider any profile P with more than a single active machine. We claim that all the jobs assigned on $\mathcal{M} \setminus \{m_0\}$ form a coalition whose beneficial move is to join m_0. By the assumption, this is a valid migration. Let S_i be the set of jobs assigned in P to an active machine M_i. In P, their total cost is 1. After the deviation, their total cost is strictly less than 1 and since the relative cost of every job in S_i remains the same, all the coalition members benefit from the deviation. We conclude that if $SO = 1$ then any SE has cost 1, and if $SO \geq 2$ then the $\frac{m}{2}$-ratio clearly holds, thus, SPoA $\leq \frac{m}{2}$.

For the lower bound, we describe an instance with unit-cost machines achieving SPoS $= \frac{m}{2}$. An example for $m = 5$ is given in Fig. 2. Given m, there are $n = 2(m-1)$ jobs consisting of $m-1$ pairs, $a_1, b_1, \ldots, a_{m-1}, b_{m-1}$. Let $\mathcal{M} = \{m_0, \ldots, m_{m-1}\}$. For $1 \leq k \leq m-1$, the processing time of jobs a_k and b_k is 2^k. Job a_1 is restricted to machine m_0, Job b_1 is restricted to machine m_1. For $2 \leq k \leq m-1$, Job a_k is restricted to m_0 or m_k, and Job b_k is restricted to m_1 or m_k.

The SO assigns all the jobs $\{a_k\}$ on machine m_0, and all the jobs $\{b_k\}$ on machine m_1. This optimal profile is not a SE. Note that since $2^k > \sum_{i=1}^{k-1} 2^i$, each of a_{m-1} and b_{m-1} has cost more than $\frac{1}{2}$. This pair would benefit from migrating to machine m_{m-1}, where each will have cost exactly $\frac{1}{2}$. After this deviation, by the same argument, each of a_{m-2} and b_{m-2} has cost more than $\frac{1}{2}$. This pair would benefit from migrating to machine m_{m-2}. Next, in turn, every pair will deviate to a new machine, resulting in the only SE of this instance in which Job a_1 is alone on m_0, Job b_1 is alone on m_1, and for every $2 \leq k \leq m-1$, the pair of jobs a_k and b_k is on m_k. There are m active machines in this SE, while only two machines are active in the SO.

We proceed to prove the bounds w.r.t. the number of players. First, we show that SPoA $\leq \frac{k}{4} + \frac{1}{2}$. We start with the following claim. Assume the SO uses x machines, where $x > 1$ as otherwise the analysis above shows SPoA$= 1$. We claim that in any SE, the number of machines that accommodate a single job is at most x. Otherwise, there is a SE P in which at least $x+1$ machines accommodate a single job each. Then, there are (at least) two jobs who share the same machine in the SO and use a machine by themselves in P. These two jobs can deviate to their machine in the SO and decrease their cost from 1 in P to less than 1, contradicting the fact that P is a SE. A corollary of the claim is that any SE costs at most $x + \frac{k-x}{2}$. Thus, SPoA $= \frac{x+(k-x)/2}{x} = 1 + \frac{k}{2x} - \frac{1}{2}$, which gets the maximal value of $\frac{k}{4} + \frac{1}{2}$ when $x = 2$. The lower bound is identical to the one described above: the SO costs 2 and the only SE costs $\frac{k-2}{2} + 2 = \frac{k}{2} + \frac{1}{2}$, thus SPoS $\geq \frac{k}{4} + \frac{1}{2}$.

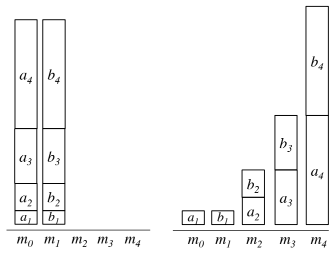
a_4
a_3
a_2
a_1
b_4
b_3
b_2
b_1
m_0 m_1 m_2 m_3 m_4
a_1 b_1 a_2 b_2 a_3 b_3 a_4 b_4
m_0 m_1 m_2 m_3 m_4

Figure 2: The social optimum (left) and the only SE (right) of a unit-cost instance achieving SPoS $= \frac{m}{2}$.

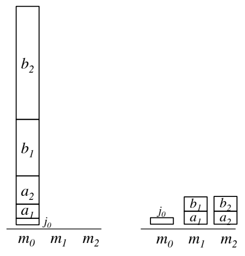

Figure 3: The social optimum (left) and the worst SE (right) of a unit-cost instance achieving SPoA $= m$.

■

While a SE is guaranteed to exist for any instance with machine-independent processing times, we

	m_1	m_2	m_3
a	2	$--$	2
b	6	4	$--$
c	$--$	4	4
d	2	$--$	$--$
e	$--$	$--$	3

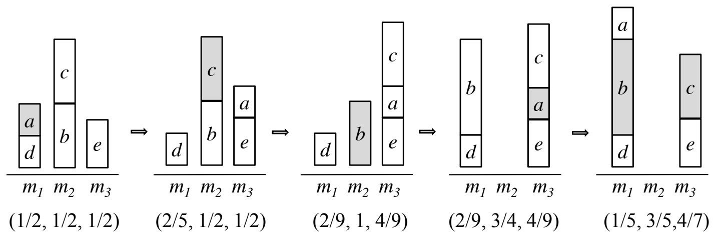

Figure 4: An instance that has no SE. The table on the left gives the processing times. The payment vector of jobs a, b and c is given below each configuration. The coalition that has a beneficial move is darker and its members deviate to the next configuration. The leftmost configuration follows the rightmost one, creating a loop. It is easy to see that the three other configurations, in which either b or c are alone on m_2, are not stable either.

show that even the slightest relaxation in this condition may result in an instance with no SE. Specifically, in Figure 4, we present an instance with unit-cost machines that has no SE. Note that all jobs except for a single one have machine-independent processing times.

Theorem 4.3 *There is an instance with $m = 3$ unit-cost machines and $k = 5$ jobs that has no SE.*

For instances with arbitrary processing times, a SE is not guaranteed to exist. For instances having a SE, the bounds on the equilibrium inefficiency depend on the processing environment: For instances with arbitrary activation costs, the analysis in Theorem 3.6 is valid also for coordinated deviations. Thus, SPoA $=$ SPoS $= k$. For instances with unit-cost machines, we prove the following.

Theorem 4.4 *CSSGs with unit-cost machines and arbitrary processing times have SPoS* $= \min\{\frac{m}{2}, \frac{k}{4} + \frac{1}{2}\}$ *and SPoA* $= \min\{m, \frac{k}{2} + \frac{1}{2}\}$.

Proof: We first show the SPoA lower bound w.r.t the number of machines. We describe an instance with unit-cost machines achieving SPoA $= m$. An example for $m = 3$ is given in Fig. 3. Given m, let $\mathcal{M} = \{m_0, ...m_{m-1}\}$. There are $n = 2m - 1$ jobs consisting on $m - 1$ pairs, $a_1, b_1, \dots, a_{m-1}, b_{m-1}$, and a job j_0, which is restricted to go to machine m_0. The processing time of j_0 on m_0 is $\varepsilon < 1$. For $1 \leq k \leq m - 1$, jobs a_k and b_k are restricted to go to m_0 or m_k. The processing time of either a_k or b_k on m_k is 1. The processing times of the $2(m - 1)$ jobs $\{a_k, b_k\}$ on m_0 are arbitrary *distinct* powers of 2.

The SO assigns all the jobs on m_0. It is easy to verify that the SO is a SE. Only one job (whose processing time is the highest power of 2) pays more than half. This job will not benefit from migrating by itself, and no job would join it and pay at least half after the deviation. However, the SO is not the only SE. Consider the profile P' in which j_0 is on m_0 and for $1 \leq k \leq m - 1$, jobs a_k and b_k are on m_k. The cost of P' is m, where j_0 has cost 1 and each of the other jobs has cost $\frac{1}{2}$. We claim that P' is a SE. The only possible deviation is into m_0. However, since the processing times on m_0 are distinct powers of 2, some job will cause more than half of the load on m_0, resulting in cost more than $1/2$. Since all the coalition members have cost $\frac{1}{2}$ in P', the deviation is not beneficial for this job. We conclude that a SE whose cost is m exists and SPoA $\geq m$.

We proceed to study the bounds w.r.t. the number of players. The SPoS lower bound follows from the example in Theorem 4.2 and the upper bound is also similar. For the SPoA, we show that SPoA $\leq \frac{k}{2} + 1$. Indeed, as in Theorem 4.2, if the SO uses one machine, then the number of machines in any SE with one job is at most one. Thus, any SE costs at most $1 + \frac{k-1}{2} = \frac{k+1}{2}$. On the other hand, if the SO uses more than one machine, then, trivially, the cost of a SE is at most k, and SPoA $\leq \frac{k}{2}$. Finally, for the lower bound, we adapt the example above by adding *dummy* machines. Then, the SO costs 1 and there is a SE that costs $\frac{k+1}{2}$. ■

5 Equilibrium Inefficiency with Respect to the Max-Cost Objective

Throughout the paper we considered the total players' cost as the cost of a profile P. That is, $cost(P) = \sum_{1 \leq i \leq k} cost_i(P)$. An alternative objective, which is also common in the study of job-scheduling games (e.g., [27, 9]) is to minimize the *maximal* cost of a player. Formally, for a given profile P, let $mcost(P) = \max_{1 \leq i \leq k} cost_i(P)$. In this section we provide tight bounds for the equilibrium inefficiency with respect to the max-cost objective. The inefficiency of a game G is defined as in Definition 1.1, with respect to the max-cost. Recall that $\Upsilon(G)$ is the set of pure Nash equilibria of the game G. Then PoA$(G) = \max_{P \in \Upsilon(G)} mcost(P)/OPT(G)$ and PoS$(G) = \min_{P \in \Upsilon(G)} mcost(P)/OPT(G)$.

The SPoA and the SPoS are defined similarly, where $\Upsilon(G)$ refers to the set of strong equilibria. As we show, stability may be extremely inefficient with respect to the max-cost objective.

Theorem 5.1 *For CSSGs with unit-cost machines PoA $=$ SPoA $= k$ and PoS $=$ SPoS $= \frac{k}{2}$ with respect to the max-cost objective.*

Proof: We start with the price of anarchy. Since PoA $\geq$ SPoA, it is sufficient to show that for every game PoA $\leq k$ and that there exists a game for which SPoA $= k$. The upper bound trivially holds. For the lower bound consider a game with $\mathcal{M} = \{m_0, m_1, m_2\}$. The capable machines for Player 1 are $\{m_0, m_1\}$. For $2 \leq i \leq k$, the capable machines for Player i are $\{m_0, m_2\}$. All the jobs have the same processing time on m_0, while for m_2, it holds that $p_{2,2} = k - 2$ and for $2 < i \leq k$, we have $p_{i,2} = 1$. Consider a profile P in which m_0 is empty, job 1 is assigned on m_1, and all other jobs are assigned on m_2. Note that Player 1 is alone on m_1 and Player 2 creates half of the load on m_2. Thus, $cost_1(P) = 1, cost_2(P) = \frac{1}{2}$, and for $2 < i \leq k$, we have $cost_i(P) = \frac{1}{2(k-2)}$.

We claim that P is a SE. Note that players $3, \ldots, k$ will not benefit from any coordinated deviation, as their cost on m_0 is always at least $\frac{1}{k}$. The only players who may benefit from migrating to m_0 are Players 1 and 2. However, their joint deviation is profitable only to Player 1. The SO assigns all the jobs on m_0. We get $mcost(P) = 1, mcost(\text{SO}) = \frac{1}{k}$, and SPoA$= k$.

We turn to analyze the price of stability. We show SPoS $=$ PoS $= \frac{k}{2}$. Since SPoS $\geq$ PoS, it is sufficient to show that for every game SPoS $\leq \frac{k}{2}$ and that there exists a game for which PoS $= \frac{k}{2}$. We start with the upper bound. If the SO assigns all the jobs on a single machine m_0, then it must be a SE: the total cost of every set of players is at most 1. By deviating to a new machine, their total cost does not reduce, it is therefore impossible that every member of the set strictly reduces his cost. Thus, if the SO uses a single machine, this profile is an SE and SPoS$= 1$. If the SO assigns the jobs on two or more machines, then the max-cost of a job in any profile and in particular the SO is at least $1/\frac{k}{2} = \frac{2}{k}$. On the other hand, the maximal possible cost of a job is 1, therefore SPoS $\leq \frac{k}{2}$.

We show that PoS $\geq \frac{k}{2}$. Consider an instance with two machines $\mathcal{M} = \{m_0, m_1\}$ and an even number of players. All jobs have the same length. There are $\frac{k}{2}$ jobs that must be assigned on m_0, a single job that must be assigned on m_1, and the other $\frac{k}{2} - 1$ jobs can be assigned on both machines. The SO is balanced, with $\frac{k}{2}$ jobs on each machine. Thus, $mcost(\text{SO}) = \frac{2}{k}$. In the unique NE profile, all the jobs except for one are on m_0. Thus, $mcost(\text{NE}) = 1$, and the PoS is $\frac{k}{2}$. ■

Theorem 5.2 *For CSSGs with arbitrary-cost machines PoS $=$ PoA $=$ SPoS $=$ SPoA $= k$ with respect to the max-cost objective.*

Proof: Since PoA $\geq$ SPoA $\geq$ SPoS $\geq$ PoS, it is sufficient to show that for every game PoA $\leq k$, and that there exists a game for which PoS $= k$. The upper bound trivially holds. For the lower bound consider an instance with two machines $\mathcal{M} = \{m_0, m_1\}$. Let $c(m_0) = 1 - \varepsilon$ and $c(m_1) = k$. All jobs have the same length. There are $k - 1$ jobs that can be assigned on both machines, and a single job must be assigned on m_1. The SO assigns all the jobs on m_1. The maximal cost of a job is 1. In the unique NE, there are $k - 1$ jobs on m_0, and a single job on m_1. This single job achieves cost k, and the bound follows. ■

References

[1] N. Andelman, M. Feldman, and Y. Mansour. Strong price of anarchy. *Games and Economic Behavior*, 65(2):289–317, 2009.

[2] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM J. Comput.*, 38(4):1602–1623, 2008.

[3] S. Albers. On the Value of Coordination in Network Design. *SIAM J. Comput.*, 38(6):2273–2302, 2009.

[4] R. Aumann. Acceptable points in games of perfect information. *Contributions to the Theory of Games*, 4:287–324, 1959.

[5] G. Avni, O. Kupferman, and T. Tamir. Network-formation games with regular objectives. In Proc. 17th FoSSaCS LNCS 8412, pages 119–133. Springer, 2014.

[6] B. Chen and S. Gürel. Efficiency analysis of load balancing games with and without activation costs, *Journal of Scheduling* 15(2):157–164, 2012.

[7] H. Chen and T. Roughgarden. Network design with weighted players. *Theory of Computing Syst.*, 45(2):302–324, 2009.

[8] A. Epstein, M. Feldman, and Y. Mansour. Strong equilibrium in cost sharing connection games. *Games and Economic Behavior*, 67(1):51–68, 2009.

[9] M. Feldman and T. Tamir, Conflicting Congestion Effects in Resource Allocation Games. *Journal of Operation Research*, 60(3):529-540, 2012.

[10] D. Fotakis, S. Kontogiannis and P.Spirakis. Selfish unsplittable flows. *Theoret. Comput. Sci.*, 348(23):226239. 2005.

[11] M. Gairing and F. Schoppmann, Total Latency in Singleton Congestion Games. In *Proc. 3rd WINE*, pages 381–387, 2007.

[12] V. Gkatzelis, K. Kollias, and T. Roughgarden. Optimal Cost-Sharing in General Resource Selection Games. Submitted to *Operations Research*, 2014.

[13] T. Harks and M. Klimm. On the existence of pure Nash equilibria in weighted congestion games. *Math. Oper. Res.* 37(3):419–436. 2012.

[14] R.M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[15] K. Kollias and T. Roughgarden. Restoring pure equilibria to weighted congestion games. In *Proc. 38th ICALP*, LNCS 6755, pages 539–551, 2011.

[16] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. *Computer Science Review*, 3(2):65–69, 2009.

[17] I. Milchtaich, Congestion games with player-specific payoff functions. *Games and Economic Behavior* 13(1):111-124, 1996.

[18] I. Milchtaich. Weighted congestion games with separable preferences. *Games and Economic Behavior*, 67(2):750–757, 2009.

[19] C. H. Papadimitriou. Algorithms, games, and the internet. In *Proc. 33rd STOC*, pages 749–753, 2001.

[20] M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. 2008.

[21] R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[22] R. W. Rosenthal, The network equilibrium problem in integers. *Networks* 3(1):53-59, 1973.

[23] T. Roughgarden and O. Schrijvers. Network Cost-Sharing without Anonymity. *ACM Transactions on Economics and Computation* vol.4(2), 2016.

[24] A. S. Schulz and N. E. Stier Moses On the performance of user equilibria in traffic networks. In *Proc. 14th SODA*, pages 86–87, 2003.

[25] L. S. Shapley. Additive and non-additive set functions. *Ph.D. thesis, Department of Mathematics*, Princeton University.

[26] V. Syrgkanis. The complexity of equilibria in cost sharing games. In *Proc. 6th WINE*, LNCS 6484, pages 366–377, 2010.

[27] B. Vöcking. *Algorithmic Game Theory*, Chapter 20: Selfish Load Balancing. Cambridge University Press, 2007.

[28] P. von Falkenhausen and T. Harks. Optimal cost sharing protocols for scheduling games. *Mathematics of Operations Research* 38(1):184–208, 2013.

[29] F. Xie, Z. Xu, Y. Zhang, and Q. Bai. Scheduling games on uniform machines with activation cost. *Theor. Comput. Sci.* 580:28–35, 2015.

[30] F. Xie, Y. Zhang, Q. Bai, and Z. Xu. Inefficiency analysis of the scheduling game on limited identical machines with activation costs *IPL* 116 (4):316–320, 2016.

A Additional cost-sharing schemes

In this work, we analyzed cost-sharing on unrelated machines assuming proportional cost-sharing scheme. That is, the cost of a machine splits among the jobs assigned to it proportional to their resource usage. This natural scheme is commonly used (e.g., [22, 17, 2, 12]) when the cost of a resource should split among its non-uniform users proportional to their demand.

As shown in Section 3, with proportional cost-sharing, a PNE may not exist even in very small instances. In this section we consider two additional cost-sharing rules, namely the (unweighted) Shapley value [25], and its generalization for weighted players suggested in [15].

For completeness, we define below the Shapley value and its weighted generalization, when applied to CSSGs. Given a profile $P = \langle s_1, s_2, \ldots, s_k \rangle \in (M_1 \times M_2 \times \ldots \times M_k)$ of a CSSG, consider a machine $j \in \mathcal{M}$. Let $A_j(P)$ denote the set of jobs assigned to machine j in P. The activation cost of the machine equals 0 if $A_j(P) = \emptyset$ and $c(j)$ otherwise. The Shapley value of job i is defined as the expected marginal increase in the joint cost caused by i, where the expectation is taken over all possible random orderings of the jobs in $A_j(P)$. In our model, this marginal increase is $c(j)$ if i is the first job assigned to machine j, and 0 otherwise. Since all the jobs have the same probability to be first in a random ordering, the Shapley-value cost-sharing rule induces *uniform cost-sharing*. Specifically, the cost of Player i in the profile P is $cost_i(P) = \frac{c(s_i)}{|A_j(P)|}$. Note that the cost of the profile remains the total activation-cost of non-idle machines. When the cost of a machine is shared evenly by the jobs assigned to it, a PNE always exists. Moreover, a PNE can be computed in polynomial time by the general algorithm for finding a PNE in fair cost-sharing games with singleton strategies [26]. Also, $\Phi(P) = \sum_{j \in \mathcal{M}} c(j) \cdot H(L_j(P)/p_j)$, where $H(0) = 0$, and $H(k) = 1 + \frac{1}{2} + \ldots + \frac{1}{k}$, is a potential function whose value reduces with every improving step of a player. We summarize in the following theorem.

Theorem A.1 *A PNE always exists and can be computed efficiently in every CSSG with unweighted Shapley-value cost-sharing rule.*

While guaranteeing stable solution, uniform cost sharing is considered unfair. It is reasonable to charge more from players that consume more resources. Weighted variants of the Shapley value use non-uniform distributions over orderings. In our model, since the first job assigned to a machine fully pays its activation cost, the whole cost-sharing mechanism depends on the probability that a job is first in a random ordering. The proportional cost-sharing scheme used in this paper corresponds to a distribution in which the probability of a job to be first in an ordering is proportional to the load it generates on the machine. Thus, $cost_i(P) = \frac{p_{i,s_i}}{L_{s_i}(P)} \cdot c(s_i)$.

As detailed in [12], changing the cost-sharing method being used leads to a different game, and hence to different equilibrium outcomes in the induced game. The goal is therefore to define distribution over the orderings that lead to more stable games with lower equilibrium inefficiency. Consider for example network cost-sharing games. With proportional cost-sharing, games with at least three players need not have a PNE [7]. The following variant of weighted Shapley values, suggested by Kollias and Roughgarden [15] (and therefore denoted *KR-Shapley value*), leads to a network cost-sharing game that has a potential function, and thus, always has a PNE. In the following, we adapt this variant to our game and analyze whether it leads to stable CSSGs.

Recall that $A_j(P)$ is the set of jobs assigned to machine j in a profile P. For every job $i \in A_j(P)$ let X_i be an exponentially distributed random variable with rate $\lambda_i = 1/p_{i,j}$. The weighted KR-Shapley share of i is the probability that X_i is the largest random variable among the random variables of jobs in $S_j(P)$. Formally, $cost_i(P) = c(j) \cdot Pr[X_i = \max_{k \in A_j(P)} X_k]$. Next, we show that unfortunately even small CSSGs with cost-sharing rule based on KR-Shapley values may not admit a PNE.

Theorem A.2 *With weighted KR-Shapley cost-sharing rule, there is a CSSG with $m = 3$ and $k = 4$ with no PNE.*

Proof: The game is the same game as the one depicted in Figure 1. We calculate below the payment vector associated with each configuration. We first note that for machines that are assigned two jobs, the KR-Shapley value boils down to proportional cost sharing. Specifically, assume that two jobs with processing times x and y are assigned on some machine. Then the KR-Shapley value of the first job is $\frac{\frac{1}{y}}{\frac{1}{x}+\frac{1}{y}} = \frac{x}{x+y}$, which is equal to its proportional share.

Therefore, the payment vector may differ from the one in Figure 1 only for profiles in which some machine is assigned three jobs. Consider the leftmost profile, in which $A_1(P) = \{a, b, d\}$. The three jobs a, b and d generate on m_1 loads 2, 1, and 2 respectively. The KR-Shapley value of job a is

$$\frac{\frac{1}{p_{2,1}}}{\frac{1}{p_{1,1}} + \frac{1}{p_{2,1}} + \frac{1}{p_{3,1}}} \cdot \frac{\frac{1}{p_{3,1}}}{\frac{1}{p_{1,1}} + \frac{1}{p_{2,1}}} + \frac{\frac{1}{p_{3,1}}}{\frac{1}{p_{1,1}} + \frac{1}{p_{2,1}} + \frac{1}{p_{3,1}}} \cdot \frac{\frac{1}{p_{2,1}}}{\frac{1}{p_{1,1}} + \frac{1}{p_{3,1}}} = \frac{5}{12}.$$

By symmetry, this is also the KR-Shapley value of job c, and therefore, the KR-Shapley value of job b is $1 - \frac{10}{12} = \frac{1}{6}$. Given that $c(a) = 30$, the resulting payment vector of the leftmost profile is

$(12.5, 5, 12, 12.5)$. By the same calculations, the payment vector of the rightmost profile is $(12.5, 5, 14, 12.5)$. The other payment vectors are identical to the ones in the figure, since every machine is assigned at most two jobs. Therefore, the cycle of beneficial steps described in Figure 1 is valid also with KR-Shapley cost-sharing rule. It is easy to verify that the two other configurations (in which no machine accommodates two jobs from a, b, c) are not stable either. ∎

An interesting open problem along the lines of [28, 15] is to suggest a fair cost-sharing mechanism, whose application induces a game that has a potential function.