Cost-Sharing Games in Real-Time Scheduling Systems

Tami Tamir(✉)

School of Computer Science, The Interdisciplinary Center (IDC), Herzliya, Israel
tami@idc.ac.il

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 its 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 result 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.

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 [10,11,26,30].

Traditional research interest in cluster systems has been high performance, such as high throughput, low response time, or load balancing [10,30]. 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., [4,14,28]).

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

© Springer Nature Switzerland AG 2018
G. Christodoulou and T. Harks (Eds.): WINE 2018, LNCS 11316, pp. 423–437, 2018.
https://doi.org/10.1007/978-3-030-04612-5_28

with *activating* the server. This scenario arises in cloud computing management, as well as in several applications in media-on-demand systems, optical-network design, and shuttle services.

As a real-life toy example, consider a big carousel (merry-go-round) in an amusement park. Rides may start in every quarter of an hour (defining the time slots), and every operation of the carousel costs a predefined amount, which is independent of the number of riders. The park attracts many groups of kids along the day. All the groups would like to have a ride on the carousel during their visit in the park. The carousel owner would like to schedule the group rides in a way that minimizes the total activation cost. The carousel's capacity is not a problem, as the number of simultaneous visitors in the park never exceeds the carousel capacity.

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. Back to our merry-go-round example, in the corresponding game, every group selects its riding time, with the understanding that groups riding together share the carousel's activation 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., 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.

1.1 Preliminaries

An instance G of our 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 $T = max_{j \in \mathcal{J}} d_j$ be the maximal deadline 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*. Let c_t denote the *activation cost* of the server in slot t.

A schedule S determines for every job j the time slot s_j in which it 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. Every feasible schedule S corresponds to a profile of the game. 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, $\ell_t(S) = \sum_{j|s_j=t} w_j$. 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 $cost_j(S) = w_j \cdot c_{s_j}/\ell_{s_j}(S)$. The total cost of a schedule is $cost(S) = \sum_{j \in \mathcal{J}} cost_j(S)$. Note that $cost(S)$ also equals the total activation cost of non-idle slots, that is, $cost(S) = \sum_{t|\ell_t(S)>0} c_t$. This cost-sharing scheme fits the commonly used proportional cost-sharing rule for weighted players (e.g., [6,9,22,32]), when the cost of a resource splits among its users proportional to their demand.

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 $cost_j(S_{-j}, s'_j) \geq cost_j(S)$.[1]

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 the study of resource allocation games in applied systems.

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., $OPT = \min_{S \in P(G)} cost(S)$. We quantify the inefficiency incurred due to self-interested behavior according to the *price of anarchy* (PoA) [29] and *price of stability* (PoS) [6] 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. Let $\mathcal{G}$ be a family of games, and let G be a game in $\mathcal{G}$. Let $\Upsilon(G)$ be the set of pure Nash equilibria of the game G. Assume that $\Upsilon(G) \neq \emptyset$.

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

A firmer notion of stability requires that a profile is stable against *coordinated deviations*. A set of players $\Gamma \subseteq \mathcal{J}$ forms a *coalition* if there exists a joint move where each job $j \in \Gamma$ strictly reduces its cost. When BRD is applied with

[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.

coordinated deviations, in every step some coalition performs a joint beneficial move. A profile S is a *Strong Equilibrium* (SE) if there is no coalition $\Gamma \subseteq \mathcal{J}$ that has a beneficial joint move from S [7]. The *strong price of anarchy* (SPoA) and the *strong price of stability* (SPoS), introduced in [5], are defined similarly to the PoA and PoS in Definition 1, where $\Upsilon(G)$ refers to the set of strong equilibria.

1.2 Related Work

This paper links two well-studied areas (i) cost-sharing games, and in particular cost-sharing games with singleton strategies, and (ii) 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* [12, 29, 32] 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 [6]. On the other hand, *Weighted* cost-sharing games, with proportional cost-sharing need not have a pure NE and the PoS may be as high as the number of players [6, 16].

The paper [33] 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 [1, 17, 25]. 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 [8]. Job scheduling on unrelated machines is a special case of this class [9].

Variants of cost-sharing games have been the subject of extensive research. It is well-known that games with player-specific costs [31] as well as other sharing variants need not have a NE. Another line of research study the effect of different cost-sharing mechanisms on the equilibrium inefficiency [16, 20, 23, 24]. A lot of attention has been given to scheduling congestion games (e.g., [18, 34]), which can be thought of as a special case of weighted congestion games with singleton strategies. The paper [21] provides bounds on the PoS for singleton congestion games, with weighted and unweighted players.

The SPoA and SPoS measures were introduced by [5], 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 [9], and for network formation games [3, 5].

To the best of our knowledge, none of the above rich literature on job-scheduling games consider the special structure of players' strategies in the setting of *real-time* scheduling.

There is a wide literature also on *real-time scheduling*, either on a single or on parallel machines (see surveys in [14,26]). All previous work on real-time scheduling consider systems controlled by a centralized authority determining the jobs' assignment. We are not aware of any results in which this setting is analyzed as a non-cooperative game. 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 [4,13]. On the other hand, with unit-weight unit-length jobs, these problems are polynomially solvable [10,14]. The papers [19,28] 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 $j \in \mathcal{J}$. Specifically, we analyze the following four classes of games:

$\mathcal{G}_{1,1}$ = {games with unit slot-activation costs and unit job-weights}.
$\mathcal{G}_{1,v}$ = {games with unit slot-activation costs and variable job-weights}.
$\mathcal{G}_{v,1}$ = {games with variable slot-activation costs and unit job-weights}.
$\mathcal{G}_{v,v}$ = {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. In addition, $\text{PoS}(\mathcal{G}_{1,v}) = 1$ and for this class we present an $O(n^2)$-time algorithm for computing, for any $G \in \mathcal{G}_{1,v}$, a NE profile $S^\star$ such that $cost(S^\star) = OPT(G)$. This result heavily exploits the interval-structure of the players' strategy space, and is in contrast to other singleton cost-sharing games, in which computing an optimal stable solution in NP-hard, even with unit-weight players [15]. The guaranteed existence of a SE is in contrast to other singleton cost-sharing games in which a SE may not exist [9]. Finally, we present an $O(n^2 + T)$-time algorithm for computing a social optimum profile for general instances.

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 [6,9], 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 $\text{PoS} = \Omega(\log n)$ [6], we show that $\text{PoS}(\mathcal{G}_{v,1})$ is the constant $\frac{8}{3}$.

Moreover, when combined with our algorithm for computing a social optimum, the PoS upper-bound proof is constructive.

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, and all PoS upper bounds are constructive, that is, for each of the four classes we present an algorithm for computing a NE whose cost is at most $PoS(\mathcal{G})\cdot OPT(G)$.

Table 1. Our results for equilibrium inefficiency with respect to unilateral deviations.

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

In Sect. 5 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 instances with variable slot-activation costs, and variable job-weights, our analysis for unilateral deviations 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 SPOA($\mathcal{G}_{1,v}$) = SPOS($\mathcal{G}_{1,1}$) and both measures are a constant – arbitrarily close to 2. Combined with our convergence proof, we conclude that natural dynamics, even with coordinated deviations allowed, are guaranteed to converge to a solution whose cost is less than $2OPT$. This result distinguishes our game from other games in which the strong price of anarchy was analyzed and shown to be either equal to the PoS ($O(\log n)$ in network formation games, and $O(n)$ in scheduling on unrelated machines) or to 1 (single-source connection games) [2,5].

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 or close to optimal solutions. We conclude in Sect. 6 with some open problems and directions for future work. Due to space constraints, some of the proofs are omitted.

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 an

$O(n^2)$ algorithm for calculating a strong equilibrium. Both results are valid for general instances – with variable slot-activation costs and variable job-weights. The algorithm generalizes an algorithm from [33] for finding a NE in unweighted singleton games.

Theorem 1. *For every $G \in \mathcal{G}_{v,v}$, any application of BRD, with unilateral or coordinated deviations, converges to a NE or a SE, respectively.*

Theorem 2. *For every $G \in \mathcal{G}_{v,v}$, a strong equilibrium exists, and can be computed efficiently.*

We turn to consider the class $\mathcal{G}_{1,v}$. We show that for any $G \in \mathcal{G}_{v,1}$, a NE assignment whose cost equals the social optimum exists, and can be computed in time $O(n^2)$.

Theorem 3. *$PoS(\mathcal{G}_{1,v}) = 1$, and for every $G \in \mathcal{G}_{1,v}$, a NE whose cost is $OPT(G)$ can be computed efficiently.*

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

Algorithm 1. Computes a NE schedule of cost $OPT(G)$ for $G \in \mathcal{G}_{1,v}$

```
1: Sort the jobs such that $d_1 \le d_2 \le \cdots \le d_k$
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: Remove all the jobs from the server and reassign them as follows:
8: while there are unassigned jobs do
9:     For every slot $b_i$, let $A(b_i)$ be the total weight of jobs for which $b_i \subseteq I_j$.
10:    Let $i^\star = \arg\min_i c_{b_i}/A(b_i)$.
11:    Assign all jobs $j$ for which $b_{i^\star} \subseteq I_j$ in slot $b_{i^\star}$.
12:    Remove the assigned jobs from the instance.
13: end while
```

The proof of the algorithm combines two claims. The first claim, whose proof is based on an exchange argument, shows that the number of slots open during the first phase is minimal. The second claim refers to the stability of the schedule produced in the second phase.

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 and the updates of $A(b_i)$ take $O(n^2)$. Thus, the time complexity of Algorithm 1 is $O(n^2)$ and is independent of T.

Note that the resulting schedule does not produce a strong equilibrium. In Sect. 5 we show that $SPOS = 2$. Specifically, Algorithm 1 fails when coordinated deviations are allowed, since moving to an idle slot may be beneficial for a coalition, but never for a single job.

Finally, we consider the problem of computing a (not necessarily stable) social optimum profile for instances with variable activation costs.

Theorem 4. *For every $G \in \mathcal{G}_{v,v}$, a profile whose cost is $OPT(G)$ can be computed efficiently.*

Proof. Let $G \in \mathcal{G}_{v,v}$. Since the server's capacity on each slot is not limited, the social optimum is independent of the jobs' weights. Also, 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 $j_2's$ slot once the assignment is done.

By the above, the jobs in $\mathcal{J}$ 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, $\alpha(j_1, j_2)$ is the cost of a cheapest slot in $I(j_1) \cap I(j_2)$. After the table α is computed, the algorithm 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). The calculation of $\alpha(j_1, j_2)$ for all $1 \leq j_1 \leq j_2 \leq n$ takes time $O(T + n^2)$. Calculating $C(j)$ takes $O(j)$, for a total of $O(n^2)$ for the whole table C. Thus, the total time complexity of the algorithm is $O(T + n^2)$.

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 $\mathcal{G}_{1,1}$. Being a sub-class of $\mathcal{G}_{1,v}$, Theorem 3 implies that $\text{PoS}(\mathcal{G}_{1,1}) = 1$. We show that the interval-structure of the players' strategies, limits the PoA to $\Theta(\sqrt{n})$.

Theorem 5. $PoA(\mathcal{G}_{1,1}) = \sqrt{4n+1} - 1$.

Proof. We begin with the lower bound. Let $n = h^2 + h$ for some integer h. We present a game $G \in \mathcal{G}_{1,1}$ 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 $\mathcal{J}$ includes $h - i$ jobs for which $I_j = [i, h+1)$. For $i = 1, \ldots, h$, the set $\mathcal{J}$ 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 $j \in \mathcal{J}$ 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 towards slot $h+1$, that is, to slots with a lower load. Since the server is busy in $[0, h)$ and $[h+1, 2h)$, $cost(S) = 2h$ and the PoA bound follows.

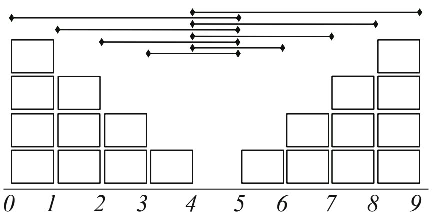

Fig. 1. A NE achieving PoA $= \sqrt{4n+1} - 1 = 8$ for $n = 20$ unit-weight jobs. The jobs' intervals are shown above the schedule.

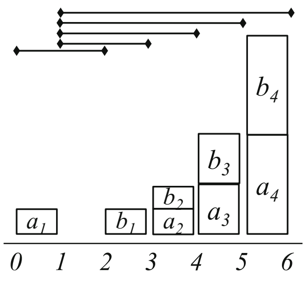

Fig. 2. A NE achieving PoA $= \frac{n}{2} + 1 = 5$ for $n = 8$ variable-weight jobs.

For the upper bound, let $S^\star$ be a social optimum schedule and assume that $cost(S^\star) = m$. Let $b_1 < b_2 < \cdots < b_m$ be the sequence of slots in which the server is busy in $S^\star$. Let S be a NE schedule. 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 $s_j^\star \in \{b_1, \ldots, b_m\}$ be the slot in which Job j is processed in $S^\star$, and let s_j be the slot in which j is processed in S. If $s_j \leq s_j^\star$, then let i be the minimal index such that $s_j \leq b_i$, and let $j \in L_i$. In other words, j belongs to the L-set of the earliest busy slot in $S^\star$ that can process it. Symmetrically, if $s_j > s_j^\star$, then let i be the maximal index such that $t_j > b_i$, and let $j \in R_i$. In other words, j belongs to the R-set of the latest busy slot in $S^\star$ that can process it. 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. 3. We first show that the loads on non-idle slots accommodating jobs from the same set form a strictly decreasing or increasing sequence.

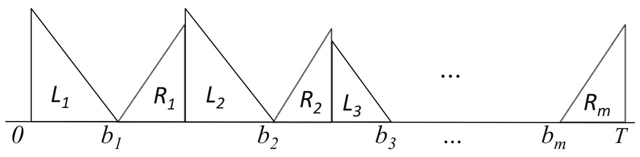

Fig. 3. The structure of a NE given that in an optimal solution the server is busy in slots $b_1, b_2, \ldots, b_m$.

Observation 6. *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 $\ell_{t_1}(S) \le \ell_{t_2}(S)$. Since $t_1 < t_2$, we have that t_2 is closer to b_i. Since $b_i \subseteq I_{j_1}$ and $t_1 < t_2 \le b_i$, it must be that $t_2 \subseteq I_{j_1}$, thus, j_1 can migrate to t_2 and reduce its cost to $\frac{1}{\ell_{t_2}(S)+1} < \frac{1}{\ell_{t_1}(S)}$, contradicting the stability of S. The analysis for R_i is symmetric (note that the word 'symmetric' is accurate here).

Observation 7. *In S, for every $1 \le i \le m$, there is at most one slot in $[b_i + 1, b_{i+1})$ 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 $[b_1 + 1, b_2)$, 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 $[b_m + 1, T)$ jobs from R_m are processed in some slots with increasing loads.

In the sequel, we assume that no slot accommodates jobs from both R_i and L_{i+1}. It can be shown that an instance with such a slot, t, can be replaced by an instance in which all the jobs processed in t are from R_i (specifically, their interval is $(b_i, t]$), with the same social optimum and the same worst NE.

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

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 8, 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 $[b_m + 1, T)$, and for every $1 \leq i < m$, at most $f(|R_i|) + f(|L_{i+1}|)$ slots are busy in $[b_i + 1, b_{i+1})$.

Given that $OPT = cost(S^\star) = m$, the PoA is at most $\frac{1}{m}\sum_{i=1}^{m}(f(|L_i|) + f(|R_i|))$. Since $f(k)$ is convex and $\sum_{i=1}^{m}(|L_i| + |R_i|) = n$, by Jensen's inequality [27], the PoA gets its maximal value when $m = 1$ and $L_1 = R_1 = n/2$. Specifically, for every $G \in \mathcal{G}_{1,1}$, we have $PoA(G) \leq 2f(n/2) = 2 \cdot \frac{1}{2}(\sqrt{4n+1} - 1) = \sqrt{4n+1} - 1$.

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 9. $PoA(\mathcal{G}_{1,v}) = \frac{n}{2} + 1$.

Proof. We begin with the upper bound and show that $\text{PoA(G)} \leq \frac{n}{2} + 1$ for every $G \in \mathcal{G}_{v,1}$. First note that if the social optimum assigns the jobs on two or more slots, then $PoA(G) \leq n/2$ follows form the fact that the maximal cost of a solution is n. Assume that $OPT(G) = 1$, and let t be a slot such that $t \subseteq I(j)$ 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 $G \in \mathcal{G}_{v,1}$ over n jobs, such that $\text{PoA}(G) = \frac{n}{2} + 1$. An example for $n = 8$ is given in Fig. 2. 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 $3 \leq j \leq z$, $w(a_j) = w(b_j) = 2^{j-2}$. The intervals of the jobs are $I(a_1) = [0, 2)$ and $I(b_1) = [1, 2)$. For $2 \leq j \leq z$, Jobs $I(a_j) = I(b_j) = [1, j+2)$. Note that for all jobs $j \in \mathcal{J}, [1, 2) \subseteq I(J)$. Thus, $OPT(G) = 1$ 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_0 and b_0 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.

4 Instances with Variable Slot-Activation Costs

In this section we discuss the equilibrium inefficiency of the classes $\mathcal{G}_{v,1}$ and $\mathcal{G}_{v,v}$. 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. On the other hand, while the PoS equals $O(\log n)$ in other singleton unweighted cost-sharing games [6], the interval strategy structure of real-time scheduling game guarantees that with unit-weight players, the PoS is $O(1)$. Moreover, our proof is constructive. First, a social optimum profile is computed (as shown in Theorem 4), and then the SO is converted to a stable profile whose cost is at most $\frac{8}{3} \cdot OPT(G)$.

Theorem 10. *$PoA(\mathcal{G}_{v,1}) = n$ and $PoS(\mathcal{G}_{v,1}) = \frac{8}{3}$.*

Theorem 11. *$PoA(\mathcal{G}_{v,v}) = n$ and for every $\varepsilon > 0$, there exists a game $G \in \mathcal{G}_{v,v}$ for which $PoS(G) = n - \varepsilon$.*

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, the following upper bound follows from simple standard arguments, and the lower bound follows from Theorem 11.

Theorem 12. *$SPoA(\mathcal{G}_{v,v}) < n$ and for every $\varepsilon > 0$, there exists a game $G \in \mathcal{G}_{v,v}$ for which $SPoS(G) \geq n - \varepsilon$.*

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. Formally,

Theorem 13. *$SPoA(\mathcal{G}_{1,v}) < 2$, and for every $\varepsilon > 0$, there exists a game $G \in \mathcal{G}_{1,1}$ for which $SPOS(G) \geq 2 - \varepsilon$.*

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.
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.

References

1. Ackermann, H., Röglin, H., Vöcking, B.: On the impact of combinatorial structure on congestion games. J. ACM **55**(6), 25:1–25:22 (2008)
2. Adany, R., Tamir, T.: Algorithms for battery utilization in electric vehicles. Appl. Artif. Intell. **28**(3), 272–291 (2014)
3. Albers, S.: On the value of coordination in network design. SIAM J. Comput. **38**(6), 2273–2302 (2009)
4. Albers, S.: Energy-efficient algorithms. Commun. ACM **53**(5), 86–96 (2010)
5. Andelman, N., Feldman, M., Mansour, Y.: Strong price of anarchy. Games Econ. Behav. **65**(2), 289–317 (2009)
6. 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)
7. Aumann, R.: Acceptable points in general cooperative n-person games. In: Contributions to the Theory of Games IV, vol. 4 (1959)
8. Avni, G., Kupferman, O., Tamir, T.: Network-formation games with regular objectives. J. Inf. Comput. **251**, 165–178 (2016)
9. Avni, G., Tamir, T.: Cost-sharing scheduling games on restricted unrelated machines. Theor. Comput. Sci. **646**, 26–39 (2016)

10. Baptiste, P.: Batching identical jobs. Math. Methods Oper. Res. **52**(3), 355–367 (2000)
11. Bar-Noy, A., Guha, S., Naor, J., Schieber, B.: Approximating the throughput of multiple machines in real-time scheduling. SIAM J. Comput. **31**(2), 331–352 (2001)
12. Caragiannis, I., Flammini, M., Kaklamanis, C., Kanellopoulos, P., Moscardelli, L.: Tight bounds for selfish and greedy load balancing. Algorithmica **61**(3), 606–637 (2011)
13. Chang, J., Erlebach, T., Gailis, R., Khuller, S.: Broadcast scheduling: algorithms and complexity. ACM Trans. Algorithms **7**(4), 47:1–47:14 (2011). https://doi.org/10.1145/2000807.2000815. Article No. 47
14. Chang, J., Gabow, H.N., Khuller, S.: A model for minimizing active processor time. Algorithmica **70**(3), 368–405 (2014)
15. Chekuri, C., Chuzhoy, J., Lewin-Eytan, L., Naor, J., Orda, A.: Non-cooperative multicast and facility location games. IEEE J. Sel. Areas Commun. **25**(6), 1193–1206 (2007)
16. Chen, H., Roughgarden, T.: Network design with weighted players. Theory Comput. Syst. **45**(2), 302–324 (2009)
17. de Jong, J., Klimm, M., Uetz, M.: Efficiency of equilibria in uniform matroid congestion games. In: Gairing, M., Savani, R. (eds.) SAGT 2016. LNCS, vol. 9928, pp. 105–116. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53354-3_9
18. von Falkenhausen, P., Harks, T.: Optimal cost sharing for resource selection games. Math. Oper. Res. **38**(1), 184–208 (2013)
19. Flammini, M., Monaco, G., Moscardelli, L., Shachnai, H., Shalom, M., Tamir, T., Zaks, S.: Minimizing total busy time in parallel scheduling with application to optical networks. Theor. Comput. Sci. **411**(40–42), 3553–3562 (2010)
20. Fotakis, D., Kontogiannis, S., Spirakis, P.: Selfish unsplittable flows. Theor. Comput. Sci. **348**(2–3), 226–239 (2005)
21. Gairing, M., Schoppmann, F.: Total latency in singleton congestion games. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 381–387. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77105-0_42
22. Gkatzelis, V., Kollias, K., Roughgarden, T.: Optimal cost-sharing in general resource selection games. J. Oper. Res. **64**(6), 1230–1238 (2016)
23. Harks, T., Klimm, M.: On the existence of pure nash equilibria in weighted congestion games. Math. Oper. Res. **37**(3), 419–436 (2012)
24. Harks, T., Miller, K.: The worst-case efficiency of cost sharing methods in resource allocation games. Oper. Res. **59**(6), 1491–1503 (2011)
25. Ieong, S., McGrew, R., Nudelman, E., Shoham, Y., Sun, Q.: Fast and compact: a simple class of congestion games. In: Proceedings of the 20th AAAI, pp. 489–494 (2005)
26. Irani, S., Pruhs, K.R.: Algorithmic problems in power management. SIGACT News **36**(2), 63–76 (2005)
27. Jensen, J.L.W.V.: Sur les fonctions convexes et les ingalits entre les valeurs moyennes. Acta Math. **30**, 175–193 (1906)
28. Khandekar, R., Schieber, B., Shachnai, H., Tamir, T.: Real-time scheduling to minimize machine busy time. J. Sched. **18**(6), 561–573 (2015)
29. Koutsoupias, E., Papadimitriou, C.: Worst-case equilibria. Comput. Sci. Rev. **3**(2), 65–69 (2009)
30. Leung, J., Kelly, L., Anderson, J.H.: Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press Inc., Boca Raton (2004)

31. Milchtaich, I.: Congestion games with player-specific payoff functions. Games Econ. Behav. **13**(1), 111–124 (1996)
32. Rosenthal, R.W.: A class of games possessing pure-strategy nash equilibria. Int. J. Game Theory **2**, 65–67 (1973)
33. 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). https://doi.org/10.1007/978-3-642-17572-5_30
34. Vöcking, B.: Selfish load balancing. In: Algorithmic Game Theory. Cambridge University Press (2007)