Conflicting Congestion Effects in Resource Allocation Games

Michal Feldman[1] and Tami Tamir[2]

[1] School of Business Administration and Center for the Study of Rationality, Hebrew University of Jerusalem. *E-mail:* `mfeldman@cs.huji.ac.il`.
[2] School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. *E-mail:* `tami@idc.ac.il`.

Abstract. We consider resource allocation games with heterogeneous users and identical resources. Most of the previous work considered cost structures with either negative or positive congestion effects. We study a cost structure that encompasses both the resource's load and the job's share in the resource's activation cost.
We consider the *proportional* sharing rule, where the resource's activation cost is shared among its users proportionally to their lengths. We also challenge the assumption regarding the existence of a fixed set of resources, and consider settings with an unlimited supply of resources.
We provide results with respect to equilibrium existence, computation, convergence and quality. We show that if the resource's activation cost is shared equally among its users, a pure Nash equilibrium (NE) might not exist. In contrast, under the proportional sharing rule, a pure NE always exists, and can be computed in polynomial time. Yet, starting at an arbitrary profile of actions, best-response dynamics might not converge to a NE. Finally, we prove that the price of anarchy is unbounded and the price of stability is between 18/17 and 5/4.

1 Introduction

In resource allocation applications, tasks are assigned to resources to be performed. For example, in job scheduling models, jobs are assigned to servers to be processed, and in network routing models, traffic is assigned to network links to be routed. In the last decade, algorithmic game theory has introduced game theoretic considerations to many of these problems [17, 13, 21, 3, 2]. At the heart of the game theoretic view is the assumption that the players have *strategic* considerations and act to minimize their own cost, rather than optimizing the global objective. In resource allocation settings, this would mean that the jobs *choose* a resource instead of being assigned to one by a central designer.

The literature is divided into two main approaches with respect to the cost function. The first class of models emphasizes the negative congestion effect, and assumes that the cost of a resource is some non-decreasing function of its load. Job scheduling [11, 23] and selfish routing [10, 21] belong to this class of models. The second class assumes that each resource has some *activation cost*, which should be covered by its users, thus a user wishes to share its resource with additional users in attempt to decrease its share in the activation cost. Roughly speaking, the cost of using a resource in this class is some decreasing function of its load. Positive congestion effects have been considered in network design games [8, 5, 2].

We claim that in practice both the positive and the negative congestion effect take place. On the one hand, a heavy-loaded resource might be less preferred due to negative congestion effects; on the other hand, resources do have some activation cost, and sharing this cost with other users releases the burden on a single user. Our goal is to combine these two components into a unified cost function. Consequently, the cost function in our model is composed of (i) the load on its resource, and (ii) its share in the activation cost of its chosen resource.

An additional assumption we wish to challenge is the existence of an *a priori* given set of resources. In many practical settings a set of users controlling some jobs have the opportunity to utilize a new resource at their own cost. For example, a user might be able to purchase a dedicated server for his job if he is willing to cover its cost. Consequently, we consider settings in which the number of resources is unlimited *a priori*. (Obviously, the number of resources will never exceed the number of users.)

In our model, each resource is associated with some fixed activation cost, which should be jointly incurred by the set of jobs using it. A crucial question in this setting is how to divide the resource cost among its users. Sharing of joint costs among heterogeneous players is a common problem, and a large number of sharing rules have been proposed for this problem, each associated with different efficiency and fairness properties [15, 16, 12]. Here, our focus is not on the mechanism design point of view. Rather, we analyze two specific sharing rules with respect to equilibrium existence, computation, convergence and quality. The first rule is the *uniform* sharing rule, under which the resource's cost is shared evenly among its users. The second rule is the *proportional* sharing rule, under which the resource's cost is shared among its users in proportion to their sizes. Note that under both sharing rules, for a sufficiently small activation cost, the unique NE will be one in which each job is processed by a different resource. In the other extreme, for a sufficiently large activation cost (in a sense that will be formalized below), the unique NE will be one in which all the jobs will be assigned to a single resource.

1.1 Our Results

Equilibrium existence: Our game in its general form does not comply with the family of *potential games* (or congestion games), which always admit a NE in pure strategies [19, 14]. Thus we need to pursue new techniques for proving equilibrium existence. In particular, as we show, the cost sharing method strongly affects the equilibrium existence. Specifically, in the uniform sharing model a pure NE might not exist, while in the proportional sharing model a pure NE always exists. This motivates the use of this sharing model in our study of the remaining aspects.

Computational complexity: Under a job scheduling model with a fixed number of machines and where a user's cost is the load of its chosen machine, the longest processing time (LPT) algorithm always results in a NE [10]. Here, we devise an algorithm that computes a NE for our setting in polynomial time. The main challenge of the algorithm is to determine the number of active machines.

Convergence to equilibrium: Even if a NE exists, it is not necessarily the case that natural dynamics (like best-response dynamics (BRD), where each job, in turn, performs a best-response to the current profile) always lead to a NE. Yet, in potential games [14], BRD is guaranteed to converge to a NE. BRD is known to converge to a NE both in resource allocation games that ignore the negative congestion effects and in those ignoring the activation costs [6]. However, as we show, this is not the case in our unified model, that is, BRD might not converge to a NE. Yet, if all the jobs are of equal size, the game is a congestion game (as in [2]), and convergence of BRD is guaranteed.

Equilibrium quality: A NE may not be socially optimal. In order to quantify the inefficiency we define an objective function, and compare its value under the optimal solution and its value under some NE.

We quantify the inefficiency according to well-established measurements, namely the *price of anarchy* (PoA) [13, 18] and the *price of stability* (PoS) [2]. The PoA is defined as the ratio between the cost of the worst NE and the cost of the optimal solution, while the PoS is defined as the ratio between the cost of the best NE and the cost of the optimal solution. These metrics have been studied in a variety of applications, such as selfish routing [20], job scheduling [13, 4], network formation [7, 1, 2], facility location [22] and more. The objective function we consider is the egalitarian one, i.e., we wish to minimize the cost of the job that incurs the highest cost. We show that the PoA is not bounded. For the PoS we give an upper bound of 5/4 and a lower bound of 18/17.

All missing proofs are given in the full version of this paper [9].

2 Model and Preliminaries

An instance of our game, $G = \langle I, B \rangle$, consists of a set of n jobs, each associated with length p_j (processing time, bandwidth requirement, etc.). Let $I = \{p_1, \ldots, p_n\}$ denote the job lengths. Also given is a set of identical resources $M = \{M_1, M_2, \ldots\}$ (machines, links, etc.), each associated with an activation cost B. If the set of machines is limited, we denote $m = |M|$. While our model is general, we use terminology of job scheduling for simplicity of presentation.

The action space S_j of player j is defined as all the individual resources, i.e., $S_j = M$. The joint action space is $S = \times_{j=1}^{n} S_j$. In a joint action $s \in S$, player j selects machine s_j as its action. We denote by R_i^s the set of players on machine M_i in the joint action $s \in S$, i.e., $R_i^s = \{j : s_j = M_i\}$. The load of M_i in s, denoted by $L_i(s)$, is the sum of the weights of the players that chose machine M_i. In particular, a player can chose to be on a *dedicated* machine (i.e., assigned to a machine with no additional jobs). In this case, $L_i(s) = p_j$.

The cost function of player j, denoted by c_j, maps a joint action $s \in S$ to a real number, and is composed of two components; one depends on the total load on the chosen resource, and the other is its share in the resource's activation cost. Formally, the cost of player j under a joint action s in which $s_j = M_i$ is $c_j(s) = f(L_i(s), b_j(s))$, where $L_i(s) = \sum_{j \in R_i^s} p_j$ is the total load of players served by M_i, and $b_j(s)$ is j's share in the cost B. The function f is increasing in both $L_i(s)$ and $b_j(s)$. In this paper, we assume that $c_j(s) = L_i(s) + b_j(s)$.

The resource's activation cost may be shared among its users according to different sharing rules, two of which we consider in this paper. Under the *uniform sharing* rule, all the jobs assigned to a particular resource share its cost equally. Formally, a job assigned to M_i under joint action s pays $b_j(s) = B/|R_i^s|$. Under the

proportional sharing rule, the jobs assigned to a particular resource share its cost proportionally to their sizes. Formally, a job assigned to M_i under joint action s pays $b_j(s) = \frac{p_j B}{L_i(s)}$. For example, let $G = \langle I = \{1, 2\}, B = 12\rangle$, and let s be the schedule in which both jobs are assigned to the same machine. Then, under uniform sharing $c_1(s) = c_2(s) = 3 + 12/2 = 9$, while under proportional sharing, $c_1(s) = 3 + 12/3 = 7,\ c_2(s) = 3 + 2 \cdot 12/3 = 11$.
Nash Equilibrium (NE): A joint action $s \in S$ is a *pure* Nash Equilibrium if no player $j \in N$ can benefit from unilaterally switching his action.

Let $g(s)$ denote the social cost function under the joint action s. The optimal social cost is $OPT = \min_{s\in S} g(s)$. We consider the egalitarian objective function, in which the goal is to minimize the highest cost some player incurs. Formally, $g(s) = \max_j c_j(s)$. Let $\Phi(G)$ be the set of Nash equilibria of the game G. If $\Phi(G) \neq \emptyset$ then the *PoA (PoS) is the ratio between the* maximal *(*minimal) cost of a Nash equilibrium and the social optimum, i.e., $\max_{s\in\Phi(G)} g(s)/OPT$ $(\min_{s\in\Phi(G)} g(s)/OPT)$.

2.1 Proportional Sharing Rule - Useful Observations

In this section we present several observations that provide some intuition regarding proportional sharing. These observations will be used repeatedly in the sequel. The first observation specifies the conditions under which a job prefers to migrate from one machine to another. Note that in the standard model (where a job's cost depends only on the load on its chosen machines), the equivalent condition is simply $L_{i'}(s)+p_j > L_i(s)$. In our model, however, a migration might be beneficial even if it involves an increase of load.

Lemma 1. *Consider a schedule s. Suppose $j \in R_i^s$, and let $\rho = \frac{L_i(s)(L_{i'}(s)+p_j)}{p_j}$. Job j reduces its cost by a migration to machine i' if and only if $L_{i'}(s)+p_j > L_i(s)$ and $B > \rho$ or $L_{i'}(s) + p_j < L_i(s)$ and $B < \rho$.*

Proof. The cost of job j under schedule s is $c_j(s) = L_i(s) + p_j B/L_i(s)$. Let s' be the obtained schedule after j's migration to machine $M_{i'}$. It holds that $c_j(s') = L_{i'}(s) + p_j + p_j B/(L_{i'}(s) + p_j)$. The assertion follows immediately from comparing $c_j(s)$ and $c_j(s')$.

The following observations provide lower and upper bounds for an agent's individual cost.

Observation 1 *In any joint action s, for every job j, $c_j(s) \geq 2\sqrt{p_j B}$. Additionally, for every j s.t. $p_j \geq B$, $c_j(s) \geq p_j + B$.*

Observation 2 *In any NE, s, for every job j, $c_j(s) \leq p_j + B$.*

The following observation, whose proof can be easily derived by Lemma 1, provides some insight into beneficial and non-beneficial migrations of jobs.

Observation 3 *(i) A job j of length $p_j < B$ which is assigned to a machine with load smaller than B cannot reduce its cost by migrating to a machine with load greater than B or to a dedicated machine. (ii) Given an assignment s of jobs of lengths smaller than B s.t. $L_{i'}(s) + p_j \geq L_i(s)$ for every i, i' and j assigned to machine M_i, if $L_i(s) + L_{i'}(s) > B$, then no migration is beneficial.*

2.2 Longest Processing Time (LPT) Rule

LPT is a well-known scheduling heuristic [11]. The LPT rule sorts the jobs in a non-increasing order of their lengths and greedily assigns each job to the least loaded machine. In the traditional load-balancing problem, the LPT rule is known to produce a NE [10]. However, the stability of an LPT assignment in our setting is not clear since LPT cares about the machines' loads solely and does not consider the activation costs. Obviously, under an unlimited supply of resources, LPT will simply assign each job to a new machine, and the resulting schedule is not necessarily a NE. A natural generalization of LPT, in which each job is assigned to a machine minimizing its cost, does not necessarily lead to a NE either, even with unit-size jobs (consider for example $G = \langle I = \{1, 1, 1, 1\}, B = 4 - \varepsilon\rangle$). In this paper we use a variant of LPT (see Sections 3 and 4). The next lemma provides an important non-trivial property of the LPT algorithm, to be used in the sequel.

Lemma 2. *Let I be a set of jobs s.t. $p_j < C$ for every j. Let m be the minimal number of machines s.t. an LPT-schedule of I on m machines has makespan at most C. The total load on any two machines in the LPT-schedule on m machines is greater than C.*

3 Equilibrium Existence and Computation

3.1 No Equilibrium Under the Uniform Sharing Rule

Under the uniform sharing rule a pure NE might not exist. Consider for example the instance $G = \langle I = \{1, 10\},\ B = 4\rangle$. On dedicated machines, the jobs' costs are 5 and 14 respectively. If they are assigned together, each job pays 13. Thus, no schedule is stable: the short job will escape to a dedicated machine, while the long job will join it. This example motivates the use of the proportional sharing rule.

3.2 Equilibrium Under the Proportional Sharing Rule

In this section we prove that under the proportional sharing rule and unlimited supply of resources a pure NE always exists. Moreover, a NE can be found in time $O(nlog^2n)$. Our algorithm, denoted LPT*, uses as a subroutine the assignment rule *Longest Processing Time* (LPT) [11]. Given an instance I, let $I_{short} \subseteq I$ be the subset of jobs having length less than B, and let $I_{long} = I \setminus I_{short}$.

Algorithm LPT*:

1. Schedule each of the jobs in I_{long} on a dedicated machine.
2. The jobs of I_{short} are scheduled by algorithm LPT. The number of machines, m, is the minimal number of machines such that LPT produces a schedule having makespan at most B (i.e., LPT on $m-1$ machines produces a schedule having makespan more than B).

Note that the number of machines used in the second step is well defined, since all the participating jobs are shorter than B, therefore, a schedule having makespan less than B exists. The running time of LPT* is $O(nlog^2n)$. Long jobs are identified and scheduled in time $O(n)$, the short jobs are sorted in time $O(nlogn)$ and then LPT is executed at most $logn$ times (binary search for the right value of m - which is an integer in the range $[1, n]$).

Theorem 4. *The profile $\bar{s}$ obtained by LPT* is a NE.*

Minimal Lexicographic Assignment: In the traditional load balancing game with a fixed number of machines, the minimal lexicographic profile is known to be a NE [10]. In our model, this profile is not well-defined as the number of machines is not fixed. Let $\hat{s}^*_k$ be the lexicographically minimal assignment of I_{short} on k machines. Let m be such that the makespan under $\hat{s}^*_m$ is smaller than B whereas the makespan under $\hat{s}^*_{m-1}$ is at least B. Let $\hat{s}^*$ be the profile in which: (i) every long job is assigned to a dedicated machine, and (ii) the jobs of I_{short} are assigned according to $\hat{s}^*_m$. The proof of Theorem 4 can be easily tuned to show that $\hat{s}^*$ is a NE. However, this profile cannot be found efficiently. Moreover, as shown in Theorem 9, both $\bar{s}$ and $\hat{s}^*$ might incur arbitrarily large cost compared to the social optimum.
Identical Jobs: A simpler case is when all the jobs have the same length. Note that for this case the uniform and the proportional sharing rule coincide.

Theorem 5. *If all jobs have the same length, a NE can be computed in linear time.*

Limited Supply of Resources: Assume that the number of machines that can be used is limited. Let $m = |M|$ be the given number of machines, and let m^* be the number of machines required by algorithm LPT*. If $m^* \leq m$ then clearly LPT* produces a NE. Otherwise, it can be seen that the assignment according to LPT rule on m machines results produces a NE. Thus,

Theorem 6. *Every resource allocation game under the proportional sharing rule and a limited supply of resources admits a Nash equilibrium in pure strategies. The NE can be computed efficiently*

3.3 Convergence of Best-Response Dynamics

In this section we show that unlike other job scheduling games, in our model best-response-dynamics (BRD) do not necessarily converge to a Nash equilibrium. BRD is a local-search method in which, starting from an arbitrary joint action, in each step, some player is chosen and plays its best-response strategy (i.e., the strategy that minimizes its cost, given the strategies of the other players). By considering the instance $\langle I = \{10, 10, 10, 20\}, B = 72\rangle$, and the initial joint action $\{(10, 10); (10, 20)\}$, we get:

Theorem 7. *Under proportional sharing, BRD might not converge to a NE.*

Yet, with unit-size jobs the resulting game is a congestion game [19], thus BRD is guaranteed to converge to a NE (note that while the set of resources is not given, a game with a fixed set of n resources is equivalent to our game, thus it is a congestion game). Moreover, one can easily verify that the function $P(s) = \sum_i B \cdot H_{x_i} + \frac{1}{2}x_i^2$ where x_i denotes the number of jobs on machine i, $H_0 = 0$, and $H_k = 1 + 1/2.. + 1/k$, is a potential function for the game.

4 Equilibrium Quality

In this section we provide bounds for the price of anarchy (PoA) and the price of stability (PoS). In particular, we present sufficient condition for having $PoA = PoS = 1$, we show that the PoA is unbounded, and finally, we prove that the PoS is less than 5/4 and provide an example in which the PoS is 18/17.

Theorem 8. *If there exists a job j s.t. $p_j \geq B$, then $PoA = PoS = 1$.*

Therefore, we would like to analyze the PoA and PoS for instances in which all the jobs have load less than B. We first present an upper bound for the PoA which depends on the length of the longest job. Let $p = \alpha B$ be the length of the longest job in the instance, for some $\alpha < 1$,

Lemma 3. $PoA \leq \frac{1+\alpha}{2\sqrt{\alpha}}$.

However, α can be arbitrarily small, therefore, the PoA is not bounded, as we show below.

Theorem 9. *For any given r, there exist instances for which $PoA > r$, even with unit-size jobs.*

Proof. Given r, let $B = 4\lceil r \rceil^2$ and consider an instance with B unit-length jobs. An optimal schedule groups the jobs in sets of $\sqrt{B} = 2\lceil r \rceil$, each paying $2\sqrt{B}$. A possible NE is to schedule all the jobs on a single machine. This is a NE because each job incurs a cost of $B+1$ which cannot be reduced by migrating to a new machine. In particular, this is the NE produced by LPT*, and by finding the minimal lexicografic assignment. For this instance, $\alpha = 1/B$, and the analysis in the proof of Theorem 3 is tight. Moreover, the above construction can be repeated with B^{z+1} jobs, each of length $1/B^z$ to get $PoA = \Omega(B^{O(z/2)})$.

For standard load balancing games, it is well-known that the price of stability is 1, even for the model of unrelated machines [23]. We show that this is not the case in our model. By analyzing the instance $G = \langle I = \{2,1,1\},\ B = 4\rangle$, we get

Theorem 10. *In the resource allocation game under the proportional sharing rule, $PoS \geq \frac{18}{17}$.*

On the other hand, the price of stability is bounded by a small constant:

Theorem 11. *In any resource allocation game under the proportional sharing rule, $PoS \leq \frac{5}{4}$.*

Proof. Let αB be the length of the longest job in the instance, for $\alpha < 1$. If $\alpha > 0.25$ then by Theorem 3, $PoA < \frac{5}{4}$, and the assertion follows since $PoS \leq PoA$.

Thus, assume that $\alpha \leq 0.25$, and let $c = \sqrt{\alpha}$. Let m be the minimal number of machines such that algorithm LPT on m machines produces a schedule whose makespan is at most $2cB$. Let s be the profile obtained by LPT on m machines. We show that s is a NE: Note that for any $\alpha \leq 0.25$, $c \leq 0.5$ and thus the makespan is at most B. Therefore, by Observation 3(i), no job will migrate to a dedicated machine. Also, by Lemma 2 (applied with $C = 2cB$), the total load on any two machines is at least $2cB$, and since the maximal gap in the load between any two machines is at most αB, we have that for any two machines having loads $L_i, L_{i'}$, it holds that $L_i L_{i'} \geq (c - \frac{\alpha}{2})B(c + \frac{\alpha}{2})B = (\alpha - \alpha^2/4)B^2$. Finally, the load on any machine is at least $(c - \alpha)B$. A known property of schedules produced by LPT is that any migration involves increase in the load. By Lemma 1 such a migration is profitable for a job of length p migrating from load L_i into load $L_{i'}$ only if $B > L_i(L_{i'} + p)/p$. However $L_i(L_{i'} + p)/p = (L_i L_{i'}/p) + L_i \geq ((\alpha - \alpha^2/4)B^2/\alpha B) + (c - \alpha)B = (1 - \alpha/4 + c - \alpha)B > B$ for any $\alpha \leq 0.25$ (since $\sqrt{\alpha} > \frac{5}{4}\alpha$).

The maximal cost of a job in s is at most $2cB + \alpha BB/2cB = \frac{5}{2}\sqrt{\alpha}B$. By Observation 1 the cost of the longest job is at least $2\sqrt{\alpha}B$, thus $PoS \leq \frac{5}{4}$.

References

1. S. Albers, S. Elits, E. Even-Dar, Y. Mansour, and L. Roditty. On Nash Equilibria for a Network Creation Game. In *SODA*, 2006.
2. E. Anshelevich, A. Dasgupta, J. Kleinberg, É. Tardos, T. Wexler, and T. Roughgarden. The PoS for Network Design with Fair Cost Allocation. In *FOCS*, 2004.
3. E. Anshelevich, A. Dasgupta, E. Tardos, and T. Wexler. Near-Optimal Network Design with Selfish Agents. In *STOC*, 2003.
4. A. Czumaj and B. Vöcking. Tight Bounds for Worst-case Equilibria. *SODA*, 2002.
5. A. Epstein, M. Feldman, and Y. Mansour. Strong Equilibrium in Cost Sharing Connection Games. In *ACMEC*, 2007.
6. E. Even-Dar and Y. Mansour. Fast Convergence of Selfish Rerouting. *SODA*, 2005.
7. A. Fabrikant, A. Luthra, E. Maneva, C. Papadimitriou, and S. Shenker. On a Network Creation Game. In *PODC*, 2003.
8. J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the Cost of Multicast Transmissions. In *J. of Computer and System Sciences*, vol. 63, pages 21–41, 2001.
9. M. Feldman and T. Tamir. Conflicting Congestion Effects in Resource Allocation Games. *http://www.faculty.idc.ac.il/tami/Papers/coco.pdf.*
10. D. Fotakis, S. Kontogiannis, M. Mavronicolas, and P. Spiraklis. The Structure and Complexity of Nash Equilibria for a Selfish Routing Game. In *ICALP*, 2002.
11. R. Graham. Bounds on Multiprocessing Timing Anomalies. *SIAM J. Appl. Math.*, 17:263–269, 1969.
12. S. Herzog, S. Shenker, and D. Estrin. Sharing the "Cost" of Multicast Trees: An Axiomatic Analysis. In *IEEE/ACM Transactions on Networking*, 1997.
13. E. Koutsoupias and C. H. Papadimitriou. Worst-case Equilibria. In *STACS*, 1999.
14. D. Monderer and L. S. Shapley. Potential Games. *Games and Economic Behavior*, 14:124–143, 1996.
15. H. Moulin and S. Shenker. Serial Cost Sharing. *Econometrica*, 60:1009–1037, 1992.
16. H. Moulin and S. Shenker. Strategyproof Sharing of Submodular Costs: Budget Balance Versus Efficiency. *Journal of Economic Theory*, 18:511–533, 2001.
17. N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.
18. C. Papadimitriou. Algorithms, Games, and the Internet. In *STOC*, 2001.
19. R. W. Rosenthal. A Class of Games Possessing Pure-Strategy Nash Equilibria. *Internation Journal of Game Theory*, 2:65–67, 1973.
20. T. Roughgarden. The Price of Anarchy is Independent of the Network Topology. In *STOC*, 2002.
21. T. Roughgarden and E. Tardos. How Bad is Selfish Routing? *Journal of the ACM*, 49(2):236 – 259, 2002.
22. A. R. Vetta. Nash Equilibria in Competitive Societies with Applications to Facility Location, Traffic Routing and Auctions. In *FOCS*, 2002.
23. B. Vöcking. *Algorithmic Game Theory*, chapter *Selfish Load Balancing*. Cambridge University Press, 2007.