algorithms

Article

Equilibrium Inefficiency and Computation in Cost-Sharing Games in Real-Time Scheduling Systems [†]

Eirini Georgoulaki [1], Kostas Kollias [2,*] and Tami Tamir [3]

[1] National and Kapodistrian University of Athens, 157 72 Athens, Greece; eirini.geo.98@gmail.com
[2] Google Research, Mountain View, CA 94042, USA
[3] The Interdisicplinary Center, Herzliya 46150, Israel; tami@idc.ac.il
* Correspondence: kostaskollias@google.com

† An extended abstract of this paper appears in the proceedings of the 15th Conference on Web and Internet Economics (WINE) 2019, Columbia, NY, USA, 12 December 2019.

Abstract: We study cost-sharing games in real-time scheduling systems where the server's activation cost in every time slot is a function of its load. We focus on monomial cost functions and consider both the case when the degree is less than one (inducing positive congestion effect for the jobs) and when it is greater than one (inducing negative congestion effect for the jobs). For the former case, we provide tight bounds on the price of anarchy, and show that the price of anarchy grows to infinity as a polynomial of the number of jobs in the game. For the latter, we observe that existing results provide constant and tight (asymptotically in the degree of the monomial) bounds on the price of anarchy. We then turn to analyze payment mechanism with arbitrary cost-sharing, that is, when the strategy of a player includes also its payment. We show that our mechanism reduces the price of anarchy of games with n jobs and unit server costs from $\Theta(\sqrt{n})$ to 2. We also show that, for a restricted class of instances, a similar improvement is achieved for monomial server costs. This is not the case, however, for unrestricted instances of monomial costs, for which we prove that the price of anarchy remains super-constant for our mechanism. For systems with load-independent activation costs, we show that our mechanism can produce an optimal solution which is stable against coordinated deviations.

Keywords: real-time scheduling; cost-sharing games; price of anarchy

Citation: Georgoulaki, E.; Kollias, K.; Tamir, T. Equilibrium Inefficiency and Computation in Cost-Sharing Games in Real-Time Scheduling Systems. *Algorithms* **2021**, *14*, 103. https://doi.org/10.3390/a14040103

Academic Editor: Frank Werner

Received: 28 December 2020
Accepted: 23 March 2021
Published: 25 March 2021

1. Introduction

The study of real-time scheduling is motivated by the emergence and popularity of cloud computing. In particular, the problem of minimizing the system's activation cost correspond to the goal of reducing the power consumption in large computing systems [1–5]. An instance of the real-time scheduling problem consists of a server and a collection of jobs that need to be processed by the server. Each job j has a release time r_j and a deadline d_j. Time is slotted and each job needs to be processed on a single slot in $[r_j, d_j)$. The problem arises also in various operational problems. For example, assume that a ship carries cargo containers from one port to another. Jobs have delivery requirements leading to release times and deadlines within a given period of T days. An optimal schedule corresponds to the minimum number of times the ship needs to be sent in order to deliver all the packages on time. The motivating assumption is that it costs roughly the same to send the ship, regardless of the load it carries [3].

The study of real-time scheduling systems as a cost-sharing game was initiated in [6]. In this game, each job is controlled by a selfish agent who act to optimize its own utility rather than any global objective. Thus, each agent *chooses* the slot in which its job is processed. In the model studied in [6], the server has a fixed activation cost per time slot, which models the energy spent to keep the server open. In this paper, we consider a more general setting, in which the activation cost of the server depends on the load that the server has to process at any given time slot t, i.e., the energy spent is a function $c(l_t)$ that depends on the number of jobs l_t that are to be processed in time slot t. This generalized model captures many applications in cloud computing and data center optimization, and is the main focus of this work. As a concrete application, consider large computing clusters operated by large software companies, to which multiple users submit jibs for processing, paying for the use of computational resources through cost centers. Understanding the inefficiency caused due to selfish behaviour in such a system is an important goal, as well as suggesting various payment mechanisms that minimize this inefficiency.

In the standard cost-sharing setting studied in [6], each of the jobs processed at time t assumes an equal share of the server cost $c(l_t)$ (or a proportional share for a more general setting where each job places a different load on the server). Given this rule, we would expect each job j to optimize for its individual cost share and declare the slot that minimizes the ratio of the server cost to the number of jobs (which is precisely the individual cost share) among all slots in its interval $[r_j, d_j)$. When this is true for every job, we have an assignment that is a *Nash equilibrium* (NE). The inefficiency of a NE is captured by the *price of anarchy* (PoA), which is the worst case ratio of the total cost in a NE assignment over the total cost in the optimal assignment.

For the base case of unit server costs, we have $c(0) = 0$ and $c(x) = 1$ for every $x > 0$. The PoA of such games was shown by Tamir [6] to be $\Theta(\sqrt{n})$ and the question of what happens when the server has a cost that depends on the load placed on it was posed as an open problem. A major part of our results focuses precisely on answering this question. We study cost functions of the form $c(x) = x^d$ for some parameter $d > 0$. Note that for $d < 1$ the cost share function $c(x)/x$ is monotonically decreasing and that for $d > 1$ it is monotonically increasing. The game we study here belongs to the class of *congestion games* [7], the PoA of which has received significant attention in the literature. Our games are equivalent to singleton congestion games with uniform resources arranged on a line and with strategy spaces that correspond to intervals on this line. We provide tight PoA bounds for the case of positive externalities (which is the best motivated case for our application of interest), i.e., $d < 1$, showing that the PoA is $\Theta(n^{(1-d)/2})$. For $d > 1$, the upper bound for general congestion games shows the PoA does not depend on the number of jobs and is at most $d^{\Theta(d)}$ (i.e., constant for fixed d). We observe that an early $d^{\Theta(d)}$ lower bound instance [8] designed for routing games applies to our setting, showing that the PoA is in fact $d^{\Theta(d)}$.

Subsequently, we focus on the design of *coordination mechanisms* for the problem in an effort to improve the PoA. A coordination mechanism [9] is a set of a priori rules the system designer can set without knowledge of the instance. A coordination mechanism can, for example, modify the cost shares of jobs, expand or restrict their strategy spaces, etc. The modified rules of the game under the coordination mechanism will change the set of NE outcomes, hopefully to the improvement of the PoA. We design a simple coordination mechanism that merely expands the strategy space of each job by asking them to declare not only a slot but also a payment. It is clear that this mechanism requires no knowledge of the specifics of the instance and is very simple to implement. We show that this simple modification has a surprisingly strong impact on the PoA for the case of unit server costs, reducing it from infinity—specifically $\Theta(\sqrt{n})$—down to 2. For monomial cost functions of degree less than 1, we prove that our mechanism has super-constant PoA for general instances; however, we also prove that it does achieve an improvement from super-constant to constant for a special family of instances. Specifically, for instances such that the optimal solution uses a single slot (which includes, for example, instances with a common release time or a common deadline), our mechanism reduces the PoA from $\Theta(n^{(1-d)/2})$ to $\Theta(1)$. (Recall that for monomials of degree larger than 1, it is known that the PoA of congestion games is already constant [8,10] as a function of n without applying any coordination mechanism.) Finally, we show that our mechanism can be applied to yield an optimal solution that is stable against coordinated deviations of coalitions, in an environment in which different slots may have different cost.

1.1. Related Work

The work most closely related to our paper is the one in [6] where the model we study is introduced and various results are obtained for the case of constant (but possibly time-dependent) server costs with respect to the price of anarchy and the related concepts of the strong price of anarchy (inefficiency of equilibria with respect to coordinated deviations) and the price of stability (inefficiency of the best Nash equilibrium as opposed to the worst). Our work also has ties to literature on the price of anarchy of congestion games, cost-sharing in congestion games, and coordination mechanisms, which we discuss below.

Congestion games were introduced by Rosenthal [7] as a class of games that guarantee the existence of (pure) Nash equilibria. In a congestion game, there is a set of resources that the participants can use. Each of the participants has a strategy space that allows them to select one of given subsets of resources, something that allows for various expressive models, including routing in networks. The cost of each resource, which is a function of the number of participants using it, is distributed equally among its users. The price of anarchy of a special case of congestion games was first studied by Koutsoupias and Papadimitriou [11], who first introduced the price of anarchy. A long sequence of follow-up work has given a strong understanding of the price of anarchy in congestion games [8,10,12–14]. Generalizations to weighted models have also been studied [10,15], albeit with the drawback that, in contrast to standard congestion games, the existence of a (pure) Nash equilibrium is not guaranteed [16].

Cost-sharing aspects in the study of congestion games were brought into the picture to correct for the absence of equilibria in weighted games. Kollias and Roughgarden [17] showed how the Shapley value cost-sharing method can be applied to restore pure equilibria, whereas Gopalakrishnan et al. [18] showed that the more general class of generalized weighted Shapley values are the only ones that can guarantee this property. The price of anarchy of the induced games has been the subject of extensive study, with examples including those in [19–24]. Authors who used cost-sharing to improve the efficiency of equilibria include Chen et al. [25], who focused on the price of stability, and von Falkenhausen and Harks [26], who provided an approach which requires knowledge of the instance at hand.

A more general approach seeking to improve the price of anarchy is that of coordination mechanisms. A coordination mechanism gives the designer more freedom in designing the game. This freedom includes modifying the strategy spaces of the participants or changing how costs are defined on the resources. For example, coordination mechanisms were applied in a multiple machine makespan minimization scheduling setting by Christodoulou et al. [9] by means of introducing different scheduling policies on different machines. Follow up works on coordination mechanisms include those in [27–30].

Our coordination mechanism is also related to the model of *arbitrary cost sharing* and *resource buying games*, where each player picks a set of resources and submits a different payment for each one. This setting has been studied comprehensively for network design games. The work in [31] shows that a NE is not guaranteed to exist under arbitrary cost-sharing and that the PoA and PoS are large (almost equal to the number of players n). For the special case of a common destination node, the PoS is 1 and a NE is guaranteed to exist. An SPoA of $\Theta(\log n)$ is given in [32]. Other works that study arbitrary cost-sharing in network design games include those in [33–38]. Arbitrary cost sharing in the presence of convex costs was studied by Georgoulaki et al. [39]. Summarizing the results and comparing against fair cost-sharing, we observe that, in the general network design game, arbitrary cost-sharing loses the NE existence property and increases the PoS from logarithmic to linear. The situation improves for the common destination case where NE existence is maintained and the PoS improves to 1. Our work in effect provides the first instance in which arbitrary cost-sharing provides an improvement in the PoA over fair cost-sharing.

1.2. Summary of Our Results and Roadmap

Section 2 presents our model and notation. In Section 3.1, we focus on the case of monomial cost functions of degree $d < 1$ and prove that the PoA is approximately $2n^{(1-d)/2}/(1+d)$, where n is the number of jobs in the game. In Section 3.2, we discuss the case when $d > 1$ and observe that early results on congestion games [8] can be used to show that the PoA is $d^{\Theta(d)}$. Finally, in Section 4, we define our coordination mechanism and, in Section 4.1, we prove that it achieves a strong improvement on the PoA for the case of unit server costs (from $\Theta(\sqrt{n})$ to 2). Switching to monomials with $d < 1$, even though it is the case that the PoA grows to infinity for our mechanism as well, we are still able to show in Section 4.2 that we can reduce the PoA from $\Theta(n^{(1-d)/2})$ to $\Theta(1)$ for the class of *common slot instances*, in which the optimum uses a single slot (note that this class includes, among others, instances with a common release time or a common deadline). As stated above, the PoA for the case with $d > 1$ is already constant. Section 5 concludes the paper.

2. Preliminaries

The game is specified by the following parameters: (a) a cost function $c(x)$ that gives the cost at any given time slot as a function of the number of jobs x that have to be processed at that time step; (b) a time horizon T that specifies the available time slots as $t = 1, 2, \ldots, T$; and (c) a set of jobs J, with each $j \in J$ having an integer release time r_j and an integer deadline d_j such that $0 < r_j < d_j < T$. We denote by $I(j) = [r_j, d_j)$ the feasible interval of job j. Note that the slot t corresponds to time interval $[t-1, t)$. Job j needs to be processed in one time slot in its interval.

Let s_j denote the slot declared by job j such that $r_j \leq s_j < d_j$ and let s denote the *assignment*, i.e., the vector of declared slots. The load on slot t, $l_t(s) = |\{j : s_j = t\}|$, is the number of jobs declaring slot t. The cost on slot t is then $c(l_t(s))$ and the *total cost* is:

$$C(s) = \sum_{t=1}^{T} c(l_t(s)).$$

An assignment s is a *Nash equilibrium* (NE) when for every job j we get:

$$\frac{c(l_{s_j}(s))}{l_{s_j}(s)} \leq \frac{c(l_t(s)+1)}{l_t(s)+1},$$

for every $t \neq s_j$ in $[r_j, d_j)$. This expression suggests that the cost share at the slot declared by j is at most the cost share j would get by deviating to any other slot in its interval. The inefficiency of equilibrium solutions is given by the *price of stability* (PoS), and the *price of anarchy* (PoA), which are, respectively, the best and worst case ratio of the total cost in a NE over the total cost in the optimal assignment:

$$PoS = \frac{\min_{s \text{ a NE}} C(s)}{\min_{s^*} C(s^*)}, PoA = \frac{\max_{s \text{ a NE}} C(s)}{\min_{s^*} C(s^*)}.$$

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. An assignment s is a *Strong Equilibrium* (SE) if there is no coalition $\Gamma \subseteq J$ that has a beneficial joint move from s [40]. The *strong price of anarchy* (SPoA) and the *strong price of stability* (SPoS) introduced in [41], measure the inefficiency of strong equilibria.

$$SPoS = \frac{\min_{s \text{ a SE}} C(s)}{\min_{s^*} C(s^*)}, SPoA = \frac{\max_{s \text{ a SE}} C(s)}{\min_{s^*} C(s^*)}.$$

3. Monomial Cost Functions

3.1. Decreasing Cost Shares: $d < 1$

In this section, we analyze the PoA for the case when the cost function equals x^d with $d < 1$, for which the cost share $c(x)/x$ is strictly decreasing. The main result of the section is the following theorem, which we prove in the two subsequent lemmas.

Theorem 1. *For games with n jobs and cost function $c(x) = x^d$ with $d < 1$, the worst case PoA is $\Theta(n^{(1-d)/2})$.*

Lemma 1. *Given a game with n jobs and cost function $c(x) = x^d$ with $d < 1$, for any NE s and any optimal assignment s^* we get:*

$$\frac{C(s)}{C(s^*)} = O\left(n^{\frac{1-d}{2}}\right).$$

Proof. Consider an optimal assignment s^* and focus on a slot t that holds $l_t(s^*) = l_t$ jobs in s^*. We write $J(t)$ for the set including these l_t jobs. The jobs in $J(t)$ have a cost of $c(l_t)$ in s^*, and we wish to bound the cost they can have in any given NE s.

We begin by proving the claim that, in the NE s and for any positive integer x, there exist at most two slots that have x jobs and include jobs from $J(t)$. We prove this claim by contradiction. Suppose there are at least three slots with x jobs that host jobs from $J(t)$. Pick any such three slots and let t' be the median. If $t' \geq t$, then we have at least two slots that are greater than or equal to t and hold x jobs including some from $J(t)$, otherwise we have at least two slots that are less than or equal to t with this property. We treat the case when there are two slots that are greater than or equal to t. The other case is symmetric. Call these slots t' and t'' with $t'' > t'$. Consider some job $j \in J(t)$ that uses t'' in s. The allowed interval of j includes t, since it is scheduled on it in s^*, and t'', since it is scheduled on it in s. Since $t \leq t' < t''$, it follows that t' is also in the allowed interval for j. However, we can observe that j has an incentive to deviate from t'' to t' and improve its cost from $c(x)/x$ to $c(x+1)/(x+1)$. This contradicts the fact that s is a NE and proves our original claim that at most two slots can have jobs from $J(t)$ and exactly x jobs.

Given the above, we return to the task of upper bounding the total payments of jobs in $J(t)$ in s. By the claim in the previous paragraph, at most $2x$ jobs from $J(t)$ can pay $c(x)/x$ in s. This means at most two jobs can pay the maximum $c(1)/1$, at most four jobs the second highest $c(2)/2$, etc. It follows that the total payments in s of the jobs in $J(t)$ are upper bounded by:

$$\sum_{j=1}^{h_t} 2c(j),$$

where h_t is the smallest integer such that $h_t^2 + h_t \geq l_t$. Then, the ratio of the total cost paid by the jobs in $J(t)$ in s over the same cost in s^* is at most:

$$\frac{\sum_{j=1}^{h_t} 2c(j)}{c(l_t)}.$$

Suppose t is in fact the slot that maximizes this ratio, meaning the above expression is an upper bound on the PoA. We get:

$$\begin{aligned}\frac{C(s)}{C(s^*)} &\leq \frac{\sum_{j=1}^{h_t} 2c(j)}{c(l_t)} = \frac{\sum_{j=1}^{h_t} 2j^d}{l_t^d} \\ &\leq \frac{\int_0^{h_t+1} x^d dx}{l_t^d} \leq \frac{2(h_t+1)^{1+d}}{(1+d)l_t^d} \\ &\leq \frac{2(h_t+1)^{1+d}}{(1+d)h_t^d(h_t-1)^d} \\ &= O\left(h_t^{1-d}\right) = O\left(l_t^{\frac{1-d}{2}}\right) = O\left(n^{\frac{1-d}{2}}\right).\end{aligned}$$

This completes the proof of the upper bound. □

Lemma 2. *For every positive integer h and for every cost function $c(x) = x^d$ with $d < 1$, there exists a game with $n = h^2 + h$ jobs, and a NE s of that game, such that:*

$$\frac{C(s)}{C(s^*)} = \Omega\left(h^{1-d}\right) = \Omega\left(n^{(1-d)/2}\right),$$

where s^* is an optimal assignment of that game.

Proof. Our instance has $n = h^2 + h$ jobs and $2h + 1$ slots. For ease of exposition, we shift time and call the slots $-h, -h+1, \dots, -1, 0, 1, \dots, h-1, h$. There exist j jobs that can use slots $[-j, 0]$ and j jobs that can use slots $[0, j]$ for $j = 1, 2, \dots, h$. Observe that slot 0 is the only one that is common to all intervals. The optimal solution s^* would place all jobs on 0 for a total cost:

$$C(s^*) = (h^2 + h)^d = \Theta(n^d).$$

Consider the following assignment s, which we argue is a NE. Every one of the j jobs with interval $[-j, 0]$ selects slot $-j$ and every one of the j jobs with interval $[0, j]$ selects slot j. Note that the number of jobs on slot t is $|t|$. This fact implies the assignment is actually a NE, since the jobs on any slot t share the slot with $|t| - 1$ other jobs, while every other slot t' between t and 0 has $|t'| \leq |t| - 1$ jobs. The cost of this assignment is:

$$C(s) = \sum_{j=1}^{h} 2j^d \geq 2\int_1^h x^d dx = \frac{2}{d+1}\left(h^{d+1} - 1\right) = \Theta\left(n^{(d+1)/2}\right).$$

Taking the ratio $C(s)/C(s^*)$ completes the proof. □

Remark 1. *From the proofs of Lemmas 1 and 2, it follows that the worst case PoA is in fact approximately:*

$$\frac{2n^{\frac{1-d}{2}}}{1+d}.$$

3.2. Increasing Cost Shares: $d > 1$

We now discuss the PoA for the case when the cost function equals x^d with $d > 1$. In this case, the cost share $c(x)/x$ is strictly increasing. We observe that the general upper bound on the PoA of congestion games and an early lower bound designed for routing games which applies to our model yield the following theorem.

Theorem 2 ([8] Theorem 4.3)**.** *For games with cost function $c(x) = x^d$ with $d > 1$, the worst case PoA is $d^{\Theta(d)}$.*

Note that the PoA is constant when d is fixed and is independent of the number of players n, in contrast to the unit and $d < 1$ cases. In this work, we are mostly interested in this behavior of the PoA as a function of n. However, observing the PoA as a function of d is also of interest and we note that the $d^{\Theta(d)}$ expression hides a gap. For example, for $d = 2$, the lower bound of [8] is 2 and the general upper bound for congestion games is 5/2, however the correct value for our setting is 2.01 as given by bounds in [42,43] (designed for the load balancing game but still apply for our setting). Identifying the precise PoA value as a function of d for more general values of d appears to be a challenging open problem that we leave as future work.

3.3. Price of Stability

In this section, we show that, independent of d, for every monomial cost function, the price of stability is 1, that is, there exists an optimal stable assignment. On the other hand, we show that, for arbitrary cost function, the PoS is unbounded.

Theorem 3. *For every monomial cost function, x^d, $PoS = 1$.*

Proof. We show that every social optimal is a NE. Let s^* be any social optimal solution. $C(s^*) = \sum_{t=1}^{T} l_t(s^*)^d$. If $d = 1$, then, independent of the schedule, the cost of every job is 1, and s^* is a NE. Assume by contradiction that some player has a beneficial migration from a slot with load l_1 into a slot with load l_2. Thus, $l_1^{d-1} > (l_2 + 1)^{d-1}$. The change in the total cost is $(l_2 + 1)^d - l_2^d + (l_1 - 1)^d - l_1^d$.

If $d > 1$, then the migration is beneficial only if $l_1 > l_2 + 1$. For every $d > 1$, the function x^d is convex, thus the change in the total cost is negative for every $l_1 > l_2 + 1$.

If $d < 1$, then the migration is beneficial only if $l_1 < l_2 + 1$. For every $d < 1$, the function x^d is concave, thus the change in the total cost is negative for every $l_1 < l_2 + 1$.

We conclude that any beneficial migration contradicts the optimality of s^*, implying that the social optimum is a NE. □

Theorem 4. *For arbitrary cost function, $PoS = \Theta(n)$.*

Proof. Given n, ϵ, we describe a game with $PoS = (n - \epsilon)/2$. Let $c(x) = 1, \forall x < n$ and $c(n) = n - \epsilon$. The set J includes n jobs where $n - 1$ jobs must be processed in slot 1, and job n can choose between slots 1 and 2.

In the only optimal schedule, s^* all jobs except for job n are assigned on slot 1 and job n is assigned on slot 2. $c(s^*) = 1 + 1 = 2$. However, s^* is not a NE. By migrating to slot 1, job n reduces its cost from 1 to $(n - \epsilon)/n < 1$. The total cost of the resulting schedule, in which all the jobs are assigned on slot 1 is $c(n) = n - \epsilon$. This schedule is the only NE for this instance. Thus, $PoS = (n - \epsilon)/2$. □

4. Coordination Mechanism

In this section, we design a coordination mechanism that applies a simple modification to the game, without knowing anything about the instance, and significantly improves the PoA. For the case of constant (unit) costs, we show that the PoA is brought down to 2. For $c(x) = x^d$ with $d < 1$, we prove that the PoA is improved from super-constant to constant for the class of instances which have a single slot occupied in the optimal solution. This class includes instances with a common release time or a common deadline for all jobs, as well as instances with a batch of jobs centered around a common slot. We prove that, without this restriction, the PoA for $c(x) = x^d$ with $d < 1$ remains super-constant even under our mechanism. Note that for $d > 1$ the PoA is known to be constant even without applying any coordination mechanism.

Our mechanism changes the strategy space of each job j, from being simply a slot $s_j \in [r_j, d_j)$, to a pair (s_j, ξ_j) where $s_j \in [r_j, d_j)$ is again the declared slot and $\xi_j \geq 0$ is a payment. The mechanism will open slot t under strategies (s, ξ) if and only if:

$$\sum_{j:s_j=t} \xi_j \geq c(l_t(s)),$$

i.e., if the jobs selecting slot s cover the server cost with their declared payments. We assume every job j has an infinite cost for not being processed (i.e., when slot s_j is not opened), an assumption that is implicitly present in the base model as well, since jobs do not have the option of staying out of the game. In this framework, the NE condition asserts that each (s_j, ξ_j) are such that:

$$\begin{aligned} &\sum_{j':s_{j'}=s_j} \xi_{j'} \geq c(l_{s_j}(s)) \\ &\text{and } \xi_j \leq \max\left\{0, c(l_t(s)+1) - \sum_{j':s_{j'}=t} \xi_{j'}\right\}, \forall t \in [r_j, d_j) \setminus \{s_j\} \\ &\text{and } \xi_j \leq \max\left\{0, c(l_{s_j}(s)) - \sum_{j':s_{j'}=s_j, j'\neq j} \xi_{j'}\right\}. \end{aligned} \tag{1}$$

The first inequality guarantees slot s_j is open, the second that there is no slot t that job j can move to, pay the minimum needed to open it (or keep it open), and get a lower payment, and the third that the job should pay as much as necessary to keep its current slot open.

Lemma 3. *Every NE (s, ξ) of our coordination mechanism is such that for every occupied slot t we have $\sum_{j:s_j=t} \xi_j = c(l_t(s))$.*

Proof. Suppose this is not the case. Then, there exists some t such that:

$$\sum_{j:s_j=t} \xi_j > c(l_t(s)).$$

Consider any job j such that $s_j = t$ and $\xi_j > 0$. Clearly, such a job must exist. We get:

$$\sum_{j':s_{j'}=s_j} \xi_{j'} > c(l_{s_j}(s)) \Rightarrow \xi_j > c(l_{s_j}(s)) - \sum_{j':s_{j'}=s_j, j'\neq j} \xi_{j'},$$

which violates (1) and gives a contradiction. □

Using Lemma 3, we may simplify the NE condition (1) as:

$$\sum_{j':s_{j'}=s_j} \xi_{j'} = c(l_{s_j}(s)) \text{ and } \xi_j \leq c(l_t(s)+1) - \sum_{j':s_{j'}=t} \xi_{j'}, \forall t \in [r_j, d_j) \setminus \{s_j\}. \tag{2}$$

4.1. Unit Server Costs

Lemma 4. *Let (s, ξ) be a NE of our coordination mechanism in a game with unit server costs. Then, for every job j, either $\xi_j = 0$ or every slot in $[r_j, d_j) \setminus s_j$ is unoccupied.*

Proof. Suppose $\xi_j > 0$. Then, by (2), we get that for any slot $t \in [r_j, d_j) \setminus \{s_j\}$:

$$1 - \sum_{j':s_{j'}=t} \xi_{j'} \geq \xi_j > 0. \tag{3}$$

However, by Lemma 3, we know that either t is unoccupied or $\sum_{j':s_{j'}=t} \xi_{j'} = 1$. From this, and given (3), we get that t is unoccupied. □

Theorem 5. *The PoA of our coordination mechanism for unit server costs is* 2*.*

Proof. Focus on a slot t and the set of jobs $J(t)$ that use it in a given optimal assignment. The total cost paid by these jobs in the optimal assignment is 1. We show that the same set of jobs pay at most 2 in any NE (s, ξ).

We examine two cases. The first is the case when slot t is open in (s, ξ). The jobs from $J(t)$ that use t in (s, ξ) pay at most 1, as given by Lemma 3 (they might be paying less than 1 as t might have additional jobs outside $J(t)$). The jobs in $J(t)$ that use other slots pay 0, as given by Lemma 4 and the fact that all of these jobs have an occupied slot in their windows other than the one they are using, namely, slot t. This proves that, in this case, the total payments of jobs in $J(t)$ are at most 1.

We now focus on the case when slot t is not open in (s, ξ). We first show that the jobs from $J(t)$ that are using slots larger than t are paying a total of at most 1. Focus on the smallest such slot t'. By the fact that, (s, ξ) is a NE and using Lemma 3 we get that the total payments on t' are 1. Now focus on any slot $t'' > t'$ and any job $j \in J(t)$ that uses t''. Slot t' must be in the window of job j as both t'' and t satisfy this property and t' lies between them. Then, by Lemma 4 we get that $\xi_j = 0$. This proves that the total payment over all jobs j that use slots larger than t is 1. The proof is symmetric for slots smaller than t, which shows that for every slot t in an optimal solution and for the jobs $J(t)$ using it, the total payment of the jobs $J(t)$ in a NE is at most 2. This directly implies an upper bound of 2 for the PoA.

The above analysis is tight, as demonstrated in the following game: There exist two jobs j_1, j_2. Job j_1 can be scheduled at slots 1 or 2, whereas job j_2 can be scheduled at slots 2 or 3. Assigning job j_1 to slot 1, job j_2 to slot 3, and setting $\xi_1 = \xi_2 = 1$ gives rise to a NE with total cost 2. It is easy to verify that this is a NE as each job has only one alternative slot, slot 2, where again they would have to pay a unit cost. The optimal assignment places both jobs in slot 2 for total cost 1, thus, the PoA is 2. □

We now present a simple algorithm for computing an optimal assignment p^* and determining payments ξ such that (p^*, ξ) is a strong Nash equilibrium (SE).

Theorem 6. *Algorithm 1 produces a SE profile.*

Algorithm 1 Optimal Coordination Mechanism algorithm for unit-cost

```
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:     Let $\xi_j = 1$ and $\xi_k = 0$ for every job $k \neq j$ assigned in step 3.
5:     Remove the assigned jobs from the instance.
6: end while
```

Proof. The only players with positive payment are those assigned in their due-date slot. Let j_1, j_2 be two jobs such that $\xi_{j_1} > 0$ and $\xi_{j_2} > 0$. Assume without loss of generality that $d_{j_1} < d_{j_2}$. Since j_2 is not removed in the iteration in which j_1 is considered, it must be that $I(j_1) \cap I(j_2) = \emptyset$. A deviation of a player may be beneficial only if after the deviation it shares the cost of its target slot. However, since $I(j_1) \cap I(j_2) = \emptyset$ for any two paying players, no beneficial deviation of any coalition exists. □

To prove the optimality of the schedule produced in Algorithm 1, we prove a stronger claim, that every strong NE profile is optimal. In fact, this is valid for every game with fixed activation costs, not necessarily unit. Formally,

Theorem 7. *The SPoA of our coordination mechanism for fixed server costs is* 1*.*

Proof. For a game G, let p be a profile such that $cost(p) > cost(p^*)$ for some optimal profile p^*. We show that p is not a SE. Let A^* be the set of active slots in p^*, and let $J^*(t)$ be the set of jobs assigned to slot t in p^*. Since $cost(p) > cost(p^*)$, for at least one slot $t \in A^*$, it holds that the total payment in p of jobs in $J^*(t)$ is more than c_t. This implies that the subset of $J^*(t)$ consisting of jobs with positive payment in p can migrate to slot t, each reducing its cost. □

In light of the bound of PoA $= 2$, proved in Theorem 5, we conclude that natural dynamics that allow coordinated deviations, exploit the coordinated mechanism in an optimal way.

4.2. Monomial Server Costs with $d < 1$

We first restrict ourselves to instances such that the optimal solution uses a single slot. Clearly, this is the case if and only if there is some slot that is included in the interval of every job in the instance. This is the case, for example, when the jobs have common release times or deadlines. Note that by the instance in Lemma 2 (or simple modifications of it for the case of a common release time or a common deadline) we get that the PoA is infinite in the original game. In the next theorem, we prove that our coordination mechanism reduces the PoA to a constant for every $d < 1$ for the instances under consideration which we call *common slot instances*.

Theorem 8. *For common slot instances and $c(x) = x^d$ with $d < 1$, the PoA of our coordination mechanism is $\Theta(1)$.*

Proof. For simplicity of exposition, suppose that, in a given game, we shift time so that the only slot used by the optimal solution s^* is slot 0. Let s be a NE of the game. For simplicity and without loss of generality (due to symmetry), suppose the non-negative slots have a larger or equal cost compared to the negative slots in s. If the number of non-negative slots used in s is 1, then a bound of 2 on the PoA of the instance follows trivially, so we assume there are at least 2 non-negative slots used in the game.

Let t and t' be used slots in s such that $0 < t < t'$. Let x be the number of jobs on t and y the number of jobs on t'. Observe that t must lie in the allowed interval of all jobs using t' since $0 < t < t'$ and these jobs use t' in s and 0 in s^*. This means each of these jobs has the option to move to t and pay the marginal contribution $c(x+1) - c(x)$. In addition, note that, by Lemma 3, at least one of the jobs on t' has to pay, in s, the average cost share $c(y)/y$. The above, combined with the equilibrium condition (2), imply:

$$c(x+1) - c(x) \geq \frac{c(y)}{y}. \tag{4}$$

Note that:

$$\begin{aligned} c(x+1) - c(x) &= (x+1)^d - x^d \\ &= x^{d-1} \frac{\left(1+\frac{1}{x}\right)^d - 1}{\frac{1}{x}} \\ &\leq x^{d-1} \lim_{x \to +\infty} \frac{\left(1+\frac{1}{x}\right)^d - 1}{\frac{1}{x}} \\ &= dx^{d-1}, \end{aligned} \tag{5}$$

where the inequality follows by $d < 1$. Combining (4) with (5), we get:

$$dx^{d-1} \geq y^{d-1} \Rightarrow y \geq d^{\frac{1}{d-1}}x.$$

This suggests that every slot t must have at least $d^{1/(d-1)}$ (which is always larger than e) times the number of jobs as every other slot in $[0, t)$. This in turn implies that as we move from the largest slot closer and closer to 0, the number of jobs decreases by at least a factor $d^{1/(d-1)}$. We write $\alpha = d^{1/(d-1)}$. Then, if h is the number of jobs on the last occupied slot, we get that the cost on non-negative slots is at most:

$$\sum_{j=0}^{+\infty}\left(\frac{h}{\alpha^j}\right)^d = h^d\sum_{j=0}^{+\infty}\left(\frac{1}{\alpha^d}\right)^j = \frac{h^d}{1-\frac{1}{\alpha^d}} = \frac{h^d}{1-d^{d/(1-d)}}.$$

Recall that we assume the non-negative slots have at least as much cost as the negative ones, meaning the total cost of s is at most:

$$C(s) \leq \frac{2h^d}{1-d^{d/(1-d)}}.$$

By the fact that we have h jobs on the largest slot, there at least h jobs in the game and we get:

$$C(s^*) \geq h^d.$$

Taking the ratio gives:

$$\frac{C(s)}{C(s^*)} \leq \frac{2}{1-d^{d/(1-d)}},$$

which is a constant for every given d. □

Remark 2. *For $c(x) = \sqrt{x}$ and for common slot instances with n jobs, our coordination mechanism reduces the PoA from $4n^{1/4}/3$ to at most 4.*

We note that for common slot instances there always exists a NE. In fact, again, we prove that any optimal solution can be a NE with the correct payment vector.

Theorem 9. *For $c(x) = x^d$ with $d < 1$ and common slot instances, our coordination mechanism induces games such that, for every optimal assignment s^*, there exists a vector of payments ξ such that (s^*, ξ) is a NE.*

Proof. An optimal assignment will clearly place all jobs on the same slot. Charging each job the fair share x^{d-1} results in a NE since $x^{d-1} < 1$ with 1 being the cost any job would have to pay to deviate to a different slot and open it. □

Another class of instances for which an optimal and stable solution can be computed efficiently is the class of Laminar instances. A real-time scheduling instance is *laminar* if the intervals $I(j)$ for all jobs j form a laminar family, i.e., for any two jobs $j, j' \in \mathcal{J}$, we have $I(j) \cap I(j') = \emptyset$ or $I(j) \subset I(j')$ or $I(j') \subset I(j)$.

Since the job intervals are laminar, the jobs can be represented by a forest F of rooted trees, where each vertex in a tree in F corresponds to a job, and a vertex $v(j)$ is an ancestor of a vertex $v(j')$ if and only if $I(j') \subset I(j)$. Define the level of each vertex in the forest as follows. Any root of a tree in the forest is at level 1. For all other vertices v, the level of v is 1 plus the level of its parent.

Theorem 10. *Algorithm 2 produces a NE profile of optimal cost.*

Algorithm 2 An algorithm for finding a NE schedule such that $cost(s) = OPT$ for Laminar instances

```
1: Build the corresponding forest.
2: while there are unassigned jobs do
3:     Let $j$ be a node of maximal level. Assign $j$ and all the jobs in the path connecting $j$
       and the root of its tree on any slot $t \in I(j)$.
4:     Let $\xi_j = c(l_t)$ and $\xi_k = 0$ for every job $k \neq j$ assigned in step 3.
5:     Remove the assigned jobs from the instance.
6: end while
```

Proof. Consider the forest F. It is defined such that for every two leaves, v_1, v_2, the intervals corresponding to the jobs v_1 and v_2 do not overlap. Thus, the social optimum is at least the number of leaves in F. The profile produced by the algorithm is a social optimum because its cost is exactly the number of leaves in F.

We show it is a NE. Clearly, any job k for with $\xi_k = 0$ is stable. In addition, the only jobs with a positive payment correspond to leaves in F. Since the intervals of different leaves do not overlap, for every leaf-job, j, the only active slot in $I(j)$ is the slot it is paying for; thus, j has no feasible migration. □

Earlier in the section we prove that our coordination mechanism reduces the PoA to a constant for common slot instances. On the contrary, we show that, for general instances, the PoA remains super-constant even for our mechanism.

Theorem 11. *For a game with n jobs and $c(x) = x^d$ with $d < 1$, the worst case PoA of our mechanism is $\Omega(n^{d(1-d)})$.*

Proof. Consider a large number of jobs n such that n^d is integer. Our instance has n^d slots and n^d jobs $1, 2, \ldots, n^d$ such that job j can only be scheduled on slot j. All other $n - n^d$ jobs can be scheduled on any slot. Let s^* be the assignment where every one of the unrestricted jobs is scheduled in slot 1. We get:

$$C(s^*) = \left(n - n^d + 1\right)^d + \left(n^d - 1\right) = \Theta\left(n^d\right).$$

The first term of the sum comes from slot 1 and the second term comes from slots $2, 3, \ldots, n^d$ which hold one job each.

Now consider the following NE s. The unrestricted jobs are split equally among all slots, with each slot having $n^{1-d} - 1$ of them. The payments declared by the unrestricted jobs are 0 and the full cost of each slot is paid for by the corresponding restricted job. This outcome is a NE since the unrestricted jobs get to freeload while the restricted jobs can't move to a different slot and have to pay enough to keep their slot open and avoid the large cost of remaining unscheduled. We get:

$$C(s) = n^d \left(n^{1-d}\right)^d = \Theta\left(n^{2d-d^2}\right).$$

Taking the ratio $C(s)/C(s^*)$ completes the proof. □

4.3. Variable Load-Independent Slot Costs

In this section, we consider mechanisms for instances in which different slots may have different costs. Denote by c_t the cost of slot t. Note that this cost is fixed and independent of the load on t.

Our main result is an algorithm that determines an assignment as well as payments for the players such that the resulting profile is a strong NE.

Theorem 12. *For arbitrary fixed activation costs, for any instance, it is possible to calculate a schedule p^* and vector of payments ξ such that (p^*, ξ) is a SE, and $cost(p^*) = cost(OPT)$.*

Proof. We first remove from the instance jobs whose interval includes an interval of another job. Note that, if $I(j_2) \subseteq I(j_1)$, then j_1 can be temporarily removed from the instance. Once (p^*, ξ) are finalized, we set $\xi_{j_1} = 0$, and assign j_1 on the same slot as j_2. If $I(j_2) = I(j_1)$, then one of the jobs is removed (with an arbitrary tie-breaking rule).

Following the above preprocessing, the jobs in J can be sorted such that $r_1 \leq \cdots \leq r_n$ and $d_1 \leq \cdots \leq d_n$. An optimal solution, p^*, can be computed for such instances using dynamic programming [6]. Given p^*, we show how to split the activation costs among the jobs such that (p^*, ξ) is a SE.

For a job j, let $c^j_{min} = min_{t \in \{r_j+1,\ldots,d_j\}} c_t$ denote the cost of a cheapest slot in $I(j)$.

We turn to describe the players' payments. Let $A = \{t_1 < t_2 < \ldots < t_B\}$ be the set of active slots in p^*. For every $1 \leq i \leq B$, let $P(i)$ denote the set of jobs such that $j \in P(i)$ if and only if $A \cap I(j) = t_i$. That is, slot t_i is the only active slot in j's interval. Since A is an optimal solution, $P(i)$ is not empty, as, otherwise, t_i can be removed from A, contradicting its optimality. The jobs in $P(i)$ are going to pay for t_i. For simplicity, we first describe an algorithm to determine payments such that the resulting profile is a NE. Then, we present an algorithm for SE. □

Lemma 5. *It is possible to determine payments ξ, such that (p^*, ξ) is a NE.*

Proof. For every $j \in P(i)$, none of the other slots in A is feasible for j. Thus, as long as its payment is at most c^j_{min}, job j would not deviate from t_i. By the optimality of p^*, we have that $c_{t_i} \leq \sum_{j \in P(i)} c^j_{min}$. Thus, it is possible to determine payments $\xi_j \leq c^j_{min}$ and $\sum_{j \in P(i)} \xi_j = c_{t_i}$. □

We turn to show how it is possible to split the activation cost of slot t_i among the jobs in $P(i)$ such that the resulting profile is a SE.

Lemma 6. *It is possible to determine payments ξ, such that (p^*, ξ) is a SE.*

Proof. We first set $\xi_j = 0$ for every job $j \in J \setminus \cup_i P(i)$. For simplicity, we remove all these jobs from the input, and only consider the paying jobs—that are in $\cup_i P(i)$. For every $j \in P(i)$, no slot in $A \setminus \{t_i\}$ is feasible for j, therefore, $j \in P(i)$ may deviate only to a slot t such that $t_{i-1} + 1 \leq t \leq t_{i+1} - 1$. By the preprocessing (removal of jobs whose interval includes an interval of another job), the sorting of the jobs in $\cup_i P(i)$ according to their release times agrees also with their sorting according to finish times. In order for ξ to be a SE, for every potential deviation of a coalition Γ, we bound the total payments of $\Gamma's$ members. Specifically, the total payment of players that can potentially migrate to a slot t must be less than c_t, as, otherwise, these players can form a coalition and split the payment for activating t in a way that all of them benefit. Our algorithm is based on defining two sets of constraints:

1. *T stability* constraints, guaranteeing that (p^*, ξ) is a SE
2. *B cost-covered* constraints, guaranteeing that (p^*, ξ) is feasible, that is, the cost all slots in A is covered

We first define the constraints, and then show they can be fulfilled. Specifically, if they cannot, then p^* is not optimal. Note that, in an instance with variable slot costs, the size of the input is $\Omega(T)$, therefore, the algorithm is polynomial in the size of the input.

Consider first deviations into a slot t, where $r_1 \leq t < t_1$. Since the jobs are sorted by release times, and this sorting agrees with the sorting according to finish time, the constraint for avoiding a migration into t is $\sum_{j=1}^{x} \xi_j \leq c_t$, where x is the number of jobs for which $t \in I(j)$. In other words, for some prefixes of J, we have a constraint regarding their total payment.

Consider next deviations into a slot t, where $t_i + 1 \leq t < t_{i+1}$ for $1 \leq i < B$. A coordinated deviation into t may be performed by a coalition consisting of a possibly empty set of jobs assigned on t_i (moving right) and a possibly empty set of jobs assigned on t_{i+1} (moving left). By the sorting, the coalition consists of a *consequent* set of jobs. Let γ be the index of leftmost job for which $t \in I(j)$. Let x be the number of jobs for which $t \in I(j)$. The constraint for avoiding a migration into t is $\sum_{j=\gamma}^{\gamma+x-1} \xi_j \leq c_t$. Finally, we define the constraints that avoid a coordinated deviation into a slot $t > t_B$. Each of theses constraints considers the total payments of some suffix of the jobs, that is, $\sum_{j=\gamma}^{n} \xi_j \leq c_t$ for some γ.

Fulfillment of the above *stability*-constraints prevent beneficial deviations of any coalition. The constraints bound the total payment of every potential *maximal* coalition, and this bound clearly applies to every subset of jobs of a potential maximal coalition.

In addition, for every slot t_i in A, we define a *cost-covered* constraint $\sum_{j \in P(i)} \xi_j \geq c_{t_i}$. The cost-covered constraints imply that the active slots in p are paid for.

Given the complete set of stability and cost-covered constraints, we turn to present an algorithm for setting the payments such that all the constraints are fulfilled. During the algorithm, for every $1 \leq i \leq B$, $P(i)$ is updated to be the set of jobs assigned to t_i whose payment is not determined yet. For every slot t, let v_t denote the updated bound on the total payment of players that are assigned to or may migrate to slot t and whose payment is not determined yet. Initially, for all t, $v_t = c_t$.

Consider the first job j_1. It only participates in stability-constraints of type $\sum_{j=1}^{x} \xi_j \leq c_t$. Recall that c_{min}^1 is the minimal cost of a slot for which a stability-constraint that includes ξ_1 exists. First, set $\xi_1 = \min(c_{t_1}, c_{min}^1)$. Next, apply the fact that ξ_1 is fixed, by updating the constraints in the following way. For every $t \in I(j_1)$, let $v_t = c_t - \xi_1$. The stability-constraint $\sum_{j=1}^{x} \xi_j \leq c_t$ is replaced by a constraint $\sum_{j=2}^{x} \xi_j \leq v_t$. Note that the new constraint still refers to a consequent set of jobs. In addition, the cost-covered constraint $\sum_{j \in P(1)} \xi_j = c_{t_1}$ is replaced by $\sum_{j \in P(1) \setminus \{1\}} \xi_j \geq v_{t_1}$. Finally, set $P(1) = P(1) \setminus \{1\}$.

We proceed in a similar way for $k = 2, \ldots, n$. Assume $k \in P(i)$. Note that, when job k is considered, the value of ξ_j is already determined for every $j < k$; therefore, job k participates only in constraints of type $\sum_{j=k}^{k+x-1} \xi_j \leq v_t$. Let v_{min}^k be the minimal bound on a slot for which a constraint that includes ξ_k exists. Set $\xi_k = \min(v_{t_i}, v_{min}^k)$. Next, we apply the fact that ξ_k is fixed by updating the constraints in the following way: For every $t \in I(k)$, let $v_t = v_t - \xi_k$. The stability-constraint $\sum_{j=k}^{x+k-1} \xi_j \leq v_t$ is replaced by a constraint $\sum_{j=k+1}^{x+k-1} \xi_j \leq v_t - \xi_k$. Note that the new constraint still refers to a consequent set of jobs. In addition, the cost-covered constraint of slot t_i is replaced by $\sum_{j \in P(i) \setminus \{k\}} \xi_j \geq v_{t_i}$. Finally, set $P(i) = P(i) \setminus \{k\}$.

The process fails if, for some i, after all jobs in $P(i)$ are considered, the cost-covered constraint of slot t_i is not fulfilled. that is, $v_{t_i} > 0$. We show that in this case p^* is not optimal. Assume by contradiction that t_i is a slot whose cost is not covered. For every $k \in P(i)$, the value of ξ_k was determined such that some stability-constraint it tight. Specifically, if $\xi_k = v_{min}^k$, then job k, possibly together with jobs preceding it, covers the cost of the slot corresponding to v_{min}^k. Thus, the jobs currently on t_i can be reassigned on slots whose total cost can be fully covered by jobs in $P(i-1) \cup P(i)$, and is less than c_{t_i}. This is a contradiction to the optimality of p^*.

Thus, the cost-covered constraint for all the active slots in A are fulfilled, implying that (p^*, ξ) is feasible. Since all the stability constraints that correspond to preventing all possible coordinated deviations are fulfilled, (p^*, ξ) is a SE.

We conclude that, given a social optimum schedule p^*, it is possible to determine payments such that (p^*, ξ) is a SE. □

Recall that Theorem 7 states that SPoA = 1 for every game with fixed activation costs. Algorithm 1 implies that an optimal solution that is stable against coordinated deviations not only exists, but can also be computed efficiently.

5. Conclusions and Open Problems

In this work, we tackle the problem of load-dependent server costs in real-time scheduling systems, a question that was posed as an open problem by Tamir [6]. We precisely characterize the price of anarchy for monomial cost functions. Furthermore, we come up with a novel coordination mechanism that achieved a spectacular improvement of the price of anarchy in the original model of unit server costs and in a restricted subclass of instances in our extended model.

There are several follow-up questions that emerge from our work. These include extending our results to wider classes of cost functions and settling the PoA gap between the constant bounds for $d > 1$. Most importantly, it is interesting to see whether coordination mechanisms of the type that we define here can yield similar improvements in other settings. Our mechanism relies on the simple idea of offloading the cost sharing aspect of the problem to the jobs themselves. This induces a setting where any deviation is charged the marginal contribution of the job on the new slot. Variants of the mechanism that impose, e.g., upper bounds on the cost share of a job/player might prove useful in these endeavors (for instance, such a modification would break the bad example in Lemma 11).

Another direction for future work is to 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. The resulting game is no longer a singleton game but is more structured than arbitrary congestion games.

Author Contributions: Conceptualization, E.G., K.K. and T.T.; Formal analysis, E.G., K.K. and T.T.; Writing—original draft, E.G., K.K. and T.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Albers, S. Energy-efficient algorithms. *Commun. ACM* **2010**, *53*, 86–96. [CrossRef]
2. Bar-Noy, A.; Guha, S.; Naor, J.; Schieber, B. Approximating the Throughput of Multiple Machines in Real-Time Scheduling. *SIAM J. Comput.* **2001**, *31*, 331–352. [CrossRef]
3. Chang, J.; Gabow, H.N.; Khuller, S. A Model for Minimizing Active Processor Time. *Algorithmica* **2014**, *70*, 368–405. [CrossRef]
4. Irani, S.; Pruhs, K. Algorithmic problems in power management. *SIGACT News* **2005**, *36*, 63–76. [CrossRef]
5. Khandekar, R.; Schieber, B.; Shachnai, H.; Tamir, T. Real-time scheduling to minimize machine busy times. *J. Sched.* **2015**, *18*, 561–573. [CrossRef]
6. Tamir, T. Cost-Sharing Games in Real-Time Scheduling Systems. In Proceedings of the Web and Internet Economics—14th International Conference, WINE 2018, Oxford, UK, 15–17 December 2018; pp. 423–437.
7. Rosenthal, R.W. A class of games possessing pure-strategy Nash equilibria. *Int. J. Game Theory* **1973**, *2*, 65–67. [CrossRef]
8. Awerbuch, B.; Azar, Y.; Epstein, A. The price of routing unsplittable flow. In Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, 22–24 May 2005; pp. 57–66.
9. Christodoulou, G.; Koutsoupias, E.; Nanavati, A. Coordination Mechanisms. In Proceedings of the Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, 12–16 July 2004; pp. 345–357.
10. Aland, S.; Dumrauf, D.; Gairing, M.; Monien, B.; Schoppmann, F. Exact Price of Anarchy for Polynomial Congestion Games. *SIAM J. Comput.* **2011**, *40*, 1211–1233. [CrossRef]
11. Koutsoupias, E.; Papadimitriou, C.H. Worst-case equilibria. *Comput. Sci. Rev.* **2009**, *3*, 65–69. [CrossRef]
12. Anshelevich, E.; Dasgupta, A.; Kleinberg, J.M.; Tardos, É.; Wexler, T.; Roughgarden, T. The Price of Stability for Network Design with Fair Cost Allocation. *SIAM J. Comput.* **2008**, *38*, 1602–1623. [CrossRef]
13. Christodoulou, G.; Koutsoupias, E. The price of anarchy of finite congestion games. In Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, 22–24 May 2005; pp. 67–73.
14. Roughgarden, T. Intrinsic Robustness of the Price of Anarchy. *J. ACM* **2015**, *62*, 32:1–32:42. [CrossRef]
15. Bhawalkar, K.; Gairing, M.; Roughgarden, T. Weighted Congestion Games: The Price of Anarchy, Universal Worst-Case Examples, and Tightness. *ACM Trans. Econ. Comput.* **2014**, *2*, 14:1–14:23. [CrossRef]

16. Goemans, M.X.; Mirrokni, V.S.; Vetta, A. Sink Equilibria and Convergence. In Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005), Pittsburgh, PA, USA, 23–25 October 2005; pp. 142–154.
17. Kollias, K.; Roughgarden, T. Restoring Pure Equilibria to Weighted Congestion Games. *ACM Trans. Econ. Comput.* **2015**, *3*, 21:1–21:24. [CrossRef]
18. Gopalakrishnan, R.; Marden, J.R.; Wierman, A. Potential Games Are *Necessary* to Ensure Pure Nash Equilibria in Cost Sharing Games. *Math. Oper. Res.* **2014**, *39*, 1252–1296. [CrossRef]
19. Gairing, M.; Kollias, K.; Kotsialou, G. Tight Bounds for Cost-Sharing in Weighted Congestion Games. In Proceedings of the Automata, Languages, and Programming—42nd International Colloquium, ICALP 2015, Kyoto, Japan, 6–10 July 2015; Part II, pp. 626–637.
20. Gkatzelis, V.; Kollias, K.; Roughgarden, T. Optimal Cost-Sharing in General Resource Selection Games. *Oper. Res.* **2016**, *64*, 1230–1238. [CrossRef]
21. Klimm, M.; Schmand, D. Sharing Non-anonymous Costs of Multiple Resources Optimally. In Proceedings of the Algorithms and Complexity—9th International Conference, CIAC 2015, Paris, France, 20–22 May 2015; pp. 274–287.
22. Marden, J.R.; Philips, M. Optimizing the price of anarchy in concave cost sharing games. In Proceedings of the 2017 American Control Conference, ACC 2017, Seattle, WA, USA, 24–26 May 2017; pp. 5237–5242.
23. Marden, J.R.; Wierman, A. Distributed Welfare Games. *Oper. Res.* **2013**, *61*, 155–168. [CrossRef]
24. Roughgarden, T.; Schrijvers, O. Network Cost-Sharing without Anonymity. *ACM Trans. Econ. Comput.* **2016**, *4*, 8:1–8:24. [CrossRef]
25. Chen, H.; Roughgarden, T.; Valiant, G. Designing Network Protocols for Good Equilibria. *SIAM J. Comput.* **2010**, *39*, 1799–1832. [CrossRef]
26. Von Falkenhausen, P.; Harks, T. Optimal Cost Sharing for Resource Selection Games. *Math. Oper. Res.* **2013**, *38*, 184–208. [CrossRef]
27. Azar, Y.; Fleischer, L.; Jain, K.; Mirrokni, V.S.; Svitkina, Z. Optimal Coordination Mechanisms for Unrelated Machine Scheduling. *Oper. Res.* **2015**, *63*, 489–500. [CrossRef]
28. Caragiannis, I. Efficient Coordination Mechanisms for Unrelated Machine Scheduling. *Algorithmica* **2013**, *66*, 512–540. [CrossRef]
29. Cole, R.; Correa, J.R.; Gkatzelis, V.; Mirrokni, V.S.; Olver, N. Decentralized utilitarian mechanisms for scheduling games. *Games Econ. Behav.* **2015**, *92*, 306–326. [CrossRef]
30. Immorlica, N.; Li, L.E.; Mirrokni, V.S.; Schulz, A.S. Coordination mechanisms for selfish scheduling. *Theor. Comput. Sci.* **2009**, *410*, 1589–1598. [CrossRef]
31. Anshelevich, E.; Dasgupta, A.; Tardos, É.; Wexler, T. Near-Optimal Network Design with Selfish Agents. *Theory Comput.* **2008**, *4*, 77–109. [CrossRef]
32. Epstein, A.; Feldman, M.; Mansour, Y. Strong equilibrium in cost sharing connection games. *Games Econ. Behav.* **2009**, *67*, 51–68. [CrossRef]
33. Anshelevich, E.; Caskurlu, B. Exact and approximate equilibria for optimal group network formation. *Theor. Comput. Sci.* **2011**, *412*, 5298–5314. [CrossRef]
34. Anshelevich, E.; Caskurlu, B. Price of Stability in Survivable Network Design. *Theory Comput. Syst.* **2011**, *49*, 98–138. [CrossRef]
35. Anshelevich, E.; Karagiozova, A. Terminal Backup, 3D Matching, and Covering Cubic Graphs. *SIAM J. Comput.* **2011**, *40*, 678–708. [CrossRef]
36. Cardinal, J.; Hoefer, M. Non-cooperative facility location and covering games. *Theor. Comput. Sci.* **2010**, *411*, 1855–1876. [CrossRef]
37. Hoefer, M. Non-Cooperative Tree Creation. *Algorithmica* **2009**, *53*, 104–131. [CrossRef]
38. Hoefer, M. Strategic cooperation in cost sharing games. *Int. J. Game Theory* **2013**, *42*, 29–53. [CrossRef]
39. Georgoulaki, E.; Kollias, K.; Tamir, T. Equilibrium Inefficiency in Resource Buying Games with Load-Dependent Costs. In Proceedings of the Algorithmic Game Theory—13th International Symposium, SAGT 2020, Augsburg, Germany, 16–18 September 2020; pp. 83–98.
40. Aumann, R. Acceptable Points in General Cooperative n-Person Games. In *Contributions to the Theory of Games*; Princeton University Press: Princeton, NJ, USA, 1959; Volume 4.
41. Andelman, N.; Feldman, M.; Mansour, Y. Strong price of anarchy. *Games Econ. Behav.* **2009**, *65*, 289–317. [CrossRef]
42. Caragiannis, I.; Flammini, M.; Kaklamanis, C.; Kanellopoulos, P.; Moscardelli, L. Tight Bounds for Selfish and Greedy Load Balancing. *Algorithmica* **2011**, *61*, 606–637. [CrossRef]
43. Suri, S.; Tóth, C.D.; Zhou, Y. Selfish Load Balancing and Atomic Congestion Games. In Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, SPAA'04, Barcelona, Spain, 27–30 June 2004; pp. 188–195.