# Cost-Sharing Scheduling Games on Restricted Unrelated Machines

Guy Avni[1] and Tami Tamir[2](✉)

[1] School of Computer Science and Engineering, The Hebrew University, Jerusalem, Israel
[2] School of Computer Science, The Interdisciplinary Center Herzliya, Herzliya, Israel
tami@idc.ac.il

**Abstract.** We study a very general cost-sharing scheduling game. An instance consists of $k$ jobs and $m$ machines and an arbitrary weighed bipartite graph denoting the job 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 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 SPoS and the SPoA.

## 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 operation 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 [12]. 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 act to minimize

M. Hoefer (Ed.): SAGT 2015, LNCS 9347, pp. 69–81, 2015.
DOI: 10.1007/978-3-662-48433-3_6

his own cost, rather than to optimize any global objective. Practically, this means that the jobs *choose* a machine instead of being assigned to one by a centralized scheduler. In this paper we study the corresponding cost-sharing scheduling game (CSSGs, 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 capable 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 arising 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 this most-general scheduling model is analyzed as a non-cooperative cost-sharing game.

## 1.1 Preliminaries

An instance of CSSG is given by an arbitrary weighted bipartite graph $G$ 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 $p_{j,i}$ 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 $p_{j,i}$ 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, $L_j(P) = \sum_{\{i|s_i=j\}} p_{j,i}$. When $P$ is clear from the context we omit it. The cost of Player $i$ in the profile $P$ is $cost_i(P) = \frac{p_{s_i,i}}{L_{s_i}(P)} \cdot c(s_i)$ and the cost of the profile $P$ is $cost(P) = \sum_{1 \leq i \leq k} cost_i(P)$. Note that $cost(P)$ also equals the total activation-cost of non-idle machines, that is, $cost(P) = \sum_{j \in \cup_i s_i} c(j)$.

Consider a game $G$. For a profile $P$, a job $i \in \mathcal{J}$, and a strategy $s'_i \in M_i$, let $P[i \leftarrow s'_i]$ 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 $cost_i(P[i \leftarrow s'_i]) \geq cost_i(P)$.

*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 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) [9,11] and *price of stability* (PoS) [2,14] measures. The PoA is the worst-case inefficiency of a Nash equilibrium, while the PoS measures the best-case inefficiency of a Nash equilibrium. Formally,

**Definition 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 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 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 move from $P$ [3]. 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 $p_{j,i} = p_i$ for all $j \in M_i$.

## 1.2 Related Work and Our Results

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* [13] 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, [5] closed the problem of PNE existence in these games by showing an example of a three-player game with no PNE.

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 [4]. 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) [4].

A lot of attention has been given to scheduling congestion games (for a survey, see [16]), 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 [10] (there, defined and studied as weighted congestion games with separable preferences) and [17].

The SPoA and SPoS measures where 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 where studied in [6] for network formation games in the cost-sharing setting.

In this work we complete the picture and study scheduling cost-sharing games; i.e., when jobs have an incentive to be assigned to a heavily loaded machine. CSSGs can be viewed as a generalization of classical cost-sharing games with weighted players [2]. The latter corresponds to the special case in which all the machines are identical; i.e., all machines are feasible to all jobs and the processing time of a job on a machine is independent of the machine.

The paper [15] 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).

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 instance having unit or arbitrary machine-activation costs. Our results are detailed in Table 1.

Due to space constraints, some proofs are omitted.

**Table 1.** Summary of our results. (†) Deciding whether a PNE exists is NP-complete. (‡) Adopted to our model from [2]. (§) Extension of [15].

| 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$^\dagger$ | $k$ | $k$ | no | $k$ | $k$ |
| | machine-indp. | yes | $k^\ddagger$ | $k$ | yes$^\S$ | $k$ | $k$ |

## 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 an CSSG with unit costs is a potential game. Hence, a PNE exists. We also provide tight bounds for the PoA and PoS. Let $P$ be a profile of an 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 1.** *A CSSG with unit-cost machines is a potential game.*

*Proof.* Let $G$ be an 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 $\Phi$ is a 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, $L_u=\sum_{\{i|s_i=u\}}p_{u,i}$ and $L'_u=L_u-p_{u,1}$, $L_w=\sum_{\{i|s_i=w\}}p_{w,i}$ and $L'_w=L_w+p_{w,1}$.

Clearly, the migration is be beneficial only if $L_w>0$. Thus, $cost(P')\le cost(P)$. 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 $L'_w > p_{w,1} > 0$ and $L'_u > 0$ since we analyze the case $cost(P') = cost(P)$.

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

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

Note that

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

Also,

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

Thus,

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

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.** *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 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.

The analysis for $m \leq k < 2m - 1$ is omitted.

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

*Proof.* The PNE-existence proof for $m \leq 2$ or $k \leq 3$ is omitted. 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. Figure 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 4.** *The question whether a game instance with arbitrary-cost machines has a PNE is NP-complete.*

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

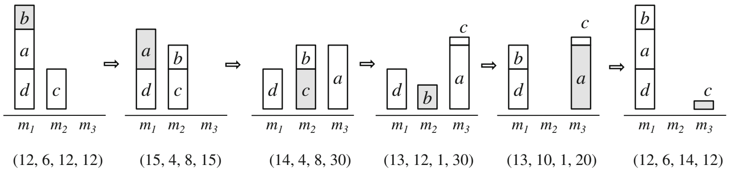

**Fig. 1.** For $c(m_1) = 30, c(m_2) = 12$, and $c(m_3) = 14$, the instance has no PNE.

*Proof (Sketch).* 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 [7]. 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 $|T| \geq n$. 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. Each copy is associated with one element of $X_1 \cup X_2 \cup X_3$.

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 $I_{noNE}$.

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. Thus, a 3DM matching exists if and only if $G$ has a PNE schedule. $\square$

*Remark 1.* An interesting open question along the lines of [8,17] is to suggest a different cost-sharing mechanism, whose application induces a potential game. It is not possible to adopt the approach suggested in [8] for weighted cost-sharing games, since in our game the Shapley values of the players are not well defined.

## 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 $M_i = \mathcal{M}$. An efficient algorithm for calculating a Strong NE for machine-independent processing times can be derived by generalizing the algorithm in [15] for fair cost sharing (unweighted jobs).

**Theorem 5.** *If the processing times are independent of the machines, then a PNE can be found efficiently and BRD converges to a PNE.*

*Remark 2.* 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 $p_{j,i} = p_j$. Since the cost of a machine is shared evenly by the jobs assigned to it, a PNE can be computed in polynomial time by the general algorithm for finding a PNE in fair cost-sharing games with singleton strategies [15]. Moreover, $\Phi(P) = \sum_{j \in \mathcal{M}} c(j) \cdot H(L_j(P)/p_j)$, where $H(0) = 0$, and $H(k) = 1 + 1/2 + \ldots + 1/k$, is a potential function whose value reduces with every improving step of a player.

### 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). Specifically,

**Theorem 6.** *The PoA of CSSGs equals the number of players.*

**Theorem 7.** *CSSGs with $m > 3$ machines and $k < m$ players have PoS $= k$.*

*Proof.* Since PoS $\leq$ PoA, Theorem 6 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 $1, \varepsilon, \varepsilon^2, \ldots, \varepsilon^{k-1}$, independent of the machine they are assigned to. Assume that a single machine, having cost 1 is feasible to all jobs. There are $k-1$ additional machines each having cost $\frac{1-\varepsilon}{1+\varepsilon}$. 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 PoA = PoS = $k$, and we are done. $\square$

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

**Theorem 8.** *CSSGs with $m \in \{2, 3\}$ machines have PoS $= m$. CSSGs with $k$ players and $m < k$ machines have PoS $= \Theta(k)$.*

## 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 $p_{j,i} = p_i$ 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 exist (omitted from this extended abstract) – even for instances with arbitrary cost machines.

**Theorem 9.** *For any instance with unit-cost machines and machine-independent processing times, any sequence of beneficial coordinated deviations converges to a SE.*

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 10.** *CSSGs with unit-costs 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 $\geq m/2$ and SPoA $\leq m/2$. The statement will follow since SPoS $\leq$ SPoA. We start with the upper bound and show SPoA $\leq m/2$. 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 $m/2$-ratio clearly holds, thus, SPoA $\leq m/2$.

For the lower bound, we describe an instance with unit-cost machines achieving SPoS $= m/2$. 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 an 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 1/2. This pair would benefit from migrating to machine $m_{m-1}$, where each will have cost exactly 1/2. After this deviation, by the same argument, each of $a_{m-2}$ and $b_{m-2}$ has cost more than 1/2.

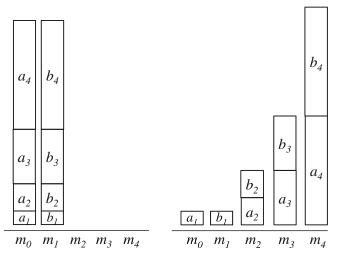
$a_4$
$b_4$
$a_3$
$b_3$
$a_2$
$b_2$
$a_1$
$b_1$
$m_0$ $m_1$ $m_2$ $m_3$ $m_4$
$a_1$ $b_1$ $a_2$ $b_2$ $a_3$ $b_3$ $a_4$ $b_4$
$m_0$ $m_1$ $m_2$ $m_3$ $m_4$

**Fig. 2.** The social optimum (left) and the only SE (right) of a unit-cost instance achieving SPoS $= m/2$.

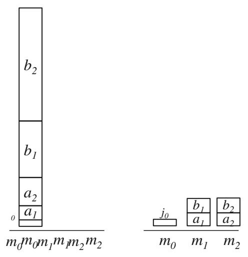

**Fig. 3.** The social optimum (left) and the worst SE (right) of a unit-cost instance achieving SPoA $= m$.

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 $\geq \frac{k}{4} + \frac{1}{2}$. □

While a SE is guaranteed to exist for any instance with machine-independent processing times, we show that even the slightest relaxation in this condition may result in an instance with no SE. Specifically, in Fig. 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 11.** *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 7 is valid also for coordinated deviations. Thus, SPoA $=$ SPoS $= k$. For instances with unit-cost machines, we prove the following.

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

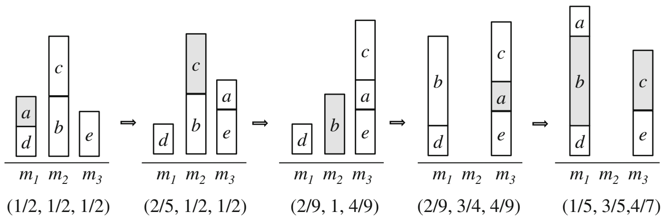

**Fig. 4.** An instance that has no SE. The table on the left gives the processing times. The payment vector of jobs $a$, $b$ and $c$ is given below each configuration. The coalition that has a beneficial move is darker and its members deviate to the next configuration. The leftmost configuration follows 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.

**Theorem 12.** *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 show the SPoA lower bound w.r.t the number of machines. The rest of the proof is omitted. 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, $a_1, b_1, \ldots, a_{m-1}, b_{m-1}$, 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 $1 \le k \le m - 1$, 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 $1 \le k \le m - 1$, 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 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 1/2 in $P'$, the deviation is not beneficial for this job. We conclude that a SE whose cost is $m$ exists and SPoA $\ge m$. $\square$

## References

1. Andelman, N., Feldman, M., Mansour, Y.: Strong price of anarchy. Games Econ. Behav. **65**(2), 289–317 (2009)
2. Anshelevich, E., Dasgupta, A., Kleinberg, J., Tardos, E., Wexler, T., Roughgarden, T.: The price of stability for network design with fair cost allocation. SIAM J. Comput. **38**(4), 1602–1623 (2008)

3. Aumann, R.: Acceptable points in games of perfect information. Contrib. Theor. Games **4**, 287–324 (1959)
4. Avni, G., Kupferman, O., Tamir, T.: Network-formation games with regular objectives. In: Muscholl, A. (ed.) FOSSACS 2014 (ETAPS). LNCS, vol. 8412, pp. 119–133. Springer, Heidelberg (2014)
5. Chen, H., Roughgarden, T.: Network design with weighted players. Theor. Comput. Syst. **45**(2), 302–324 (2009)
6. Epstein, A., Feldman, M., Mansour, Y.: Strong equilibrium in cost sharing connection games. Games Econ. Behav. **67**(1), 51–68 (2009)
7. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations, pp. 85–103. Springer, USA (1972)
8. Kollias, K., Roughgarden, T.: Restoring pure equilibria to weighted congestion games. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part II. LNCS, vol. 6756, pp. 539–551. Springer, Heidelberg (2011)
9. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. Comput. Sci. Rev. **3**(2), 65–69 (2009)
10. Milchtaich, I.: Weighted congestion games with separable preferences. Games Econ. Behav. **67**(2), 750–757 (2009)
11. Papadimitriou, C.H.: Algorithms, games, and the internet. In: Proceedings of the 33rd STOC, pp. 749–753 (2001)
12. Pinedo, M.L.: Scheduling: Theory, Algorithms, and Systems. Springer, New York (2008)
13. Rosenthal, R.W.: A class of games possessing pure-strategy Nash equilibria. Int. J. Game Theor. **2**, 65–67 (1973)
14. Schulz, A.S., Stier, N.E.: Moses On the performance of user equilibria in traffic networks. In: Proceedings of the 14th SODA, pp. 86–87 (2003)
15. Syrgkanis, V.: The complexity of equilibria in cost sharing games. In: Saberi, A. (ed.) WINE 2010. LNCS, vol. 6484, pp. 366–377. Springer, Heidelberg (2010)
16. Vöcking, B.: Algorithmic Game Theory, Chapter 20: Selfish Load Balancing. Cambridge University Press, New York (2007)
17. von Falkenhausen, P., Harks, T.: Optimal cost sharing protocols for scheduling games. In: Proceedings of the 12th EC, pp. 285–294 (2011)