Cost-sharing games in real-time scheduling systems

Tami Tamir

Accepted: 22 July 2022 / Published online: 2 December 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract
We apply non-cooperative game theory to analyze the server’s activation cost in real-time scheduling systems. An instance of the game consists of a single server and a set of unit-length jobs. Every job needs to be processed along a specified time interval, defined by the job’s release-time and due-date. Jobs may also have variable weights, which specify the amount of resource they require. We assume that jobs are controlled by selfish agents who act to minimize their own cost, rather than to optimize any global objective. The jobs processed in a specific time-slot cover the server’s activation cost in this slot, with the cost being shared proportionally to the jobs’ weights. Known results on cost-sharing games do not exploit the special interval-structure of the strategy space in our game, and are therefore not tight. We present a complete analysis of equilibrium existence, computation, and inefficiency in real-time scheduling cost-sharing games. Our tight analysis covers various classes of instances, and distinguishes between unilateral and coordinated deviations.

Keywords Cost-sharing games · Real-time scheduling · Equilibrium inefficiency · Equilibrium computation · Coordinated deviations

1 Introduction

The emergence of cloud systems as a common computation resource gives rise to plenty of optimization problems whose input is a real-time scheduling instance, consisting of time-sensitive jobs which are often business-critical. Each job needs to be processed along a specified time interval, defined by its release-time and due-date. Jobs may also have variable lengths and weights, corresponding to their resource demand (Baptiste 2000; Bar-Noy et al. 2001; Leung 2004; Irani and Pruhs 2005).

A preliminary version appears in the proc. of the 14th Conference on Web and Internet Economics (WINE), December 2018. This research is supported by The Israel Science Foundation (ISF). Grant No. 1036/17.

Tami Tamir
tami@idc.ac.il

1 School of Computer Science, Reichman University, Herzliya, Israel
Traditional research interest in cluster systems has been high performance, such as high throughput, low response time, or load balancing (Leung 2004; Baptiste 2000). In this paper we apply non-cooperative game theory to study the problem of minimizing the server’s activation cost, a recent trend in cluster computing which aims at reducing power consumption (see, e.g., Albers 2010; Chang et al. 2014; Khandekar et al. 2015).

The jobs should be processed by a server available during the whole schedule. We assume that time is slotted, and a job needs to be processed along one time-slot in order to be completed. For every time-slot, we are given the server’s cost for this slot. The server has unlimited capacity, and the cost is independent of the load (as long as it is non-zero). In other words, the cost is associated with activating the server.

As a real-life toy example, consider a huge hall presenting a laser-show in an amusement park. Laser-shows may start in every half an hour (defining the time slots), and every projection of the show costs a predefined amount, which is independent of the number of viewers. The park attracts many groups of visitors along the day. Every group would like to enter the hall and view the laser-show during its visit in the park. The owner would like to schedule the groups in a way that minimizes the total projection cost. The hall’s capacity is not a problem, as the show is projected on the ceiling of a huge hall (while this may not be true in crowded parks, this assumption fits other applications of our problem, such as broadcasting in media-on-demand systems and additional services with a low marginal per-client cost).

In this paper, we analyze the game corresponding to this job-scheduling scenario. We assume that jobs are controlled by selfish agents who act to minimize their own cost, rather than to optimize any global objective. Thus, each agent chooses the slot in which its job is processed instead of being assigned to one by a central authority. The selfish agents are aware of the cost-sharing mechanism, and the other agents around. Back to our laser-show example, in the corresponding game, every group selects its show time, with the understanding that groups viewing the show together share the show’s cost with the share being proportional to the groups’ sizes.

While game theory has become an essential tool in their study, many real-world applications do not necessarily fit the basic framework assumed in their common analysis. In particular, the setting of real-time scheduling induces a game in which the resources form a line and the strategy space of each player is defined by an interval in this line. Thus, our paper belongs to the rich literature on congestion and cost-sharing games with limited strategy space (e.g., singletons, matroids, paths in graphs, etc.). As we show, the strategies’ interval structure induces a game which is more stable than general singleton cost-sharing games. While some of our results are simple adaptations of previously studied games, most of them require different techniques and new tools, that exploit the unique interval-strategy structure. To the best of our knowledge, this is the first paper that studies real-time scheduling using non-cooperative game theory.
1.1 Preliminaries

An instance $G$ of real-time scheduling game consists of a set $\mathcal{J}$ of $n$ unit-length jobs, and a single server. Every job $j \in \mathcal{J}$ is associated with a time interval, $I(j) = [r_j, d_j]$, where $r_j$ and $d_j$ denote its release-time and due-date. In addition, every $j \in \mathcal{J}$ has a weight $w_j > 0$.

Let $T = \max_{j \in \mathcal{J}} d_j$ denote the maximal due-date of a job. We assume that the server is available along the interval $[0, T]$. Time is slotted, and a job can start its processing only at integral time points. For $t = 1, \ldots, T$ we refer to $[t-1, t)$ as the $t$-th slot, so the interval $[a, b)$ includes the $b - a$ slots $a + 1, \ldots, b$. Let $c_t$ denote the activation cost of the server in slot $t$. Note that the activation cost is load-independent, and for all $t$, the capacity of the server in slot $t$ is unlimited. When clear in context, we abuse notation and use $t$ both as an index in $\{1, 2, \ldots, T\}$, and as the unit-length interval $[t-1, t)$.

The system lacks a centralized scheduler. Every job corresponds to a selfish agent, which selects the time-slot in which it is processed. Specifically, the strategy space of job $j$ is the set of time-slots in $I(j)$. A profile of the game is a schedule $S = (s_1, \ldots, s_n)$, where $s_j$ is the time-slot in which job $j$ is processed, such that $[s_j - 1, s_j) \subseteq I(j)$. We say that the server is busy at time-slot $t$ if it processes at least one job in slot $t$. Otherwise, the server is idle at time $t$. For a profile $S$, the load on slot $t$ is denoted $\ell_t(S)$ and is given by the total weight of jobs processed in slot $t$. That is, $\ell_t(S) = \sum_{j \mid s_j = t} w_j$. All the jobs processed in slot $t$ share the server’s activation cost $c_t$ in a way proportional to the load they generate, given by their weights. Formally, the cost of job $j$ in a profile $S$ is $\text{cost}_j(S) = w_j \cdot c_{s_j} / \ell_{s_j}(S)$. This cost-sharing scheme fits the proportional cost-sharing rule for weighted players, where the weights correspond to the players’ weights. This rule is commonly used (e.g., Rosenthal 1973; Anshelevich et al. 2008; Gkatzelis et al. 2016; Avni and Tamir 2016) when the cost of a resource should split among its users proportional to their demand.

The social cost of a schedule is $\text{cost}(S) = \sum_{j \in \mathcal{J}} \text{cost}_j(S)$. Note that $\text{cost}(S)$ also equals the total activation cost of non-idle slots, that is, $\text{cost}(S) = \sum_{\ell_t(S) > 0} c_t$.

For a profile $S$, a job $j \in \mathcal{J}$, and a slot $s'_j \subseteq I(j)$, let $(S_{-j}, s'_j)$ denote the profile obtained from $S$ by replacing the strategy of job $j$ by $s'_j$. That is, the profile resulting from a migration of job $j$ from slot $s_j$ to slot $s'_j$. A profile $S$ is a pure Nash equilibrium (NE) if no job can benefit from unilaterally deviating from his strategy in $S$ to another strategy; i.e., for every job $j$ and every slot $s'_j \subseteq I(j)$ it holds that $\text{cost}_j(S_{-j}, s'_j) \geq \text{cost}_j(S)^1$.

Best-response dynamics (BRD) is a natural method by which players proceed toward a NE via the following local search method: as long as the strategy profile is not a NE, an arbitrary player is chosen and is offered to reduce his cost by deviating to his best strategy given the profile of all other players. The question of BRD

---

1 Throughout this paper, we consider pure strategies, as is the case for the vast literature on cost-sharing games. Unlike mixed strategies, pure strategies may not be random, or drawn from a distribution.
convergence and the quality of possible BRD outcomes are major issues in the study of resource allocation games in applied systems (Rosenthal 1973; Even-Dar et al. 2003).

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. For a game $G$, let $P(G)$ be the set of feasible profiles of $G$. We denote by $OPT(G)$ the cost of a social optimal (SO) solution; i.e., $OPT(G) = \min_{S \in P(G)} \text{cost}(S)$. We quantify the inefficiency incurred due to self-interested behavior according to the price of anarchy (PoA) (Koutsoupias and Papadimitriou 2009) and price of stability (PoS) (Anshelevich et al. 2008; Schulz and Stier Moses 2003) 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 $Y(G)$ be the set of pure Nash equilibria of the game $G$. Assume that $Y(G) \neq \emptyset$.

- The **price of anarchy** of $G$ is the ratio between the maximal cost of a NE and the social optimum of $G$. That is, $\text{PoA}(G) = \max_{S \in Y(G)} \text{cost}(S) / \text{OPT}(G)$. The **price of anarchy** of the family of games $\mathcal{G}$ is $\text{PoA}(\mathcal{G}) = \sup_{G \in \mathcal{G}} \text{PoA}(G)$.

- The **price of stability** of $G$ is the ratio between the minimal cost of a NE and the social optimum of $G$. That is, $\text{PoS}(G) = \min_{S \in Y(G)} \text{cost}(S) / \text{OPT}(G)$. The **price of stability** of the family of games $\mathcal{G}$ is $\text{PoS}(\mathcal{G}) = \sup_{G \in \mathcal{G}} \text{PoS}(G)$.

A firmer notion of stability requires that a profile is stable against coordinated deviations. A set of players $\Gamma \subseteq J$ forms a coalition if there exists a joint move where each job $j \in \Gamma$ strictly reduces its cost. When BRD is applied with coordinated deviations, then in every step some coalition performs a joint beneficial move. A profile $S$ is a Strong Equilibrium (SE) if there is no coalition $\Gamma \subseteq J$ that has a beneficial joint move from $S$ (Aumann 1959). The strong price of anarchy (SPoA) and the strong price of stability (SPoS) introduced in Andelman et al. (2009) are defined similarly, where $Y(G)$ refers to the set of strong equilibria.

**Applications:** We describe below two major applications of real-time scheduling. A detailed description and additional applications in optical-network design, and transportation services are given in Flammini et al. (2010) and Irani and Pruhs (2005). In all these applications, the users may be selfish agents who wish to minimize their own cost.

Power-aware scheduling is a major challenge in cloud computing. The power consumption of a machine is assumed to be proportional to the time the machine is in on state. While on, a machine can process several tasks simultaneously. The number of these tasks hardly affects the power consumption.

In media-on-demand services, clients are connected through a network to a set of servers which hold a large library of video programs. Each client can choose a program he wishes to view and the time he wishes to view it. Since the number of reads per a single media-stream is not limited, clients can share a single transmission. A real-time scheduling game is induced when clients specify the interval along which they wish to view the program.
1.2 Related work

This paper links two well-studied areas (1) cost-sharing games, and in particular cost-sharing games with singleton strategies, and (2) real-time scheduling, and in particular efficient energy allocation. Each of these areas has been widely studied. We survey below the papers we find most relevant to our work.

Game-theoretic analysis became an important tool for analyzing 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 (Rosenthal 1973; Koutsoupias and Papadimitriou 2009; Schulz and Stier Moses 2003; Caragiannis et al. 2011) consist of a set of resources and a set of players who need to use these resources. Players’ strategies are subsets of resources. In cost-sharing games, such as network formation games, each resource has an activation cost that is shared by the players using it according to some sharing mechanism. With unit-weight players and uniform cost-sharing, this is a potential game, a NE exists and the PoS is logarithmic in the number of players (Anshelevich et al. 2008).

Weighted cost-sharing games are cost-sharing games in which each player has a weight, and his contribution to the load of the resources he uses as well as his payments are multiplied by this weight. Such games need not have a pure NE and the PoS may be as high as the number of players (Anshelevich et al. 2008; Chen and Roughgarden 2009).

The work in Syrgkanis (2010) studies the complexity of equilibria in a wide range of cost sharing games. The results on singleton cost-sharing games correspond to our model with unit-weight jobs. Other related work studies the impact of the strategies’ combinatorial structure (Ieong et al. 2005; Ackermann et al. 2008; de Jong et al. 2016; Bilò and Vinci 2017; Bilò et al. 2020; Fiat et al. 2006). 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 (Avni et al. 2014). Job scheduling on unrelated machines is a special case of this class (Avni and Tamir 2016).

Variants of cost-sharing games have been the subject of extensive research. It is well-known that games with players’ specific costs (Milchtaich 1996) as well as other sharing variants need not have a NE. Another line of research studies the effect of different cost-sharing mechanisms on equilibrium inefficiency (Chen and Roughgarden 2009; Fotakis et al. 2005; Harks and Klimm 2012; Harks and Miller 2011). A lot of attention has been given to scheduling congestion games (e.g., Vocking 2007; Dumitrescu et al. 2010; Caragiannis et al. 2017; von Falkenhausen and Harks 2013; Cole et al. 2015), which can be thought of as a special case of weighted congestion games with singleton strategies. The work in Gairing and Schopmann (2007) provides bounds on the PoS for singleton congestion games, with weighted and unweighted players.

The above rich literature on congestion games does not consider the special structure of players’ strategies in the setting of real-time scheduling. Recently, our setting was studied in Georgoulaki et al. (2021), where some of our results were extended for load-dependent slot-activation costs and arbitrary cost-sharing. Specifically the activation cost of slot $t$ is a function $c_t(x) = x^d$, where $x$ is the load on $t$. The paper distinguishes between $d \leq 1$ (inducing positive congestion effect for the jobs) and
$d > 1$ (inducing negative congestion effect for the jobs), and between fair and arbitrary cost-sharing mechanisms. The authors show that the price of anarchy reduces when the payment mechanism allows arbitrary cost-sharing, that is, when the strategy of a player includes also its payment.

The SPoA and SPoS measures were introduced by Andelman et al. (2009), which study a scheduling game, with the goal of minimizing the cost of the highest paying player. The SPoA and SPoS were studied also for job scheduling on unrelated machines (Avni and Tamir 2016), and for network formation games (Epstein et al. 2009; Albers 2009).

There is wide literature also on real-time scheduling, either on a single or on parallel machines (see surveys in Chang et al. (2014), Irani and Pruhs (2005)). All previous work on real-time scheduling considers systems controlled by an external authority determining the jobs’ assignment. We are not aware of any results in which this setting is analyzed using game theoretic tools. When the server has a limited capacity, and jobs have variable-weights, many problems such as minimizing the number of late jobs, or minimizing the servers’ busy time are NP-hard, even with unit-length jobs (Chang et al. 2011; Albers 2010). On the other hand, with unit-weight unit-length jobs, these problems are polynomial solvable (Baptiste 2000; Chang et al. 2014). Results in Flammini et al. (2010); Khandekar et al. (2015) provide constant approximation algorithms for the minimum busy-time problem with variable-length, variable-weight jobs.

### 1.3 Our results

We provide a complete analysis of equilibrium existence, computation, and inefficiency in real-time scheduling cost-sharing games. Our analysis distinguishes between instances with unit slot-activation costs, in which $c_t = 1$ for all $1 \leq t \leq T$, and instances with unit job-weights, in which $w_j = 1$ for all $j \in J$. Specifically, we analyze the following four classes of games:

- $G_{1,1} = \{\text{games with unit slot-activation costs and unit job-weights}\}$.
- $G_{1,v} = \{\text{games with unit slot-activation costs and variable job-weights}\}$.
- $G_{v,1} = \{\text{games with variable slot-activation costs and unit job-weights}\}$.
- $G_{v,v} = \{\text{games with variable slot-activation costs and variable job-weights}\}$.

We first show that, independent of the instance class, any application of best-response dynamics, of unilateral or coordinated deviations, converges to a NE or a SE, respectively. Also, a SE can be computed efficiently. Moreover, $\text{PoS}(G_{1,v}) = 1$ and for this class we present an $O(n^2)$-time algorithm for computing, for any $G \in G_{1,v}$, a NE profile $S^*$ such that $\text{cost}(S^*) = \text{OPT}(G)$. This result heavily exploits the interval-structure of the players’ strategy space in real-time.
cost-sharing games, and is in contrast to other singleton cost-sharing games, in which computing an optimal stable solution is NP-hard, even with unit-demand players (Chekuri et al. 2007). The guaranteed existence of a SE is in contrast to other singleton cost-sharing games in which a SE may not exist (Avni and Tamir 2016). For general instances we present an $O(n^2 + T)$-time algorithm for computing a social optimum profile.

In Sect. 3, we consider instances with unit slot-activation costs. While in many singleton cost-sharing games, $\text{PoA} = n$ even with unit-weight players, unit-cost resources, and a restricted strategy space (Anshelevich et al. 2008; Avni and Tamir 2016), the PoA in our game is only $\Theta(\sqrt{n})$ with unit job-weights, and $n/2 + 1$ with variable job-weights, and its unique analysis relies on the interval-structure of the strategies.

In Sect. 4, we study instances with variable slot-activation costs. The bad news is that the limited strategy-structure does not help in reducing the PoA. That is, the PoA may be as high as the number of players, $n$, even if $\max_t c_t / \min_t c_t$ is arbitrarily close to 1. On the other hand, while in other singleton unweighted cost-sharing games $\text{PoS} = \Theta(\log n)$ (Anshelevich et al. 2008), we show that $\text{PoS}(G_{1,v})$ is a constant. Moreover, the PoS upper-bound proof is constructive, and when combined with our algorithm for computing a social optimum, we get a polynomial algorithm for computing, for every $G \in G^v_{1,1}$, a NE whose cost is at most $8/3 \cdot \text{OPT}(G)$.

Our results for the equilibrium inefficiency with respect to unilateral deviations are summarized in Table 1. All the bounds specified in the table are tight.

In Sect. 5, we study the equilibrium inefficiency with respect to coordinated deviations. By definition, for every game $G$, $\text{PoA}(G) \geq \text{SPoA}(G) \geq \text{SPoS}(G) \geq \text{PoS}(G)$. For instances with variable slot-activation costs, and variable job-weights our analysis in Sect. 4 implies that all four measures are as high as the number of jobs.

For instances with unit slot-activation costs, our analysis of coordinated deviations is more positive and a bit surprising—showing no difference between unit and variable job-weights, and no difference between the worst and best strong equilibrium. Specifically, we show that $\text{SPoA}(G_{1,v}) = \text{SPoS}(G_{1,1})$ and both measures are a constant—arbitrarily close to 2. Combined with our convergence proof, we conclude that natural dynamics such as best-response, even with coordinated deviations allowed, are guaranteed to converge to a solution whose cost is less than $2 \cdot \text{OPT}(G)$. This result distinguishes our game from other games in which

| Slot-activation costs | Job-weights | Pure nash equilibrium |
|-----------------------|-------------|-----------------------|
|                       |             | PoS       | PoA       |
| Unit                  | Unit        | $\sqrt{4n + 1} - 1$ |           |
|                       | Variable    | $n/2 + 1$   |           |
| Variable              | Unit        | $8/3$      | $n$       |
|                       | Variable    | $n$        | $n$       |

In Table 1, the bounds specified in the table are tight.
the strong price of anarchy was analyzed and shown to be either equal to the PoS which is \( O(\log n) \) (network formation games) or to 1 (single source connection games) (Epstein et al. 2009).

In general, our results show that games in which the players’ strategies have an interval structure, are more stable than general singleton cost-sharing games, the loss due to selfish behavior is smaller, and it is possible to compute efficiently a stable and optimal solution. We conclude in Sect. 6 with some open problems and directions for future work.

2 Equilibrium existence and computation

In this section we study the stability of real-time scheduling games. We first show that any application of best-response dynamics, with unilateral or coordinated deviations, converges to a NE or a SE, respectively. We then present a \( O(T + n^2) \)-time algorithm for calculating a strong equilibrium. Both results are valid for general instances—with variable slot-activation costs and variable job-weights.

2.1 Convergence of unilateral or coordinated beneficial deviations

**Theorem 2.1** For every \( G \in \mathcal{G}_n \), any application of BRD with unilateral or coordinated deviations, converges to a NE or a SE, respectively.

**Proof** Given a profile \( S \), let \( \ell_i(S) \) be the total weight of jobs assigned in slot \( t \), and let \( c_S(t) = c_i/\ell_i(S) \) denote the marginal cost of slot \( t \) in \( S \). If \( \ell_i(S) = 0 \) then \( c_S(t) = \infty \). Note that if job \( j \) is assigned in slot \( t \) then its cost is \( w_j \cdot c_S(t) \). Let \( \Gamma \) be a coalition in \( S \). Let \( S' \) be the profile resulting from \( \Gamma \)'s deviation. Let \( a(S) \) (resp. \( a(S') \)) be the vector of length \( T \), of marginal costs in \( S \) (resp. \( S' \)) sorted in non-decreasing order. We show that \( a(S') \) is lexicographically smaller than \( a(S) \).

Let \( M_\Gamma \) be the set of slots involved in \( \Gamma \)'s deviation, that is \( t \in M_\Gamma \) if and only if some job leaves \( t \) or some job joins \( t \) in the deviation. Clearly, if \( t \notin M_\Gamma \) then \( c_{S'}(t) = c_S(t) \). Let \( t_0 = \min_{t \in M_\Gamma} c_S(t) \). Let \( t_1 = \min_{t \in M_\Gamma} c_{S'}(t) \). We show that \( c_{S'}(t_1) < c_S(t_0) \).

There are two cases: (i) some job \( j \in \Gamma \) leaves \( t_0 \). In this case, since the deviation is beneficial for \( j \), it moves to a slot \( t' \) such that \( c_{S'}(t') < c_S(t_0) \). By definition, \( c_{S'}(t_1) \leq c_{S'}(t') \), implying \( c_{S'}(t_1) < c_S(t_0) \). (ii) no job leaves \( t_0 \) and some job joins \( t_0 \). In this case, the load on \( t_0 \) increases, thus, \( c_{S'}(t_0) < c_S(t_0) \). By definition, \( c_{S'}(t_1) \leq c_{S'}(t_0) \), implying \( c_{S'}(t_1) < c_S(t_0) \), as required. Since \( t_0 \) and \( t_1 \) are the slots with the minimal marginal costs among \( M_\Gamma \) before and after the deviation, and the marginal cost does not change for slots not in \( M_\Gamma \), we conclude that \( a(S') \) is lexicographically smaller than \( a(S) \).
Given that the vector of marginal costs is lexicographically decreasing with each deviation, BRD never loops, and must therefore converge to a strong equilibrium.

### 2.2 Computing a strong equilibrium

**Theorem 2.2** For every \( G \in \mathcal{G}_{v,v} \), a strong equilibrium can be computed efficiently.

**Proof** We present a simple greedy algorithm, that generalizes an algorithm from Syrgkanis (2010) for finding a NE in unweighted singleton games, for calculating a strong equilibrium.

**Algorithm 1** - An algorithm for finding a SE for a game \( G \in \mathcal{G}_{v,v} \)

1: All jobs are initially unassigned.
2: while there are unassigned jobs do
3: For every slot \( t \), let \( A(t) \) be the total weight of jobs \( j \) for which \( t \subseteq I_j \).
4: Let \( t^* = \arg\min t \in M \{ \frac{c_t}{A(t)} \} \).
5: Assign all jobs \( j \) for which \( t^* \subseteq I_j \) in slot \( t^* \).
6: Remove the assigned jobs from the instance.
7: end while

In words, in every iteration, the algorithm calculates, for every slot \( t \), the total weight, \( A(t) \), of unassigned jobs that are capable to be processed in it. It then selects a slot, \( t^* \), with a minimal potential cost per weight ratio, and assigns in it all its feasible jobs. It then advances to the next iteration with fewer unassigned jobs.

We show that Algorithm 1 outputs a strong equilibrium: Assume that the resulting schedule is not a SE, and let \( \Gamma \) be a coalition. Let \( M_{\Gamma} \) be the set of slots involved in \( \Gamma \)'s deviation. Let \( t_0 \) be the first slot in \( M_{\Gamma} \) to be activated in Algorithm 1. By the algorithm, \( t_0 \) is not feasible for any job assigned to later slots, so the coalition does not include a migration into \( t_0 \). Thus, some job \( j \in \Gamma \) benefits by migrating from \( t_0 \) to some slot \( t' \in M_{\Gamma} \). Let \( A_0(t_0) \) and \( A_0(t') \) denote the values of \( A(t_0) \) and \( A(t') \), respectively, in the iteration in which \( t^* = t_0 \). By the choice of \( t_0 \), \( c_{t_0}/A_0(t_0) \leq c_{t'}/A_0(t') \).

Let \( S_{t'} \) denote the set of jobs in \( \Gamma \) for which slot \( t' \) is feasible. By the algorithm and the definition of \( t_0 \), these jobs were assigned starting at the iteration in which \( t^* = t_0 \) till the iteration in which \( t^* = t' \). Thus, the maximal possible load on \( t' \) after the deviation is \( A_0(t') \).

We show that the migration of \( j \) cannot be beneficial even if all the jobs in \( S_{t'} \) migrate to slot \( t' \) in \( \Gamma \)'s deviation. In the current schedule, the cost of job \( j \) is \( \frac{w_j}{A_0(t_0)} \).
By migrating, job $j$ will join load at most $A_0(t') - w_j$ on slot $t'$, thus, its cost would be at least $\frac{w_j c_j}{A_0(t')}$, since $c_{t_0}/A_0(t_0) \leq c_{t'}/A_0(t')$, $j$’s migration is not beneficial, contradicting the assumption that $\Gamma$ is a coalition.

Time complexity: For simplicity, the algorithm is described above assuming every slot in $[0, T)$ is considered. In practice, every job induces two interesting time-points, its release time and due-date, and the whole interval $[0, T)$ is partitioned into at most $2n + 1$ sub-intervals, where the potential load is uniform along each interval. From each such sub-interval, the algorithm considers only the cheapest slot, which has the minimal potential cost per weight ratio. Thus, at most $2n + 1$ slots are considered by the algorithm. This implies that Algorithm 1 can be implemented in time $O(T + n^2)$. The time complexity is dominated by the calculation of $A(t)$ for the interesting slots. Note that each interesting slot may belong to the interval of $\Omega(n)$ jobs. The $O(T)$ component is inevitable if slots have variable activation costs. With unit slot-activation costs, the algorithm can be implemented in time $O(n^2)$ be considering an arbitrary slot from each sub-interval.

\section*{2.3 Computing an optimal NE assignment—unit activation costs}

We turn to consider the class $G_{1,v}$. We first show that for any $G \in G_{1,v}$, a NE assignment whose cost equals the social optimum exists, and then we present an algorithm whose time complexity is $O(n^2)$ for finding such an optimal stable assignment.

\textbf{Theorem 2.3} \textit{PoS} $(G_{1,v}) = 1$.

\textbf{Proof} While an arbitrary optimal solution is not necessarily stable, if BRD is performed starting from any social optimum profile, by Theorem 2.1, it is guaranteed to converge to a NE. Since moving to an empty slot is never beneficial for a single job, no new slot will be activated along the BRD process. \hfill \Box

\textbf{Theorem 2.4} For every $G \in G_{1,v}$, a NE whose cost is $\text{OPT}(G)$ can be computed efficiently.

\textbf{Proof} We present an optimal algorithm that computes a NE solution whose cost is $\text{OPT}(G)$. It consists of two phases: In the first phase, a social optimum solution, $S^*$, is computed. This solution is not necessarily a NE. In the second phase, the jobs are assigned in the busy slots of $S^*$, such that the resulting schedule is stable.

The proof of the algorithm combines two claims. The first claim shows that the number of slots activated during the first phase is minimal. The second claim refers to the stability of the schedule produced in the second phase. \hfill \Box
Algorithm 2 - An algorithm for finding a NE schedule such that \( \text{cost}(S) = \text{OPT}(G) \)

1: Sort the jobs such that \( d_1 \leq d_2 \leq \cdots \leq d_k \)
2: while there are unassigned jobs do
3: Let \( j \) be the next unassigned job. Activate slot \( d_j \) and assign every job \( k \) such that \( d_j \subseteq I_k \) in slot \( d_j \).
4: Remove the assigned jobs from the instance.
5: end while
6: Let \( b_1, \ldots, b_m \) be the set of slots in which the server is busy.
7: While there are unassigned jobs do
8: For every slot \( b_i \), let \( A(b_i) \) be the total weight of jobs \( j \) for which \( b_i \subseteq I_j \).
9: Let \( i^* = \arg \max_i A(b_i) \).
10: Assign all jobs \( j \) for which \( b_i^* \subseteq I_j \) in slot \( b_i^* \).
11: Remove the assigned jobs from the instance.
12: end while

Claim 2.5 For a game \( G \in \mathcal{G}_{1,v} \), let \( S_A(G) \) denote the assignment produced by Algorithm 2, then \( \text{cost}(S_A(G)) = \text{OPT}(G) \).

Proof Recall that \( b_1, \ldots, b_m \) is the set of slots in which the server is busy. By the algorithm, these slots are selected in increasing order. We show by induction on \( 1 \leq i \leq m \), that there exists an optimal solution in which the server is busy in slots \( b_1, \ldots, b_i \).

The base case is \( i = 1 \), corresponding to the slot \( b_1 \) activated in \( d_1 = \min_{j \in [d_j]} d_j \). Clearly, in any optimal solution, at least one slot is activated in \([0, d_1)\). Let \( b'_1 \leq d_1 \) be the first slot in which the server is busy in an arbitrary solution. Since \( d_1 \) is the minimal due-date of a job, for every job \( j \) processed in slot \( b'_1 \), it holds that \( d_1 \subseteq I_j \). Thus, it is possible to replace the given solution by a one in which \( d_1 \) is the earliest busy slot.

For the induction step, the greedy choice property shown above can be applied on the suffix of the instance, consisting of jobs that are not assigned in slots \( b_1, \ldots, b_{i-1} \).

Claim 2.6 The assignment \( S_A(G) \) produced by Algorithm 2 is a NE.

Proof Assume that \( S_A(G) \) is not a NE, and let \( j \) be a job that has a beneficial migration from slot \( b_u \) to slot \( t \). Since moving to an empty slot is never beneficial with unit activation costs, \( t = b_v \) for some \( 1 \leq v \leq m \). Moreover, it must be that \( b_v \) was selected in the while loop after \( b_u \), as otherwise, since both \( \{u, v\} \subseteq [r_j, d_j) \), job \( j \) would be assigned in \( b_v \). By the algorithm, \( b_u \) is not feasible for any job assigned to later slots. Let \( A_0(b_u), A_0(b_v) \) denote the values of \( A(b_u), A(b_v) \), respectively, in the iteration in which \( i^* = b_u \). By the choice of \( b_u \), \( A_0(b_u) \geq A_0(b_v) \). Since \( j \) is assigned in slot \( u \), the maximal possible load on \( b_v \) in \( S \) is \( A_0(b_v) - w_j \), implying that a migration into it cannot be beneficial for \( j \).
Time complexity analysis: The first phase can be implemented in linear time after
the jobs are sorted by due-dates and release-times, and it therefore takes \(O(n \log n)\).
The calculation of \(A(b_i)\) is the dominant component, and may take \(\Theta(n^2)\), since \(m\)
may be \(\Theta(n)\) and each \(b_i\) may belong to the interval of \(\Theta(n)\) jobs. Once the initial
values of \(A(b_i)\) are calculated, they are updated at most \(n^2\) times during the while loop.
Thus, the whole application of Algorithm 2 takes \(O(n^2)\) and is independent of \(T\). \(\square\)

Note that the resulting schedule does not produce a strong equilibrium. In Sect. 5,
we show that \(SPoS = 2\). Specifically, Algorithm 2 fails when coordinated deviations
are allowed, since moving to an idle slot may be beneficial for a coalition, but never
for a single job.

### 2.4 Computing a social optimum—variable activation costs

Finally, we consider the problem of computing a social optimum profile for
instances with variable activation costs. We present a \(O(T + n^2)\)-time algorithm
that computes (a not necessarily stable) optimal solution. In Sect. 4, we prove that
\(PoS(G_{v,1}) = \frac{8}{3}\) by providing an algorithm whose first phase consists of performing
BRD starting from a SO assignment. Our algorithm can be used in this phase, to get
a poly-time algorithm for calculating, for every \(G \in G_{v,1}\), a NE whose cost is at most
\(\frac{8}{3} \cdot \text{OPT}(G)\).

**Theorem 2.7** For every \(G \in G_{v,v}\), a profile whose cost is \(\text{OPT}(G)\) can be computed
efficiently.

**Proof** Let \(G \in G_{v,v}\). We assume that for every two jobs \(j_1, j_2\) it holds that if \(r_{j_1} < r_{j_2}\)
then \(d_{j_1} \leq d_{j_2}\). In other words, no interval is contained in another (such an instance is
commonly denoted *proper*). This assumption is w.l.o.g, since if \(I(j_2) \subseteq I(j_1)\), then \(j_1\)
can be removed from the instance, and be assigned in the slot of \(j_2\) once the assign-
ment is determined.

By the above, the jobs in \(j\) can be sorted such that \(r_1 \leq \cdots \leq r_n\) and
\(d_1 \leq \cdots \leq d_n\). Our algorithm is based on *dynamic programming*. For every \(j_1 \leq j_2\)
let

\[
\alpha(j_1, j_2) = \begin{cases} 
\min_{t \in \{r_{j_2}+1, \ldots, d_{j_1}\}} c_t & \text{if } r_{j_2} < d_{j_1} \\
\infty & \text{otherwise}
\end{cases}
\]

In words, if \(I(j_1)\) and \(I(j_2)\) overlap, then \(\alpha(j_1, j_2)\) is the cost of a cheapest slot in
\(I(j_1) \cap I(j_2)\). For a single job, \(\alpha(j, j)\) is the cost of a cheapest slot in \(I(j)\).

Computing a table with \(\alpha(j_1, j_2)\) for all \(1 \leq j_1 \leq j_2 \leq n\) can be computed as a pre-
processing in time \(O(T + n^2)\). After the table \(\alpha\) is computed, the dynamic program
advances by computing for every \(1 \leq j \leq n\) the minimal cost, \(C(j)\), of an assignment
of jobs \(1, \ldots, j\). The base case is \(C(0) = 0\). Then, for \(j = 1, \ldots, n\), let

\[
C(j) = \min_{k<j} C(k) + \alpha(k+1, j).
\]
That is, for every \( k < j \), we consider the cheapest assignment in which the rightmost busy slot processes the jobs \( \{ k+1, \ldots, j \} \), and select the cheapest among these candidates. In particular, \( C(n) \) is the social optimum.

Standard DP backtracking can be used to retrieve the busy slots (rather than their costs). Calculating \( C(j) \) takes \( O(j) \), for a total of \( O(n^2) \) for the whole table \( C \). Adding the calculation of \( a \), the total time complexity of our algorithm is \( O(T + n^2) \).

\[ \square \]

### 3 Instances with unit slot-activation costs

#### 3.1 Unit job-weights, unit slot-activation costs

This section discusses the equilibrium inefficiency of the class \( G_{1,1} \). Being a subclass of \( G_{1,v} \), Theorem 2.3 implies that \( \text{PoS}(G_{1,1}) = 1 \). We show that the interval-structure of the players’ strategies, limits the PoA to \( \Theta(\sqrt{n}) \).

**Theorem 3.1** \( \text{PoA}(G_{1,1}) = \sqrt{4n + 1} - 1 \).

**Proof** We begin with the lower bound. Let \( n = h^2 + h \) for some integer \( h \). We present a game \( G \in G_{1,1} \) for which \( OPT(G) = 1 \) and some NE profile has cost \( 2h = \sqrt{4n + 1} - 1 \). An example for \( n = 20 \) and \( h = 4 \) is presented in Fig. 1. The game is played over \( n \) unit-weight jobs. For \( i = 0 \ldots, h - 1 \), the set \( I_j \) includes \( h - i \) jobs for which \( I_j = [i, h + 1) \). For \( i = 1, \ldots, h \), the set \( I_j \) includes \( i \) jobs for which \( I_j = [h, h + 1 + i) \).

In our example, the interval of each of the 4 jobs assigned in slot 1 is \([0, 5)\). Symmetrically, the interval of each of the 4 jobs assigned in slot 9 is \([4, 9)\), and so on. Note that for all \( j \in A \) it holds that \([h, h + 1) \subseteq I(j)\), thus, an optimal solution assigns all the jobs in slot \( h + 1 \) (slot 5 in our example). A possible NE profile \( S \) assigns \( i \) jobs for \( i = 1, \ldots, h \), in each of the slots \( h - i + 1 \) and \( h + i + 1 \). The profile \( S \) is a NE, since jobs can only migrate to an interval with a lower load. Since the server is busy in \([0, h)\) and \([h + 1, 2h)\), \( cost(S) = 2h \) and the PoA bound follows.

![Fig. 1 A NE achieving PoA=\sqrt{4n + 1} - 1 = 8 for n = 20 unit-weight jobs. The jobs’ intervals are shown above the schedule](image)
For the upper bound, let $G$ be a game achieving maximal PoA in $G_{1,1}$. Let $S^*$ be a social optimum schedule of $G$ and assume that $\text{cost}(S^*) = m$. Let $b_1 < b_2 < \cdots < b_m$ be the sequence of slots in which the server is busy in $S^*$.

Let $S$ be a most expensive NE schedule of $G$. Partition the jobs into at most $2m$ sets $L_1, R_1, \ldots, L_m, R_m$, in the following way: For every job $j$, let $s_j^* \in \{b_1, \ldots, b_m\}$ be the slot in which job $j$ is processed in $S^*$, and let $s_j$ be the slot in which $j$ is processed in $S$. If $s_j \leq s_j^*$, then let $i$ be the minimal index such that $s_j \leq b_i$, and add $j$ to $L_i$. Symmetrically, if $s_j > s_j^*$, then let $i$ be the maximal index such that $s_j > b_i$, and add $j$ to $R_i$. The partition into sets implies that if $j \in L_i \cup R_i$ then job $j$ can be processed in slot $b_i$, that is, $b_i \subseteq I(j)$.

The following observations will be used in our analysis. The structure of $S$ is sketched in Fig. 2.

**Observation 3.2** In $S$, if two slots $t_1 < t_2$ both accommodate jobs from $L_i$, then $\ell_{t_1}(S) > \ell_{t_2}(S)$. If two slots $t_1 < t_2$ both accommodate jobs from $R_i$, then $\ell_{t_1}(S) < \ell_{t_2}(S)$.

**Proof** Assume by contradiction that in some NE, $S$, two jobs $\{j_1, j_2\} \subseteq L_i$ are processed in different slots, $t_1 < t_2$ such that $\ell_{t_1}(S) \leq \ell_{t_2}(S)$. Since $t_1 < t_2$, we have that $t_2$ is closer to $b_i$. Since $b_i \subseteq I_{j_1}$ and $t_1 < t_2 \leq b_i$, it must be that $t_2 \subseteq I_{j_1}$, thus, $j_1$ can migrate to $t_2$ and reduce its cost to $\frac{1}{\ell_{t_2}(S)}$, contradicting the stability of $S$. The analysis for $R_i$ is symmetric (note that the word ‘symmetric’ is accurate here).

**Observation 3.3** In $S$, for every $1 \leq i < m$, there is at most one slot in $[b_i, b_{i+1})$ in which jobs from both $R_i$ and $L_{i+1}$ are processed.

**Proof** Assume by contradiction that there are two different slots $t_1 < t_2$ in $[b_i, b_{i+1})$, in which jobs from both $R_i$ and $L_{i+1}$ are processed. The partition into sets implies that moving to the right, towards $b_{i+1}$, is feasible for $j \in L_{i+1}$, and moving to the left, towards $b_i$, is feasible for every $j \in R_i$. In particular, some job currently assigned in $t_2$ can migrate to $t_1$ and some job, currently assigned in $t_1$ can migrate to $t_2$. This implies that $S$ cannot be a NE—as a job from a least loaded slot among $t_1$ and $t_2$ can perform a beneficial move.
We conclude that $S$ has the following structure: during $[0, b_1)$, jobs from $L_1$ are processed in some slots with decreasing loads. During $[b_1, b_2)$, jobs from $R_1$ are processed in some slots with increasing loads, then a single slot may process jobs from $R_1 \cup L_2$, and then jobs from $L_2$ are processed in some slots with decreasing loads. This middle slot with the jobs from $R_1 \cup L_2$ has the maximal load. The same structure continues until, during $[b_m, T)$ jobs from $R_m$ are processed in some slots with decreasing loads.

**Claim 3.4** In $S$, w.l.o.g., for every $1 \leq i < m$, no slot processes jobs from both $R_i$ and $L_{i+1}$.

**Proof** Assume that for some $1 \leq i < m$, there exists a slot $t$ that processes jobs from both $R_i$ and $L_{i+1}$. Consider a game $G'$ produced from $G$ by replacing the interval of each of the jobs from $L_{i+1}$ processed at time $t$, by the interval $[b_i-1, t)$. We show that $S$ is a NE also for $G'$: it is given that at least one job, $j \in R_i$, is processed in slot $t$ in $S$. By definition of $R_i$, we have that $[b_i-1, t) \subseteq I(j)$. Since $S$ is a NE, $j$ has no beneficial migration, thus, the modified jobs whose interval is $[b_i-1, t)$ and are assigned in slot $t$ do not have a beneficial migration either. The stability of $S$ in the game $G$ implies that all other jobs are stable in $G'$. Note also that $S^*$ is a valid schedule for $G'$, as $b_i$ is feasible for the modified jobs. We conclude that $\text{PoA}(G') \geq \text{cost}(S)/m = \text{PoA}(G)$. Since $G$ was chosen as a game with maximal PoA in $G_{1,1}$, it must be that $\text{PoA}(G') = \text{PoA}(G)$.

**Observation 3.5** If $k$ jobs are assigned on $h$ slots with distinct loads then $h \leq \frac{1}{2}(\sqrt{8k + 1} - 1)$.

**Proof** The number of slots is maximized if the loads are $1, 2, \ldots, h$. Thus, in order to utilize $h$ slots, at least $\sum_{i=1}^h i = \frac{1}{2}(h^2 + h)$ jobs are required, implying $h \leq \frac{1}{2}(\sqrt{8k + 1} - 1)$.

Let $f(k) = \frac{1}{2}(\sqrt{8k + 1} - 1)$. By Observation 3.5, and the structure of $S$, at most $f(|L_1|)$ slots are busy in $[0, b_1)$, at most $f(|R_m|)$ slots are busy in $[b_m, T)$, and for every $1 \leq i < m$, at most $f(|R_i|) + f(|L_{i+1}|)$ slots are busy in $[b_i, b_{i+1})$.

We present the max-PoA problem as the following optimization problem. Since $OPT \leq n$, potentially, we have $2n$ regions in the schedule; thus, we have $2n$ variables representing the number of jobs in each of these regions, specifically, for $1 \leq i \leq n$, let $x_1^i = |L_i|$ and $x_2^i = |R_i|$. Additionally, we have $n$ binary variables $y_i$, for $1 \leq i \leq n$, indicating whether at least one of $L_i$ or $R_i$ is not-empty. Finally, the variable $m$ is the sum of these indicator variables.

Given that $OPT = \text{cost}(S^*) = m$, the PoA is at most $\frac{1}{m} \sum_{i=1}^m (f(|L_i|) + f(|R_i|))$. Thus, the objective function is to maximize the PoA. The constraints guarantee that all the jobs are scheduled in at most $m$ pairs of regions, that is, $OPT = m$. 
max \( \frac{1}{m} \sum_{i=1}^{n} (\sqrt{8x_1^i + 1} - 1 + \sqrt{8x_2^i + 1} - 1) \)
\[ s.t \sum_{i=1}^{n} (x_1^i + x_2^i) = n \]
\[ y_i \leq x_1^i \text{ and } x_1^i \leq ny_i \text{ for all } 1 \leq i \leq n \]
\[ y_i \leq x_2^i \text{ and } x_2^i \leq ny_i \text{ for all } 1 \leq i \leq n \]
\[ y_{i+1} \leq y_i \text{ for all } 1 \leq i \leq n - 1 \]
\[ \sum_{i=1}^{n} y_i = m \]
\[ x_1^i, x_2^i \in \mathbb{N} \text{ and } y_i \in \{0, 1\} \text{ for all } 1 \leq i \leq n \]

Since \( f(k) \) is concave and \( \sum_{i=1}^{m} (x_1^i + x_2^i) = n \), by Jensen’s inequality (Jensen 1906), the PoA gets its maximal value when \( m = 1 \), \( y_1 = 1 \) and \( x_1^i = x_2^i = n/2 \) (assuming \( n \) is even, which is the worst case). All other variables have value 0. Specifically, for every \( G \in \mathcal{G}_{1,1} \), we have \( \text{PoA}(G) \leq 2f(n/2) = 2 \cdot \frac{1}{2}(\sqrt{4n + 1} - 1) = \sqrt{4n + 1} - 1. \)

\( \square \)

### 3.2 Variable job-weights, unit slot-activation costs

We turn to analyze instances with variable job-weights. Here again, the PoA is lower than \( n \)—the PoA in general cost-sharing games with singleton strategies, however, it is still \( \Theta(n) \).

**Theorem 3.6** \( \text{PoA} (\mathcal{G}_{1,v}) = \frac{n}{2} + 1. \)

**Proof** We begin with the upper bound and show that \( \text{PoA}(G) \leq \frac{n}{2} + 1 \) for every \( G \in \mathcal{G}_{1,v} \). First note that if the social optimum assigns the jobs on two or more slots, then \( \text{PoA}(G) \leq n/2 \) follows from the fact that the maximal cost of a solution is \( n \). Assume that \( \text{OPT}(G) = 1 \), and let \( t \) be a slot such that \( t \not\subseteq I(j) \) for every \( j \in \mathcal{J} \).

Assume by contradiction that in some NE profile \( S \), the jobs are assigned on at least \( \frac{n}{2} + 2 \) slots. This implies that for at least three slots, a single job is assigned in each of these slots. Moreover, at least two of these three slots are either in \( [t-1, T) \), or in \( [0, t) \). Assume w.l.o.g., that two jobs \( j_1, j_2 \), are assigned alone on two different slots \( t_1 < t_2 \) in \( [0, t) \). Since slot \( t \) is feasible for both jobs, the job assigned on \( t_1 \) can join the job on \( t_2 \). This migration reduces its cost from \( 1 \) to \( w(j_1)/(w(j_1) + w(j_2)) \), contradicting the assumption that \( S \) is a NE.

We proceed to prove the lower bound. For every even integer \( n \), we describe a game \( G \in \mathcal{G}_{1,v} \) over \( n \) jobs, such that \( \text{PoA} (G) = \frac{n}{2} + 1 \). An example for \( n = 8 \) is given in Fig. 3. Let \( n = 2z \). The set of jobs consists of \( z \) pairs, \( a_1, b_1, \ldots, a_z, b_z \). Each of the four jobs \( a_1, b_1, a_2, b_2 \) has weight 1. For \( 3 \leq j \leq z \), \( w(a_j) = w(b_j) = 2^{j-3} \). The intervals of the jobs are \( I(a_1) = [0, 2) \) and \( I(b_1) = [1, 3) \). For \( 2 \leq j \leq z \), \( I(a_j) = I(b_j) = [1, j + 2) \). Note that for all jobs \( j \in \mathcal{J} \), \( [1, 2) \subseteq I(J) \). Thus, \( \text{OPT}(G) = 1 \)
is achieved by assigning all the jobs in the single slot \([1, 2)\). A possible NE leaves slot 2 idle and assigns \(a_1\) in slot 1, \(b_1\) in slot 3 and for \(2 \leq j \leq z\), jobs \(a_j\) and \(b_j\) are assigned in slot \(j + 2\). We show that \(S\) is a NE: the cost for each of \(a_1\) and \(b_1\) is 1, however, these jobs cannot join any other job, as they can only move towards slot 2 which is idle. The other jobs are paired with an equal-weight job, so each has cost \(1/2\). These jobs can move towards slot 2, but each of the busy slots they can move to has load not larger than their current pair’s load. Thus, no migration is beneficial, and \(S\) is a NE. The social cost of \(S\) is \(\frac{n}{2} + 1\), implying the lower bound of the PoA.

\[\Box\]

4 Instances with variable slot-activation costs

4.1 Unit job-weights, variable slot-activation costs

In this section we discuss the equilibrium inefficiency of the class \(G_{v,1}\). We show that unlike other singleton unweighted cost-sharing games, in which the PoS equals \(O(\log n)\) (Anshelevich et al. 2008), the interval strategy structure of real-time scheduling game guarantees that with unit-weight players, the PoS is \(O(1)\). Moreover, combined with the algorithm we present in Sect. 2.4, for calculating a social optimum, our proof provides a poly-time algorithm for finding, for every \(G \in G_{v,1}\), a NE profile whose cost is at most \(\frac{8}{3} \cdot \text{OPT}(G)\).

\[\text{Theorem 4.1} \quad \text{PoA} (G_{v,1}) = n.\]

\[\text{Proof} \quad \text{The upper bound of PoA} \leq n \text{ follows, as in many other cost-sharing games, from the fact that no player pays more than cost(SO) in a NE, as otherwise deviating to its strategy in the SO is beneficial. The lower bound is a simple adaptation of the instance ‘two parallel links of cost 1 and n’ in the PoA analysis of network formation games (Anshelevich et al. 2008). Specifically, assume that} \ c_1 = 1, c_2 = n, \text{ and for all } n \text{ jobs } J_i = [0, 2). \text{ The profile in which all jobs are processed in slot 2 is a NE whose cost is } n \text{ times the social optimum—in which all jobs are processed in slot 1.} \]

\[\Box\]
Theorem 4.2 \( \text{PoS} (G_{v,1}) \leq \frac{8}{3} \), and for every \( \varepsilon > 0 \), there exists a game \( G \in G_{v,1} \) for which \( \text{PoS} (G) = \frac{8}{3} - \varepsilon \).

Proof For an instance \( G \), let \( S^* \) be a social optimum profile of \( G \). Let \( b_1 < b_2 < \cdots < b_m \) be the sequence of slots in which the server is busy in \( S^* \). We show that there exists a NE profile, \( S_0 \), whose cost is at most \( \frac{8}{3} \cdot \sum_{i=1}^{m} c_{b_i} \). The profile \( S_0 \) is constructed in two phases. First, BRD is performed from \( S^* \), which, by Theorem 2.1, is guaranteed to converge to a NE profile \( \hat{S} \). In the second phase, \( \hat{S} \) is converted to a NE \( S_0 \) whose cost can be bounded.

Let \( \hat{S} \) be the NE to which BRD on \( S^* \) converges. Partition the jobs into at most \( 2m \) sets \( L_1, R_1, \ldots, L_m, R_m \), as in the proof of Theorem 3.1. Specifically, if \( \hat{s}_j \leq s^*_j \), then let \( i \) be the minimal index such that \( \hat{s}_j \leq b_i \), and add \( j \) to \( L_i \). Symmetrically, if \( \hat{s}_j > s^*_j \), then let \( i \) be the maximal index such that \( \hat{s}_j > b_i \), and add \( j \) to \( R_i \). The partition into sets implies that if \( j \in L_i \cup R_i \) then job \( j \) can be processed in slot \( b_i \), that is, \( b_i \not\subseteq I(j) \).

Next, note that Observation 3.3 is valid also with variable-cost instances, thus, at most one slot in \( [b_i, b_{i+1}) \) accommodates jobs from both \( R_i \) and \( L_{i+1} \).

Let us analyze the slots in \( \hat{S} \) accommodating jobs from \( L_i \). The analysis for \( R_i \) is symmetric. Let \( x_1^i > x_2^i > \cdots > x_k^i \) be the busy slots opened by jobs from \( L_i \) in \( \hat{S} \) (note that \( x_1^i \) is the closest to \( b_i \)). It is easy to see that \( c_{x_1^i} \geq c_{x_2^i} \geq \cdots \geq c_{x_k^i} \), that is, the costs are decreasing with the distance from \( b_i \). Indeed, during the BRD process, \( x_k^i \) is activated by a job \( j \) for which \( b_i \not\subseteq I(j) \), and therefore each of \( x_1^i, \ldots, x_{k-1}^i \) is also feasible for \( j \). Job \( j \)’s best-response is to activate slot \( x_k^i \) only if it is not more expensive than each of \( x_1^i, \ldots, x_{k-1}^i \).

The above analysis implies that the only slot that may process jobs from both \( R_i \) and \( L_{i+1} \) is the cheapest busy slot activated in \( [b_i, b_{i+1} - 1) \). Denote by \( \text{min}_i \) the cheapest slot activated in \( [b_i, b_{i+1} - 1) \). For simplicity, in order to handle the extreme intervals, \( [0, b_1 - 1) \) and \( [b_m, T) \), as a general case, we set \( \text{min}_0 = 0 \) and \( \text{min}_m = T + 1 \). In addition, w.l.o.g., we assume that there exists at least one busy slot in each of \( [0, b_1 - 1), [b_m, T) \), and \( [b_i, b_{i+1} - 1) \) for all \( 1 \leq i \leq m \). Thus, \( \text{min}_i \) is well defined. If this is not the case then some of the costs in the following analysis should be replaced by 0, which clearly makes the total cost lower.

We conclude that \( \hat{S} \) has the following structure (see Fig. 4): For all \( 1 \leq i \leq m \), jobs from \( L_i \cup R_i \) are processed in the interval \( [\text{min}_{i-1}, \text{min}_i) \). More specifically, during \( [0, b_1) \), jobs from \( L_1 \) are processed in some slots with non-decreasing costs. During \( [b_1, b_2) \), jobs from \( R_1 \) are processed in some slots with non-increasing costs, then a single slot, \( \text{min}_1 \), whose cost is minimal among the busy slots in...
may process jobs from \( R_1 \cup L_2 \), and then jobs from \( L_2 \) are processed in some slots with non-decreasing costs. The same structure continues until, during \([b_m, T)\), jobs from \( R_m \) are processed in some slots with non-increasing costs.

**Claim 4.3** For all \( 1 \leq i \leq m \), let \( z_i \) be the most expensive busy slot in \([\min_{i-1}, \min_i - 1]\) in \( \hat{S} \). It holds that (i) \( c_{z_i} \leq c_{b_i} \); (ii) the most expensive busy slot on one side of \( z_i \) in \([\min_{i-1}, \min_i - 1]\) has cost less than \( \frac{c_{b_i}}{2} \); and (iii) the most expensive busy slot on the other side of \( z_i \) in \([\min_{i-1}, \min_i - 1]\) has cost less than \( \frac{c_{b_i}}{3} \).

**Proof** The monotonicity of the busy slots’ costs implies that if \( b_i \) remains busy in \( \hat{S} \), then it must be the most expensive slot accommodating jobs from \( L_i \cup R_i \), that is, \( z_i = b_i \). If \( b_i \) is abandoned during the BRD, then the last job to escape from it is the only job who may benefit from leaving \( b_i \) and activating a slot whose cost is \( c > \frac{c_{b_i}}{2} \). Such a migration is beneficial only if \( c_{z_i} < c_{b_i} \). We turn to analyze the case that \( z_i \) is activated by a job \( j \) that migrates out of \( b_k \) for \( k \neq i \). If \( j \) is assigned on \( b_k, k < i \) then \( \min_{i-1} \) is feasible for \( j \), therefore, it is never beneficial for \( j \) to activate a slot in \([\min_{i-1}, b_i - 1]\). If \( j \) activates a slot \( t \in [b_i, \min_{i-1} - 1) \), then it must be that all the busy slots in \([b_i - 1, t - 1]\), and in particular \( b_j \), are feasible for it, thus, \( c_t \leq c_{b_j} \). Symmetrically, if \( j \) is assigned on \( b_k, k > i \) then \( \min_{i+1} \) is feasible for it, therefore, it is never beneficial for it to activate a slot in \([b_i, \min_i - 1]\). If it activates a slot \( t \in [\min_{i-1}, b_i - 1) \) then it must be that \( b_i \) is feasible for it, thus, \( c_t \leq c_{b_i} \).

Consider now the most expensive slots opened on the different sides of \( z_i \). The cost of a slot activated by a job that migrates out of \( b_i \) when the load on \( b_i \) is \( \ell_i \), is less than \( c_{b_i}/\ell_i \). Thus, only a job that escapes from \( b_i \) when its load is 2 can activate a slot whose cost is \( c > \frac{c_{b_i}}{3} \). The migration is beneficial only if \( c < \frac{c_{b_i}}{4} \). Lastly, only a job that escapes from \( b_i \) when its load is 3 can activate a slot whose cost is \( c > \frac{c_{b_i}}{4} \), and this migration is beneficial only if \( c < \frac{c_{b_i}}{3} \). If the jobs that escaped from \( b_i \) when its load was 2 and 3 migrate to the same side of \( b_i \) then the cost of the most expensive slot on the other side of \( b_i \) is less than \( \frac{c_{b_i}}{4} \). A similar analysis is valid for jobs originated from slots other than \( b_i \) in \( S^* \): since \( b_j \in I(j) \), then when the load on \( b_i \) is \( \ell_i \), it is beneficial for job \( j \) to activate a new slot only if its less than \( c_{b_i}/(\ell_i + 1) \). Moreover, it is not possible that a job originated from \( b_i \), for \( k_1 < i \), will activate a slot \( t_1 > b_i \), and a job originated from \( b_{k_2} \) for \( k_2 > i \), will activate a slot \( t_2 < b_i \), since both \( t_1 \) and \( t_2 \) are feasible for both jobs, implying that one of these jobs did not perform its best-response. We conclude that as long as \( b_i \) is not empty, jobs originated from other slots can only activate one slot of cost \( \frac{c_{b_i}}{3} < c < \frac{c_{b_i}}{2} \) on one side of \( b_i \).

Given \( \hat{S} \), we apply Algorithm 3 to convert it to a NE \( S_0 \) in which for all for \( 1 \leq i \leq m \), the ratio between the costs of any two adjacent busy slots in \([\min_{i-1}, b_i]\) and \([b_i, \min_i - 1]\) is at least 2. For each \( 1 \leq i \leq m \), the algorithm scans \( \hat{S} \) from left to right in \([\min_{i-1}, b_i]\), and from right to left in \([b_i, \min_i - 1]\), and closes the cheapest busy slots, such that in the remaining schedule the sequence of busy slots in each side of \( b_i \) fulfills the 2-ratio. The jobs that were assigned to slots that were closed are reassigned, closer to \( b_i \). After each iteration, BRD is performed, until no new slots are activated during the BRD. The idea is to group together jobs that
can share a slot. After this grouping, the load on the target slot, $t_1$, is more than 1, guarantying that no job will escape from $t_1$ to a slot whose cost is $c_{t_1}/2$ or more.

**Algorithm 3** - An algorithm for converting $\hat{S}$ into $S_0$

1. repeat
2. for $i = 1$ to $m$ do
3. if there is a pair of adjacent busy slots $t_2 < t_1$ in $[\text{min}_{i-1}, b_i]$, such that $c_{t_1} \leq 2c_{t_2}$ then
4. let $t_2 < t_1$ be such a pair closest to $b_i$.
5. migrate all the jobs processed in $[\text{min}_{i-1}, t_2)$ to slot $t_1$.
6. end if
7. if there is a pair of adjacent busy slots $t_1 < t_2$ in $[b_i, \text{min}_i - 1)$, such that $c_{t_1} \leq 2c_{t_2}$ then
8. let $t_1 < t_2$ be such a pair closest to $b_i$.
9. migrate all the jobs processed in $[t_2, \text{min}_i - 1)$ to slot $t_1$.
10. end if
11. end for
12. if the resulting assignment is not stable, perform BRD.
13. until no new slots are open in the BRD

Before analyzing the algorithm we demonstrate it on a simple example. Consider a game with 18 jobs scheduled along $[0, 10)$. In the social optimum, $S^*$, all the jobs are assigned to slot 6, whose cost is 1. Figure 5 presents $\hat{S}$, the NE resulting from BRD starting from $S^*$. In the social optimum, the cost of each job is $\frac{1}{18}$. For $j \in \{i, ii, iii\}$, $I(j) = [0, 6)$, and these three jobs are the first to leave slot 6 and migrate to slot 1 whose cost is $\frac{1}{18} - \epsilon$. Following their migration, the cost of each job assigned to slot 6 is $\frac{1}{18} - \epsilon$. For $j \in \{iv, v, vi\}$, $I(j) = [5, 10)$. These three jobs now benefit from leaving slot 6 and migrate to slot 10 whose cost is $\frac{1}{15} - \epsilon$. The BRD continues as described in the figure until $\hat{S}$ is reached.

While $\hat{S}$ is a NE, its cost is more than $8/3$. Specifically, it has adjacent busy slots whose costs are not decreasing by factor at least 2, (e.g., $\frac{1}{10} - \epsilon > \frac{1}{2} \cdot (\frac{1}{6} - \epsilon)$). We now apply Algorithm 3 on $\hat{S}$. In the left side of slot 6, the closest pair that do not obey the 2-ratio is $\frac{1}{12} - \epsilon$ and $\frac{1}{8} - \epsilon$, thus, the jobs on the slots of costs $\frac{1}{18} - \epsilon$ and $\frac{1}{12} - \epsilon$, are all migrated to the slot of cost $\frac{1}{8} - \epsilon$. Similarly, in the right side of the slot.
slot 6, the jobs on the slots of costs $\frac{1}{15} - \epsilon$ and $\frac{1}{10} - \epsilon$, are migrated to the slot of cost $\frac{1}{6} - \epsilon$. The resulting schedule is a NE that fulfills the 2-ratio.

**Claim 4.4** Algorithm 3 terminates with a NE $S_0$ fulfilling the following properties:

1. For all $1 \leq i \leq m$, the most expensive slot in $[\min_{i-1}, \min_i]$ has cost $c_{b_i}$. The two next most expensive slots in the different sides of $b_i$ have costs less than $\frac{c_{b_i}}{2}$ and $\frac{c_{b_i}}{3}$, respectively.
2. For all $1 \leq i \leq m$, for every pair of adjacent busy slots $t_1$ and $t_2$, in $[\min_{i-1}, b_i)$ or in $[b_i, \min_i - 1)$ it holds that $\max(c_{t_1}, c_{t_2}) \geq 2 \cdot \min(c_{t_1}, c_{t_2})$.

**Proof** First, $S_0$ is a NE since every reassignment of jobs is followed by BRD, which, by Theorem 2.1, is guaranteed to converge to a NE profile. Also note that jobs’ reassignments performed in Steps 5 and 9 are valid, as jobs are migrated to a slot closer to their slot in $S^*$.

We show that the properties are fulfilled: In the for loop, if a pair of slots contradicting Property (2) is detected, then jobs are migrated to the slot $t_i$ farthest from $b_i$ such that the 2-ratio is fulfilled between $t_i$ and $b_i$. If the resulting profile is not stable, then BRD is performed. During the BRD, new slots may be activated. However, if job $j$ migrates from slot $u$ to an idle slot $v$, then either slot $u$ is emptied, or the load on $u$ before $j$’s migration is at least 2, implying that $c_v < \frac{c_u}{2}$, otherwise the migration is not beneficial. Thus, the new busy slot obey the 2-ratio. Also note that for every $1 \leq i \leq m$, in every iteration, for each side of $b_i$, the range in which the 2-ratio is valid, is extended by at least one additional slot—which guarantees termination of the repeat loop.

Property (1) is fulfilled by combining Claim 4.3 with the fact that if a slot is activated during a BRD performed in Steps 5 or 10, it is always cheaper than an already busy slot in its side of $b_i$.

For every $1 \leq i \leq m$, let $C_0(b_i)$ denote the total cost of slots in $[\min_{i-1}, \min_i]$. Claim 4.4 implies that the most expensive slot in this interval has cost at most $c_{b_i}$ (in particular, it is possible that $b_i$ itself remains busy in $S_0$). In addition, the slots on one side of $b_i$ have total cost at most $c_{b_i} \cdot \sum_{k=1}^{n} 2^{-k} < c_{b_i}$, and the slots on the other side of $b_i$ have total cost at most $c_{b_i} \cdot \frac{1}{2} \sum_{k=1}^{n} 2^{-(k-1)} < \frac{2c_{b_i}}{3}$. Note that the slot $\min_i$ itself is not handled in Algorithm 3, however, it is the cheapest slot in the segment and its cost corresponds to the smallest number in a geometric sequence. We conclude that $\sum C_0(b_i) \leq (1 + \frac{1}{2} + \frac{2}{3})c_{b_i} = \frac{8}{3} \cdot c_{b_i}$.

Given that $S_0$ is a NE and that $\text{cost}(S_0) \leq \sum_{i=1}^{m} C_0(b_i) \leq \frac{8}{3} \sum_{i=1}^{m} c_{b_i} = \frac{8}{3} \cdot \text{cost}(S^*)$, we get the upper bound for the PoS.

We turn to prove the lower bound. Given $\delta > 0$, we present an instance for which $\text{PoS} \geq \frac{8}{3} - \delta$. Let $n \geq 3$ be an odd integer, and let $\epsilon$ be a small constant such that $\delta < \frac{1}{2^{n-1}/2} + \frac{1}{3 \cdot 2^{(n-3)/2} + (n-1)\epsilon}$. An example for $n = 11$ is given in Fig. 6. In $S^*$, all the jobs are assigned in slot $\frac{n+1}{2}$ whose cost is 1. For a single job, $I(j) = \{\frac{n-1}{2}, \frac{n+1}{2}\}$.  

© Springer
That is, this job must be assigned in the busy slot of \( S^\star \). For every \( 1 \leq i \leq n - 1 \), the interval of exactly one job is \([\frac{n-1}{2} - i, \frac{n-1}{2} + i]\). The activation cost of slot \([\frac{n+1}{2} - i]\) is \(2^{-i} - \epsilon\), and the activation cost of slot \([\frac{n+1}{2} + i]\) is \(\frac{1}{3} \cdot 2^{-(i-1)} - \epsilon\).

In the only NE every job is assigned in the cheapest slot in its interval—which is the slot farthest from the middle slot. It is easy to see that any other assignment is not a NE since if two or more jobs are assigned together, then moving farther from the middle slot is beneficial for at least one job, or, if the middle slot accommodates two jobs, then a third job will join them. The cost of this NE is \(1 + \sum_{i=1}^{(n+1)/2} (2^{-i} - \epsilon) + \sum_{i=1}^{(n+1)/2} (\frac{1}{3} \cdot 2^{-(i-1)} - \epsilon)\), which is, by the choice of \( n \) and \( \epsilon \), at least \(\frac{8}{3} - \delta\). \(\square\)

**Remark** Since the game is unweighted, by Ieong et al. (2005), the BRD performed during the algorithm converges in poly-time. The number of iterations is bounded by \( n \) since if two or more jobs are assigned together, then moving farther from the middle slot is beneficial for at least one job, or, if the middle slot accommodates two jobs, then a third job will join them. The cost of this NE is \(1 + \sum_{i=1}^{(n+1)/2} (2^{-i} - \epsilon) + \sum_{i=1}^{(n+1)/2} (\frac{1}{3} \cdot 2^{-(i-1)} - \epsilon)\), which is, by the choice of \( n \) and \( \epsilon \), at least \(\frac{8}{3} - \delta\).

**4.2 Variable job-weights, variable slot-activation costs**

We now turn to analyze the most general class, \( G_{v,v} \). Recall that \( c_t \) is the activation cost of slot \( t \). As we show, allowing variable slot-activation costs, may increase significantly the equilibrium inefficiency, even if \( \max_t c_t / \min_t c_t \) is arbitrarily close to 1.

**Theorem 4.5** PoA \((G_{v,v}) = n\) and for every \( \epsilon > 0 \), there exists a game \( G \in G_{v,v} \) for which PoS \((G) = n - \epsilon\).

**Proof** The upper bound of PoA \(\leq n\) follows, as in many other cost-sharing games, from the fact that no player pays more than OPT in a NE, as otherwise deviating to its strategy in the social optimum is beneficial. For the lower bound, we present a game \( G \) for which \( OPT(G) = 1 \) is achieved by assigning all the jobs on a single slot.
of cost 1, while in the only NE, every job is assigned on a different slot. The slots’ costs are decreasing, but arbitrarily close to 1. Specifically, given \( n \), let \( I_j = [0,j] \), for \( 1 \leq j \leq n \). The slot-activation costs are \( c_i = 1 - \epsilon_i \), where \( \epsilon_i = 0 \), and for \( t > 1 \), the value of \( \epsilon_i \) fulfills the inequality listed below. Jobs’ weight form a significantly increasing sequence. Formally, the weights and the costs are determined such that \( 1 = w_1 \ll w_2 \ll \cdots \ll w_n \), where for every pair \( 1 \leq k < j \leq n \), it holds that \( \sum_{i=k}^j w_i / u_t > 1 - \epsilon_j \). In other words, job \( j \) prefers being alone in slot \( j \), rather than sharing slot \( k \) with the set of jobs \( \{ k, \ldots, j \} \) (or, clearly, with a subset of these jobs). Since slot 1 is feasible for all jobs, \( OPT(G) = 1 \). The unique NE is when job \( j \) is processed in slot \( j \). If two or more jobs are assigned together on some slot, \( t_0 \), then the heaviest job, \( j \), will benefit from migrating to slot \( j > t_0 \) - which minimizes its cost, independent of the load on the other slots, and is not feasible to the other jobs assigned in \( t_0 \). The NE’s cost is \( n - \sum_j \epsilon_j \), thus \( PoS(G) \) is arbitrarily close to \( n \).

5 Coordinated deviations

In this section we study the equilibrium inefficiency with respect to coordinated deviations. By definition, for every game \( G \), \( PoA(G) \geq SPoA(G) \geq SPoS(G) \geq PoS(G) \). For general instances, we show the following bounds.

**Theorem 5.1** \( SPoA(G_{v,v}) < n \) and for every \( \epsilon > 0 \), there exists a game \( G \in G_{v,v} \) for which \( SPoS(G) \geq n - \epsilon \).

**Proof** The lower bound as well as the fact that \( SPoA(G_{v,v}) \leq n \) follow from Theorem 4.5. We show that \( SPoA(G_{v,v}) < n \). Assume by contradiction that for some \( G \in G_{v,v} \), and an SE profile \( S \) of \( G \), it holds that \( cost(S) = n \cdot OPT(G) \). Since \( S \) is stable against unilateral deviations, no player has cost more than \( OPT(G) \), thus, it must be that the cost of every player is exactly \( OPT(G) \). Consider a coordinated deviation in which all the players select their strategy in a social optimum profile, \( S^* \). Since their new total cost is \( OPT(G) \), and every player has a positive cost, it must be that each of the players benefit from the deviation, thus, \( S \) cannot be a SE.

For instances with unit slot-activation costs, we showed in Sect. 3 that \( PoS = 1 \) and the PoA is \( \Theta(n) \) or \( \Theta(\sqrt{n}) \) depending on the uniformity of job-weights. Our analysis of the SPoA and SPoS is therefore a bit surprising—showing no difference between unit- and variable-weight jobs, and no difference between the worst and best strong equilibrium. All measures turned out to be the same constant—arbitrarily close to 2.

**Theorem 5.2** \( SPoA(G_{1,v}) < 2 \), and for every \( \epsilon > 0 \), there exists a game \( G \in G_{1,1} \) for which \( SPoS(G) \geq 2 - \epsilon \).
Fig. 7 A beneficial coordinated deviation exists if three or more slots are busy in \([b_i + 1, b_{i+1})\)

**Proof** We start with the upper bound. For an instance \(G\), let \(S^*\) be an optimal schedule and let \(m = \text{cost}(S^*)\). Let \(b_1 < \cdots < b_m\) be the sequence of busy slots in \(S^*\). The following claim bounds, for every \(1 \leq i \leq m\), the number of busy slots in the interval \([0, b_i)\) in every strong equilibrium.

**Claim 5.3** In every strong equilibrium, for every \(1 \leq i \leq m\), (i) the server is busy in at most \(2i - 1\) slots during the interval \([0, b_i)\), and (ii) if the server is busy in exactly \(2i - 1\) slots during the interval \([0, b_i)\), then the rightmost busy slot in this interval processes only jobs \(j\) for which \(b_i \subseteq I(j)\).

**Proof** The proof is by induction on \(i\). The base case is \(i = 1\). Assume that in some SE the server is busy in more than a single slot during \([0, b_1)\). Let \(\mathcal{J}_0\) be the set of jobs processed during \([0, b_1)\). Since no slot in \([0, b_1 - 1)\) is busy in \(S^*\), for every \(j \in \mathcal{J}_0\), it holds that \(b_1 \in I_j\). Thus, the jobs of \(\mathcal{J}_0\) can form a coalition and deviate together to \(b_1\). After the deviation, they all share the slot-activation cost, therefore, this is clearly a beneficial move for every \(j \in \mathcal{J}_0\). Thus, exactly one slot is busy during \([0, b_1)\), and \(b_1\) is feasible for all the jobs assigned in it.

For the induction step, let \(S\) be a SE in which the server is busy in at most \(2i - 1\) slots during the interval \([0, b_i)\). Such a profile exists by the induction hypothesis. Let \(\mathcal{J}'\) denote the set of jobs processed in \(S\) during the interval \([b_i, b_{i+1})\). In \(S^*\), every \(j \in \mathcal{J}'\) is processed either in slot \(b_i\) or earlier, or in slot \(b_{i+1}\) or later. Partition \(\mathcal{J}'\) accordingly: Let \(J_- \subseteq \mathcal{J}'\) be the set of jobs that are processed in \(S^*\) not later than \(b_i\), and let \(J_- \subseteq \mathcal{J}'\) be the set of jobs that are processed in \(S^*\) not earlier than \(b_{i+1}\). The sets’ names indicate that jobs of \(J_-\) may migrate to the left, and jobs of \(J_-\) may migrate to the right.

Assume by contradiction that the server is busy in more than \(2i + 1\) slots during the interval \([0, b_{i+1})\). This implies that the server is busy in at least three slots during the interval \([b_i, b_{i+1})\). Consider the three rightmost busy slots in this interval, and let \(u, v\), be the two leftmost slots among these three (see Fig. 7a).

We distinguish between three cases: (a) jobs from both \(J_-\) and \(J_-\) are processed in both \(u\) and \(v\) (Fig. 7a). In this case, assume w.l.o.g that \(\ell'_-(S) \geq \ell'_+(S)\), then \(S\) is not a SE (and not even a NE), since jobs from \(J_-\) can benefit from migrating from slot \(u\) to slot \(v\). (b) No job from \(J_-,\) or no job from \(J_-\) is processed in slot \(v\) (Fig. 7b). In this case all the jobs processed in slot \(v\) can form a coalition. If they all belong to \(J_-,\) then joining the jobs in slot \(u\) is beneficial; if they all belong to \(J_-,\) then joining the third busy slot in \([b_i + 1, b_{i+1})\) is beneficial, contradicting the assumption that \(S\) is a
SE. (c) No job from $J_-$, or no job from $J_-$ is processed in slot $u$. In the first case, all the jobs in slot $u$ can form a coalition and join the jobs in slot $v$. If only jobs from $J_-$ are processed in slot $u$ (Fig. 7c), the analysis is more complicated. Consider the first busy slot left of $u$ in which the server is busy (marked by * in the figure). If there are exactly 3 busy slots during $[b_i, b_{i+1}]$, then there are exactly $2i - 1$ busy slots in $[0, b_i]$, and by part (ii) of the claim, all the jobs in slot * together with all the jobs in $u$ can migrate together to $b_i$ and reduce their cost (if $*= b_i$ then only the jobs in $u$ migrate). If there are more than 3 busy slots during $[b_i, b_{i+1}]$, then contradicting the stability is simpler—the jobs on $u$ can join this slot.

We conclude that if the server is busy more than $2i + 1$ slots during the interval $[0, b_{i+1}]$, then a beneficial coordinated deviation exists, and the assignment is not a SE.

We turn to prove part (ii) of the claim for $i > 1$. By the induction hypothesis there are at most $2i - 1$ busy slots during the interval $[0, b_i]$. Note that the analysis in the proof of part (i) provides a bound on the total number of busy slots, as well as a characterization of their distribution. Specifically, for $i > 1$, there are at most 2 busy slots in $[b_i, b_{i+1}]$.

Assume that the server is busy in exactly $2i + 1$ slots during the interval $[0, b_{i+1}]$. Thus, it must be that there are exactly two busy slots $u < v$ in $[b_i, b_{i+1}]$. If only jobs from $J_-$ are processed in slot $u$, then these jobs can either migrate to slot $v$ and benefit. If only jobs from $J_-$ are processed in slot $u$, then these jobs can either migrate to the left by themselves, or form a coalition with the jobs to their left (by the induction hypothesis, $b_i$ is feasible for them) and move together to $b_i$ to reduce their cost. Thus, it must be the case that some jobs from both $J_-$ and $J_-$ are processed in slot $u$. Assume by contradiction that for some job $j$ assigned in slot $v$, $b_{i+1} \notin I(j)$. It must be that $j \in J_-$. If all the jobs in slot $v$ are in $J_-$, then they can migrate together to slot $u$. If jobs from both $J_-$ and $J_-$ are processed in slot $v$, then $S$ is not a NE - using the same arguments as in case (a). Thus, for every job assigned in $v$ we have that $b_{i+1} \subseteq I(j)$.

In particular, for $i = m$, we get that the server is busy in at most $2m - 1$ slots during the interval $[0, b_m]$. If the server is busy during $[b_m, T]$, similar arguments imply that jobs processed in these slots can form a coalition and benefit from a migration to $b_m$ (by themselves or together with jobs processed before $b_m$). We conclude that in every SE, the server is busy in at most $2m - 1$ slots during $[0, T]$, implying $SPoA \leq \frac{2m-1}{m} < 2$.

**Lower Bound:** For simplicity, we first present an instance with variable job-weights and then show how to replace it by an instance with unit job-weights. Given $\varepsilon > 0$, let $T$ be an integer such that $T + 1$ is a multiple of 4 and $T \geq \frac{2-\varepsilon}{\varepsilon}$. We describe a game $G \in \mathcal{G}_{1, \nu}$ achieving $SPoS(G) \geq 2 - \varepsilon$. Specifically, $OPT(G) = \frac{T+1}{2}$, while the only SE assigns the jobs on $T$ slots, thus $SPoS(G) = \frac{2T}{T+1} = 2 - \frac{2}{T+1} \geq 2 - \varepsilon$. An example for $\varepsilon = \frac{1}{6}$, corresponding to $T = 11$ and $OPT = 6$ is given in Fig. 8.

Let $m = \frac{T+1}{2}$. The set of jobs consists of $m - 2$ triplets of jobs. Each triplet includes jobs of weights 1, 2 and 4 (to be denoted 1-, 2-, and 4-jobs). In the social
optimum, $S^*$, the $j$-th triplet is assigned in slot $2j + 1$. In addition, there are two pairs of jobs of weights 1 and 2 that are assigned in $S^*$ in slots 1 and $T$. These pairs are required to handle the leftmost and rightmost sides of the assignment. All the 1-jobs are restricted to their assignment in $S^*$, formally, for the 1-job in the $j$-th triplet, $I = [2j, 2j + 1]$. Similarly, one 1-job is restricted to $[0, 1)$ and one is restricted to $[T − 1, T)$. The other jobs have intervals of length 2. For odd $j$, the interval of the 2-job in triplet $j$ is $[2j − 1, 2j + 1]$ and the interval of the 4-job in triplet $j$ is $[2j, 2j + 2]$. In other words, starting from $S^*$, jobs of weight 2 can migrate to the right while jobs of weight 4 can migrate to the left. For even $j$, the job intervals are symmetric - that is, jobs of weight 2 can migrate to the left while jobs of weight 4 can migrate to the right. Finally, the interval of the 2-job in slot 1 is $[0, 2)$, and the interval of the 2-job in slot $T$ is $[T − 2, T)$.

We claim that the only SE is a one in which the server is busy along the whole interval $[0, T)$ interval. Note that for every integer $0 < j ≤ T/4$, slot $4j$ is feasible only for two 4-jobs, and for every integer $0 ≤ j ≤ T/4$, slot $4j + 2$ is feasible only for two 2-jobs. Moreover, pairs of 4-jobs minimize their cost (to $\frac{1}{2}$) if they are assigned together. Thus, in every SE, the 4-jobs are paired together. Given the assignment of the 4-jobs, pairs of 2-jobs minimize their cost (to $\frac{1}{2}$) if they are assigned together.

We conclude that the game has a single SE, in which the server is busy along the whole $[0, T)$ interval. Specifically, for every integer $j ≤ T/4$, the server processes a single 1-job in every slots $4j + 1$ and $4j + 3$, a pair of 2-jobs in slot $4j + 2$, and a pair of 4-jobs in slot $4j$.

The same lower bound is achieved by an instance in $G_{1, i}$: replace every $w$-job by $w$ unit-weight jobs that have the same intervals. It is easy to see that the social optimum as well as the only SE remain the same.

6 Conclusions and open problems

In this paper we analyzed, using game theoretic tools, the server’s activation cost in real-time job-scheduling systems. We showed that the limited interval-structure of players’ strategies induces a game which is more stable than general singleton cost-sharing games. Specifically, a strong equilibrium exists even in the most general setting, and the equilibrium inefficiency bounds are significantly lower than in other singleton cost-sharing games with uniform-cost resources or unweighted players.
Our results imply that if the system is controlled by rational selfish users, then the increase in its activation cost is limited. This is valid especially if the server’s activation cost does not vary over time, or if clients have uniform resource demand, and even if users can form coalitions and coordinate their assignment.

This is the first work that studies real-time scheduling games, and it can be extended in various directions:

1. Consider games with negative congestion effect. In our setting, the slot-activation cost is shared by the jobs assigned in it, thus, jobs have an incentive to join other jobs. Games in which jobs’ costs increases with the congestion require different analysis. Some effort has already been made in this direction (Georgoulaki et al. 2021).

2. Study games with variable-length jobs, in which every job is associated with a processing time $p_j$, and should select its processing interval $[t_{j,1}, t_{j,2}] \subseteq [r_j, d_j]$ such that $t_{j,2} - t_{j,1} = p_j$. The cost of processing a job is the total cost of its process. With variable-length jobs, preemptions may be allowed, inducing a different game, in which the strategy space of job $j$ consists of all subsets of size $p_j$ of $\{ r_j + 1, r_j + 2, \ldots, d_j \}$.

3. Another interesting direction is to consider systems with limited server’s capacity. Formally, for a given parameter $B$, at most $B$ jobs may be processed in every slot. In this setting, the cost-sharing mechanism should also handle the challenge of convergence to a feasible solution.

4. Finally, our analysis of the quality of strong equilibria does not cover the class of games with variable activation costs and unit job weights. The clear bounds that carry over from the analysis of the other classes are that $n > \text{SPoA} (\mathcal{G}_{v,1}) \geq \text{SPoS} (\mathcal{G}_{v,1}) \geq 2 - \varepsilon$. Bounding the SE inefficiency for this class is an interesting and challenging open problem.

References

Ackermann H, Röglin H, Vöcking B (2008) On the impact of combinatorial structure on congestion games. J ACM 55(2):1–22
Albers S (2009) On the value of coordination in network design. SIAM J Comput 38(6):2273–2302
Albers S (2010) Energy-efficient algorithms. Commun ACM 53(5):86–96
Andelman N, Feldman M, Mansour Y (2009) Strong price of anarchy. Games Econom Behav 65(2):289–317
Anshelevich E, Dasgupta A, Kleinberg J, Tardos E, Wexler T, Roughgarden T (2008) The price of stability for network design with fair cost allocation. SIAM J Comput 38(4):1602–1623
Aumann R (1959) Acceptable points in games of perfect information. Contrib Theory Games 4:287–324
Avni G, Tamir T (2016) Cost-sharing scheduling games on restricted unrelated machines. Theoret Comput Sci 646:26–39
Avni G, Kupferman O, Tamir T (2014) Network-formation games with regular objectives. In: Proceedings of of 17th FoSSaCS, pp 119–133
Baptiste P (2000) Batching identical jobs. Math Methods Oper Res 52(3):355–367
Bar-Noy A, Guha S, Naor J, Schieber B (2001) Approximating the throughput of multiple machines in real-time scheduling. SIAM J Comput 31(2):331–352
Bilò V, Vinci C (2017) On the impact of singleton strategies in congestion games. Proc ESA 17(14):1–17
Bilò V, Flammini M, Moscardelli L (2020) The price of stability for undirected broadcast network design with fair cost allocation is constant. Games Econ Behav 123:359–376
Caragiannis I, Flammini M, Kaklamanis C, Kanellopoulos P, Moscardelli L (2011) Tight bounds for selfish and greedy load balancing. Algorithmica 61(3):606–637
Caragiannis I, Gkatzelis V, Vinci C (2017) Coordination mechanisms, cost-sharing, and approximation algorithms for scheduling. In: Proceedings of 13th WINE
Chang J, Erlebach T, Gailis R, Khuller S (2011) Broadcast scheduling: algorithms and complexity. ACM Trans Algorithms 47:1
Chang J, Gabow HN, Khuller S (2014) A model for minimizing active processor time. Algorithmica 70(3):368–405
Chekuri C, Chuzhoy J, Lewin-Eytan L, Naor J, Orda A (2007) Non-cooperative multicast and facility location games. IEEE J Sel Areas Commun 25(6):1193–1206
Chen H, Roughgarden T (2009) Network design with weighted players. Theory Comput Syst 45(2):302–324
Cole R, Correa JR, Gkatzelis V, Mirrokni VS, Olver N (2015) Decentralized utilitarian mechanisms for scheduling games. Games Econ Behav 92:306–326
de Jong J, Klimm M, Uetz M (2016) Efficiency of equilibria in uniform matroid congestion games. In: Proceedings of 10th SAGT, pp 105–116
Dumitrescu D, Lung RI, Gasko N, Nagy R (2010) Job scheduling and bin packing from a game theoretical perspective: an evolutionary approach. In: 12th international symposium on symbolic and numeric algorithms for scientific computing
Epstein A, Feldman M, Mansour Y (2009) Strong equilibrium in cost sharing connection games. Games Econom Behav 67(1):51–68
Even-Dar E, Kesselman A, Mansour Y (2003) Convergence time to nash equilibria. In: Proceedings of ICALP, pp 502–513
Fiat A, Kaplan H, Levy M, Olonetsky S, Shabo R (2006) On the price of stability for designing undirected networks with fair cost allocations. In: Proceedings of ICALP(1), pp 608–618
Flammini M, Monaco G, Moscardelli L, Shachnai H, Shalom M, Tamir T, Zaks S (2010) Minimizing total busy time in parallel scheduling with application to optical networks. Theoret Comput Sci 411(40–42):3553–3562
Fotakis D, Kontogiannis S, Spirakis P (2005) Selfish unsplitable flows. Theoret Comput Sci 348(2–3):226–239
Gairing M, Schoppmann F (2007) Total latency in singleton congestion games. In: Proceedings of 3rd WINE, pp 381–387
Georgoulaki E, Kollias K, Tamir T (2021) Equilibrium inefficiency and computation in cost-sharing games in real-time scheduling systems. Algorithms 14:103
Gkatzelis V, Kollias K, Roughgarden T (2016) Optimal cost-sharing in general resource selection games. J Oper Res 64:6
Harks T, Klimm M (2012) On the existence of pure Nash equilibria in weighted congestion games. Math Oper Res 37(3):419–436
Harks T, Miller K (2011) The worst-case efficiency of cost sharing methods in resource allocation games. J Oper Res 59(6):1491–1503
Ieong S, McGrew R, Nudelman E, Shoham Y, Sun Q (2005) Fast and compact: a simple class of congestion games. In: Proceedings of 20th AAAI, pp 489–494
Irani S, Pruhs KR (2005) Algorithmic problems in power management. SIGACT News 36(2):63–76
Jensen J (1906) Sur les fonctions convexes et les inegalits entre les valeurs moyennes. Acta Math 30(1):175–193
Khandekar R, Schieber B, Shachnai H, Tamir T (2015) Real-time scheduling to minimize machine busy time. J Sched 18(6):561–573
Koutsoupias E, Papadimitriou C (2009) Worst-case equilibria. Comput Sci Rev 3(2):65–69
Leung JY-T (2004) Handbook of scheduling: algorithms, models, and performance analysis. CRS Press, Boca Raton
Milchtaich I (1996) Congestion games with player-specific payoff functions. Games Econom Behav 13(1):111–124
Cost-sharing games in real-time scheduling systems

Rosenthal RW (1973) A class of games possessing pure-strategy Nash equilibria. Int J Game Theory 2:65–67
Schulz AS, Stier Moses NE (2003) On the performance of user equilibria in traffic networks. In: Proceedings of 14th SODA, pp 86–87
Syrgkanis V (2010) The complexity of equilibria in cost sharing games. In: Proceedings of 6th WINE, LNCS 6484, pp 366–377
Vöcking B (2007) Algorithmic game theory, Chapter 20: selfish load balancing. Cambridge University Press, Cambridge
von Falkenhausen P, Harks T (2013) Optimal cost sharing for resource selection games. Math Oper Res 38(1):184–208

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.