Conflicting Congestion Effects in Resource Allocation Games

Michal Feldman[*] Tami Tamir[†]

November 5, 2009

Abstract

We study strategic resource allocation settings, where jobs are self-interested players who choose resources with the objective of minimizing their individual cost. Our framework departs from the existing game theoretic models of resource allocation in two fundamental ways. First, while most of the previous work has considered cost structures with either negative congestion effects or positive ones, we introduce cost functions that encompass both effects. Second, we do not assume the existence of a fixed set of resources; rather, jobs can always activate new resources, but activating a new resource is costly. Specifically, in our model there is a set of heterogeneous jobs and an unlimited supply of identical resources. The cost of a job is the load on its chosen resource plus its share in the resource's activation cost, which is proportional to its length.

We provide results with respect to equilibrium existence and the inefficiency introduced due to self-interested behavior. 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 we provide a poly-time algorithm for computing it. The algorithm is a variant of the LPT (Longest processing time) algorithm, whose analysis requires the establishment of a new non-trivial property of schedules obtained by this rule.

With respect to the inefficiency of equilibria, we prove that there is no universal bound for the worst-case inefficiency (as quantified by the "price of anarchy" measure). Yet, the best-case inefficiency (quantified by the "price of stability" measure) is bounded by $\frac{5}{4}$ and this is tight. These results add another layer to the growing literature on the price of anarchy and stability, which studies the extent to which selfish behavior affects system efficiency. Finally, we observe that unlike congestion games, best-response dynamics are not guaranteed to converge to a Nash equilibrium.

[*]School of Business Administration and Center for the Study of Rationality, Hebrew University of Jerusalem. *E-mail*: `mfeldman@cs.huji.ac.il`.

[†]School of Computer Science, The Interdisciplinary Center, Herzliya, Israel. *E-mail*: `tami@idc.ac.il`.

1 Introduction

1.1 Background and Motivation

In resource allocation applications, such as job scheduling, telecommunication networks and transportation systems, tasks are assigned to resources to be processed. For example, in job scheduling models, jobs are assigned to servers to be processed; similarly, in telecommunication networks, traffic is assigned to network links to be routed. The Operations Research literature has traditionally treated these problems as combinatorial optimization problems. In the last decade, many of the Operations Research problems have been studied taking into account game theoretic considerations [4, 19, 3]. This research agenda has been put forward, in part, due to the emergence of the Internet, which is managed and shared by multiple administrative authorities and users with possibly competing interests. The analysis of applications such as routing in computer networks or the creation and design of Internet-like networks without any central design or coordination can benefit a lot from the concepts and tools game theory introduce.

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[1] *choose* a resource instead of being assigned to a resource by a central designer. Due to the decentralized nature of the Internet, this is usually the case in many Internet applications. The focus in game theory is on the *stable* outcomes of a given setting, or the *equilibrium* points. A Nash equilibrium (NE) is a profile of the users' strategies such that no user can decrease its cost by a unilateral deviation from its current strategy (given that the strategies of the remaining users do not change).

Since the users' cost functions lead them in their decisions, the structure of the cost function is a key component of any game theoretic treatment of the problem. 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, 24] and selfish routing [4, 19, 3, 14, 10, 21] belong to this class of models. The second class of models assumes that each resource has some *activation cost*, which should be covered by its users. In this case, the cost is a decreasing function in the load, and a user would wish to share its resource with as many additional users. Positive congestion effects have been considered in network design games [9, 6, 2].

While the first class ignores the positive congestion effects and the second ignores the negative congestion effects, both effects take place in practice. 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 understand the effect of these two conflicting effects by combining them into a unified cost function.

Consequently, the cost function of each job is composed of two components: (i) the load on its chosen resource, and (ii) its share in the activation cost of its chosen resource[2]. A related model

[1] Jobs, players, and users are used interchangeably throughout the paper.

[2] The concept of an activation cost has been studied in a cooperative game theoretic setting of *facility location* [13]. In the facility location game, each facility has an activation cost and the cost of each set of players is the sum of the activation cost of the facilities and the cost of connecting every player to an opened facility. However, facility location games were studied mainly within the framework of cooperative game theory which is very different from our framework. Also, the individual cost of a player is independent of the players sharing a facility with it.

has been studied in [2] for routing identical jobs on a given (fixed) graph. Since they assume that the jobs are identical and the activation cost is shared equally among each resource's users, the resulting game is a congestion game, and thus (automatically) admits a pure NE. In contrast, we deal with heterogeneous users and more involved sharing rules.

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. Similarly, a service provider might wish to stretch out new lines in order to reduce the transmission delay of its traffic. Consequently, we consider settings in which there is no a-priori limit on the number of resources, and refer to them as the "unlimited supply" case (although, clearly, the number of resources will never exceed the number of users). In Section 3.1 we also deal with the case of limited supply of resources.

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 is how the resource's cost should be divided among its users. Sharing of joint costs among heterogeneous players is a common problem, for which a large number of sharing rules have been proposed, each associated with different efficiency and fairness properties [16, 17, 12]. In departure from the above literature, we analyze different sharing rules with respect to equilibrium existence, computation, convergence and quality.

In particular, we study the *uniform* sharing rule, where the resource's cost is shared evenly among its users, and the *proportional* sharing rule, where the resource's cost is shared among its users in proportion to their size. Clearly, under both sharing rules, for a sufficiently small activation cost, the unique NE will be one in which each job is processed by a dedicated resource, and for a sufficiently large activation cost, the unique NE will be one in which all the jobs will be assigned to a single resource. Our paper analyzes what happens with intermediate activation costs.

1.2 Our Results

We study four different aspects of Nash equilibrium:

Equilibrium existence: it is well known that every finite game admits a NE in mixed strategies, but not necessarily in pure strategies. In contrast to previous job scheduling models, 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 [20, 15]. Thus we need to pursue new techniques for proving equilibrium existence. We find that the proportional sharing rule, apart from being a natural rule, induces a game which always admits a pure NE. This is in contrast to the uniform sharing rule, where a NE is not guaranteed to exist.

Equilibrium inefficiency: we quantify the inefficiency incurred due to self-interested behavior according to the *price of anarchy* (PoA)[3] [14, 18] and *price of stability* (PoS) [2] measures, which quantify to what extent our systems can benefit from a central coordinator or regulator. The PoA is the worst-case inefficiency of a Nash equilibrium, and is defined as the ratio between the social cost of the worst NE and the optimal solution. The PoS measures the best-case inefficiency of a Nash equilibrium, and is defined as the ratio between the best NE and the optimal solution. These metrics have been studied in a variety of applications, such as selfish routing [22, 4, 19], job scheduling [14, 5], network formation [8, 1, 2], facility location [23] and more. The PoA and PoS measures should be quantified according to a well-defined objective function. Here, we consider the

[3]This measure has been also termed the "decentralization ratio" [3].

egalitarian objective function, i.e., we wish to minimize the cost of the highest cost among all jobs.

We find that there is no universal bound for the PoA. This result indicates that the loss due to strategic behavior can be very severe. On the other hand, we find that the PoS is bounded by 5/4 and this is tight. While a bound on the PoS ensures only that one equilibrium is approximately optimal, and thus provides a significantly weaker guarantee than a bound on the PoA, it has a natural interpretation in many network games, where the outcome can be initially designed by a central authority for subsequent use by self-interested players. Indeed, in many networking applications, agents interact with an underlying protocol that essentially proposes a collective solution to all participants.

Computational complexity: while computing a NE may be hard even for potential games, it is computationally feasible for models that are closely related to our model. For example, it is well known that 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]. In our model, LPT is not well-defined, since the number of machines is not fixed. Yet, we devise a poly-time algorithm that computes a NE.

Convergence to equilibrium: in general, natural dynamics, such as best-response dynamics (BRD), do not necessarily converge to a NE, even if one exists. It was shown that in congestion games [15] and in job scheduling games ignoring activation costs [7] BRD always converge to a NE. This is, however, shown to be false in our setting. An exception is the case of identical jobs (as in [2]), which induces a congestion game.

2 Model and Preliminaries

We first present a general game theoretic setting and then describe the resource allocation setting considered in this paper.

A game is denoted by a tuple $G = \langle N, (S_j), (c_j)\rangle$, where $N = \{1, \dots, n\}$ is the set of players, S_j is the finite action space of player $j \in N$, and c_j is the cost function of player j. The joint action space of the players is $S = \times_{i=1}^{n} S_i$. For a joint action $s = (s_1, \dots, s_n) \in S$, we denote by s_{-j} the actions of players $j' \neq j$, i.e., $s_{-j} = (s_1, \dots, s_{j-1}, s_{j+1}, \dots, s_n)$. The cost function of player j, $c_j : S \to \mathbb{R}$, maps a joint action $s \in S$ to a real number.

A joint action $s \in S$ is a *pure Nash Equilibrium* (NE) if no player $j \in N$ can benefit from unilaterally deviating from his action to another action, i.e., for every $j \in N$ and every $a \in S_j$ it holds that $c_j(s_{-j}, a) \geq c_j(s)$.

2.1 Job Scheduling on Identical Machines

While our model describe many resource allocation problems, we present it using the terminology of job scheduling for simplicity of presentation.

A *job scheduling setting* with identical machines is characterized by a set of machines $M = \{M_1, M_2, \dots\}$, a set of jobs $N = \{1, \dots, n\}$, where a job $j \in N$ has a length (i.e., processing time) p_j. An assignment method produces an assignment s of jobs into machines, where $s_j \in M$ denotes the machine job j is assigned to. The assignment is referred to as a schedule, profile, joint action or configuration (we use all these terms interchangeably). The load of a machine M_i in a schedule s is the sum of the processing times of the jobs assigned to M_i, that is $L_i(s) = \sum_{j:s_j=M_i} p_j$.

The *makespan* of a schedule is the load on the most loaded machines. We denote it $L_{max}(s) = \max_i L_i(s)$.

Given a job scheduling setting and an activation cost B, a job scheduling game is induced, where the set of players is the set of the individual jobs, and the action space S_j of each player j is the set of the individual machines. The cost function of job j in a given schedule is composed of two components: the load on the job's machine and the job's share in the machine's activation cost; i.e., given a profile s in which $s_j = M_i$, the cost of job j is given by:

$$c_j(s) = L_i(s) + b_j(s),$$

where $L_i(s)$ is the total load on machine M_i in s, and $b_j(s)$ is j's share in the activation cost of M_i.

A few notes on the cost function are in order. The load-related component of the cost function (i.e., total load on the chosen machine) is widely-used in the literature. It characterizes systems in which jobs are processed in parallel or need to use a shared resource simultaneously; thus there is no significance to the order of the jobs. One natural example is the setting of network routing, be it choosing a driving route through a congested highway network, or a transmission route through a loaded communication network [14]. The cost of each job is the delay or latency incurred on its chosen route, thus each user aims at minimizing the total load on its chosen route.

The activation cost component in the cost function depends on the *sharing rule*. Under the *uniform* sharing rule, all the jobs assigned to a particular resource share its cost uniformly; i.e., the cost of job j such that $s_j = M_i$ is $b_j(s) = \frac{B}{|\{k:s_k=M_i\}|}$. Under the *proportional* sharing rule, the jobs assigned to a particular resource share its cost proportionally to their lengths; i.e., the cost of job j such that $s_j = M_i$ is $b_j(s) = \frac{p_j B}{L_i(s)}$.

As an example, consider an instance with activation cost $B = 12$ and two jobs of lengths $1, 2$, respectively. If both jobs are assigned to the same machine, then their costs under the uniform sharing rule would be $c_1(s) = c_2(s) = 3 + 12/2 = 9$, while their costs under the proportional rule would be $c_1(s) = 3 + 12/3 = 7,\ c_2(s) = 3 + 2 \cdot 12/3 = 11$.

The social cost function of schedule s, denoted $g(s)$, is the highest cost among all the jobs; i.e., $g(s) = \max_j c_j(s)$. We also denote by OPT the optimal solution; i.e., $OPT = \min_{s \in S} g(s)$. With this we are ready to define the PoA and PoS.

Definition 2.1. Let $\Phi(G)$ be the set of Nash equilibria of the game G. If $\Phi(G) \neq \emptyset$:

- the *price of anarchy* (PoA) is the ratio between the *maximal* cost of a Nash equilibrium and the social optimum, i.e., $\max_{s \in \Phi(G)} g(s)/OPT$; and
- the *price of stability* (PoS) is the ratio between the *minimal* cost of a Nash equilibrium and the social optimum, i.e., $\min_{s \in \Phi(G)} g(s)/OPT$.

2.2 Proportional Sharing Rule - Useful Observations

We present several claims and observations that provide some intuition regarding the behavior of jobs under the proportional sharing rule. Those observations shall be used repeatedly in the sequel.

We first specify the condition under which a job is better of migrating from one machine to another. We distinguish between two types of migrations. In a load-increasing migration, the resulting load on the target machine is larger than the load on the original machine, while in a load-decreasing migration, the resulting load on the target machine is lower than the load on the original machine. Intuitively, a load-increasing (decreasing) migration might be beneficial if the job's share in the activation cost is relatively low (high). The following assertion, which follows from simple arithmetics, specifies the exact threshold for which each type of migration is beneficial. Clearly, if the target machine has the same load as the source machine, then the job's cost does not change.

Claim 2.2. *Let s be a schedule in which $s_j = M_i$ and let $\rho = \frac{L_i(s)(L_{i'}(s)+p_j)}{p_j}$. It is beneficial for job j to migrate from machine M_i to machine $M_{i'}$ if and only if $L_{i'}(s) + p_j > L_i(s)$ and $B > \rho$ (load-increasing migration) or $L_{i'}(s) + p_j < L_i(s)$ and $B < \rho$ (load-decreasing migration).*

The following observations provide lower and upper bounds for a job's individual cost.

Observation 2.3. *In any joint action s, for every job j, $c_j(s) \geq 2\sqrt{p_j B}$. Additionally, for every j such that $p_j \geq B$, $c_j(s) \geq p_j + B$.*

Proof. The cost of j when assigned together with a (possible empty) set of jobs with total length ℓ is $c_j(s) = p_j + \ell + p_j B/(p_j + \ell)$. Simple calculus shows that this term gets its minimal value for $\ell = \sqrt{p_j B} - p_j$, for which $c_j(s) = 2\sqrt{p_j B}$. If $p_j \geq B$ then the cost is minimized for $\ell = 0$, for which, $c_j(s) = p_j + B$. □

Since a job of length p_j can always migrate to a dedicated machine and incur a cost $p_j + B$, the following observation follows trivially:

Observation 2.4. *In any NE s, for every job j, $c_j(s) \leq p_j + B$.*

The following observation, whose proof can be easily derived from Claim 2.2, provides some insight into beneficial and non-beneficial migrations of jobs.

Observation 2.5. *(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.*

Finally, one can easily verify that if the total length of all jobs does not exceed B, assigning all jobs into a single machine is a NE.

Observation 2.6. *If $B \geq \sum_j p_j$, then the schedule s in which all jobs are scheduled on a single machine is a NE.*

2.3 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, where the number of machines is fixed and the incurred cost includes only the load on the machine, the LPT rule is known to produce a NE [10]. However, if the number of machines is not limited by a fixed number, the standard LPT will simply assign each job to a dedicated machine, which will not necessarily yield a NE.

A natural generalization of LPT which assigns each job to a machine that minimizes its cost (including both the load and the activation cost components) does not necessarily lead to a NE either, even with unit-length jobs. Consider for example the game $G = \langle I = \{1,1,1,1\}, B = 4-\varepsilon\rangle$. A greedy assignment, trying to minimize the cost of a newly assigned job will result in a schedule on two machines with respective loads 3 and 1 which is not a NE.

In this paper we show that employing LPT with the "right" number of machines would produce a NE. Establishing the above assertion requires the following non-trivial property of the LPT algorithm. Recall that $L_{max}(s) = \max_i L_i(s)$ denotes the makespan of a schedule s.

Lemma 2.7. *Let $I = \{p_1, \dots, p_n\}$ be a set of job lengths, and let C be some positive real number satisfying $p_j < C$ for every j. Let s_k be a schedule of I obtained from LPT with k machines. Let m be the minimal number of machines such that $L_{max}(s_m) \le C < L_{max}(s_{m-1})$. It holds that $L_i(s_m) + L_{i'}(s_m) > C$ for every $i, i' \in [m]$.*

Proof. Assume for contradiction that the lemma is false and let I be an instance of minimal size (i.e., minimal number of jobs) contradicting it. Specifically, there are two machines in s_m whose total load is at most C. Let p_{min} denote the length of the shortest job in I.

Claim 2.8. *The most-loaded machine in s_m has at least two jobs.*

Proof. Assume for contradiction that the most loaded machine in s_m has a single job of length p_j. It is easy to see that p_j must be a longest job in I.

Consider the schedule s_{m-1}. We claim that job j must be assigned in s_{m-1} with at least one additional job. To see this, consider the set $I' = I \setminus \{j\}$. We show that if job j is assigned in s_{m-1} with no additional jobs then I' also contradicts the lemma - contradicting the minimality of the instance I. Given that j is assigned alone in s_m, the assignment of I' by LPT on $m-1$ machines is identical to its assignment as part of I in s_m, and therefore has makespan at most C. Similarly, if j is assigned alone in s_{m-1}, then the assignment of I' on $m-2$ machines is identical to its assignment as part of I in s_{m-1}, and therefore, it has makesman larger than C. By the contradiction assumption, the two lightly loaded machines in s_m have total load at most C. Note that for $m = 2$ the lemma trivially holds - if LPT fails assigning the jobs on a single machine having makespan at most C then the total load of the jobs must be more than C, thus $m > 2$ and it is feasible to apply LPT on I' and $m-2$ machines. Given that $m > 2$, the machine holding job j, which is the most-loaded machine in s_m, is not one of the two lightly loaded machine in s_m. Therefore, these two machines have exactly the same load as in the schedule of I' on $m-1$ machines. Therefore I' contradicts the lemma, contradicting the minimality of the instance I. We conclude that job j is not assigned alone in s_{m-1}.

Given that a second job is assigned to j's machine in s_{m-1}, it must be that all other $m-2$ machines have load at least p_j (else, LPT will not add the second job to j's machine). Therefore, in s_{m-1}, the load on every machine is at least p_j. We also know that there exists a machine of load greater than C. We conclude that sum of the jobs' sizes, denoted P, must satisfy $P > C + (m-2)p_j$.

Now consider s_m again. By the contradiction assumptions, there exist two machines with total load at most C and the makespan is p_j. It follows that $P \le C + (m-2)p_j$, in contradiction to the above. □

We next establish a lower bound on the makespan of s_m. Since $L_{max}(s_{m-1}) > C$ and all the jobs are shorter than C, the most loaded machine in s_{m-1} holds at least two jobs. Moreover, the most loaded machine in s_{m-1} holds the shortest job, of length p_{min}; otherwise, by removing from I the jobs shorter than p_{min} we get a smaller instance contradicting the lemma, in contradiction to the minimality of I. By LPT, it holds that for every machine i, $L_i(s_{m-1}) \geq L_{max}(s_{m-1}) - p_{min} > C - p_{min}$. In addition, there exists a machine with load greater than C. Therefore, the sum of the jobs' sizes, P, is greater than $(m-2)(C-p_{min}) + C = (m-1)C - (m-2)p_{min}$.

On the other hand, by the contradiction assumption, in s_m there are two machines with total load at most C. Therefore, the total load on the remaining $(m-2)$ machines is at least $P - C > (m-2)(C-p_{min})$. Thus there must exists a machine with load greater than $C - p_{min}$. We obtain the following lower bound:

$$L_{max}(s_m) > C - p_{min}. \tag{1}$$

We next provide a lower bound on the load of any machine in s_m. Let p_m denote the length of the m-th longest job in I, and let ℓ be the last job on the most loaded machine in s_m. By Claim 2.8, there exists a longer job than job ℓ on this machine. Since LPT first assigns a single job on each machine, $p_\ell \leq p_m$. Moreover, by LPT, the load on every machine in s_m is at least $L_{max}(s_m) - p_\ell$. We obtain the following inequality for every machine i:

$$L_i(s_m) \geq L_{max}(s_m) - p_\ell \geq L_{max}(s_m) - p_m > C - p_{min} - p_m, \tag{2}$$

where the last inequality follows from Equation 1.

In s_m, the first job on every machine has load at least p_m, while every other job has load at least p_{min}. Thus, the load of every machine in s_m with at least two jobs is at least $p_m + p_{min}$. By the assumption that there exist two machines in s_m with total load of at most C, the least loaded machine has load at most $C/2$. By Equation 2, its load is greater than $C - p_{min} - p_m$. This implies that:

$$p_\ell + p_{min} > C/2. \tag{3}$$

Let M_1, M_2 denote the least loaded and the second least loaded machine in s_m, respectively. Distinguish between three cases:

case 1: M_1 holds at least two jobs. Recall that the load of every machine with at least two jobs is at least $p_m + p_{min}$, which is greater than $C/2$ by Equation 3. Therefore, the total load on any two machines exceeds C.

case 2: M_1 holds a single job and M_2 holds at least two jobs. In this case, the load of M_2 is at least $p_m + p_{min}$, and the load on M_1 is greater than $C - p_{min} - p_m$ (by Equation 2). Summed together, their total load exceeds C.

case 3: each of M_1 and M_2 holds a single job. We distinguish between two cases:

1. $|I| \leq 2(m-1)$: A known property of LPT is that if the total number of jobs is at most twice the number of machines, then LPT produces a schedule of minimum makespan [11]. Specifically, if $|I| \leq 2(m-1)$ then any schedule of I on $m-1$ machines has makespan at least $L^{(m-1)} > C$. Consider the schedule s'_{m-1}, induced by s_m by merging the jobs on M_1 and M_2 into a single machine. It holds that $L_{max}(s'_{m-1}) = \max\{L_1(s'_{m-1}) + L_2(s'_{m-1}), L_{max}(s_m)\}$. Since $L_{max}(s_m) \leq C$, it must hold that $L_1(s'_{m-1}) + L_2(s'_{m-1}) > C$.

2. $|I| > 2(m-1)$: Since each of M_1, M_2 holds a single job, there are $|I| - 2 > 2(m-2)$ jobs that are assigned to the other $m-2$ machines, implying that there must exist a machine holding at least three jobs. The first of these jobs is of size at least p_m, and the second is of size at least p_{min}. By the fact that LPT assigned a third job to this machine and not to M_1, it must be that the load on M_1 at the time of the assignment was at least $p_m + p_{min}$, which is greater than $C/2$ by Equation 3.

We conclude that the total load on any two machines is greater than C, in contradiction to the assumption. This establishes the assertion of the lemma. $\square$

3 Equilibrium Existence, Computation and Convergence

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, they both pay 13. Thus, no schedule is stable: the short job will escape to a dedicated machine, while the long job will always join it.

In contrast, as we next show, under the proportional sharing rule a pure NE always exists and can be computed in polynomial time by the following algorithm:

Algorithm LPT*:

1. Assign each job j such that $p_j \geq B$ (henceforth *long* jobs) to a dedicated machine.
2. Assign the remaining jobs (henceforth *short* jobs) by algorithm LPT on m machines, where m is the minimal number of machines such that LPT produces a schedule with makespan at most B.

Observe 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 $O(nlog^2n)$. In particular, 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 $[1, n]$). We next prove that Algorithm LPT* produces a NE. Lemma 2.7 lies at the heart of the proof.

Theorem 3.1. *Every profile obtained by algorithm LPT* is a NE.*

Proof. Let s be a profile that is obtained by algorithm LPT*. We show that no unilateral migration from s is beneficial. By Observation 2.3 no long job can benefit from migration, and by Observation 2.5(i) no short job can benefit from joining a long job or from activating a new machine. It remains to show that no short job can benefit from migrating to another LPT-machine. Let j be a short job assigned to M_i. We show that it cannot benefit by migrating to some LPT-machine $M_{i'}$. A known property of LPT is that the load of $M_{i'}$ after the migration is at least the load of M_i in s; i.e., $L_{i'}(s) + p_j \geq L_i(s)$. Consequently, only load-increasing migrations should be considered. By Claim 2.2, it suffices to show that $B \leq \frac{L_i(s)(L_{i'}(s)+p_j)}{p_j} = \frac{L_i(s)L_{i'}(s)}{p_j} + L_i(s)$. Since $L_i(s) \geq p_j$ it follows that $\frac{L_i(s)L_{i'}(s)}{p_j} + L_i(s) \geq L_{i'}(s) + L_i(s) \geq B$, where the last inequality follows by Lemma 2.7. $\square$

Remark: We defer to the appendix a simpler proof for the existence of a pure NE under the proportional sharing rule. That proof shows that any lexicographically minimal assignment on m machines such that m is the minimal number of machines for which the lexicographically minimal assignment does not exceed B is a NE. However, Algorithm LPT* is superior since it computes a NE in polynomial time, while finding a lexicographically minimal assignment is NP-hard.

3.1 Limited Supply of Resources

In departure from the assumption made throughout the paper regarding unlimited supply of resources, in this section we consider the more standard case, where there is a given number of resources, m, but where the cost function is still composed of the load and activation cost components. The following theorem shows that the existence of a pure NE holds under a limited number of resources as well.

Theorem 3.2. *For every m, any resource allocation game with m resources induced by the proportional sharing rule admits a pure NE.*

Proof. Let m^* be the number of machines required by Algorithm LPT*. If $m^* \leq m$ then clearly LPT* produces a NE. Otherwise, let s be a schedule produced by LPT on m machines. We show that s is a NE. Since LPT activates at least the number of machines activated by LPT*, all m machines are occupied. Therefore, migrating to a dedicated machine is impossible, and we only need to consider migrations among active machines. Consider a job j of length p_j which is scheduled on machine M_i. We show that it cannot benefit by migrating to machine $M_{i'}$. It is well known that in LPT schedules, no migration can cause a decrease of load (i.e., $L_i \leq L_{i'} + p_j$). In addition, if a job incurs the same load after the migration, it is indifferent. Therefore, we only need to consider load-increasing migrations. By Claim 2.2 such a migration is profitable if and only if $B > L_i(L_{i'} + p_j)/p_j$. Therefore, it suffices to show that $B \leq L_i(L_{i'} + p_j)/p_j$. But since $L_i/pj \geq 1$, it suffices to show that $B \leq L_{i'} + p_j$. We distinguish between two cases: If there is a machine with a long job and at least one additional job, then the load on any machine is at least B. In particular $L_{i'} \geq B$, thus $L_{i'} + p_j > B$. The second case is when all the big jobs are on dedicated machines. Since $m < m^*$, the LPT-schedule of the short jobs has makespan larger than B. In particular, it means that any machine has load larger than $B - p_{min}$, implying $L_{i'} + p_j > B - p_{min} + p_j \geq B$. □

3.2 Convergence of Best-Response Dynamics

Best-Response Dynamics (BRD) is a local search method where in each step some player is chosen and plays its best-response strategy, given the strategies of the others. We next show that unlike other job scheduling games, in our model best-response-dynamics (BRD) do not necessarily converge to a Nash equilibrium.

Observation 3.3. *Under the proportional sharing rule BRD might not converge to a NE.*

Proof. Consider the instance with four jobs of lengths $10, 10, 10, 20$ and $B = 72$. In a machine with two jobs of length 10, each of the jobs incur a cost of $36 + 20 = 56$. In a machine with two jobs of lengths 10 and 20, the cost of the jobs of lengths 10 and 20 are $24 + 30 = 54$ and $48 + 30 = 78$, respectively. In a machine with three jobs of lengths $10, 10, 20$, each of the jobs of length 10 incur a cost of $18 + 40 = 58$ while the long job's cost is $36 + 40 = 76$. Consider a profile in which two jobs of length 10 are assigned to one machine and two jobs of lengths $10, 20$ are assigned to a second machine. The long job can reduce its cost by migrating to the other machine. Then, one of the jobs of length 10 assigned to the machine the long job joined can benefit by migrating to the other machine. That brings the system back to the initial configuration. Thus, BRD are not guaranteed to converge. □

In contrast to the last example, if all jobs have the same length, then the induced game is a congestion game [20] (with $m = n$), and thus best response dynamics always converge to a NE.

One can easily verify that the function $\Phi(s) = \sum_i B \cdot H_{\ell_i} + \frac{1}{2}\ell_i^2$, where ℓ_i denotes the number of jobs on machine i, $H_0 = 0$, and $H_k = 1 + 1/2 + \ldots + 1/k$, is a potential function for the game. It would be interesting to study conditions that guarantee the convergence of best response dynamics or other dynamics that ensure convergence.

4 Equilibrium Quality

In this section we study the inefficiency caused due to strategic behavior, as quantified by the price of anarchy (PoA) and price of stability (PoS) measures. We compute the PoA and PoS with respect to the objective of minimizing the highest cost among all the jobs; that is, given a profile s, the social cost of s is given by $g(s) = \max_j c_j(s)$.

We first observe that if there exists a job of length at least B, then every NE is optimal.

Observation 4.1. *If there exists a job j such that $p_j \geq B$, then $PoA = 1$.*

Proof. Let $j' = argmax_j p_j$. In particular, $p_{j'} \geq B$. By Observation 2.4, for every NE profile s and every job j, $c_j(s) \leq p_j + B$. Therefore, $max_j c_j(s) \leq p_j + B \leq p_{j'} + B$. On the other hand, in any schedule, and in particular the optimal profile s^*, $max_j c_j(s^*) \geq c_{j'}(s^*) \geq p_{j'} + B$, where the last inequality follows from Observation 2.3. Thus $PoA = 1$ as required. $\square$

In light of the last observation, we study the PoA and PoS for instances in which all the jobs are smaller than B. For this case, there is no universal bound for the PoA, as the following theorem shows.

Theorem 4.2. *For any given $r > 0$, there exists an instance for which the price of anarchy is greater than r.*

Proof. Given r, let $B = 4\lceil r \rceil^2$ and consider an instance with B unit-length jobs. An optimal schedule assigns the jobs in groups of $2\lceil r \rceil = \sqrt{B}$ jobs on each machine. In this schedule, the cost of every job is $2\sqrt{B}$. However, a schedule which assigns all the jobs on a single machine is also a NE, since each jobs incurs a cost of $B+1$, which cannot be reduced by migrating to a new machine. We get:

$$PoA \geq \frac{B+1}{2\sqrt{B}} > \frac{4\lceil r \rceil^2}{2 \cdot 2\lceil r \rceil} \geq r.$$

$\square$

While there is no universal bound for the price of anarchy, the following theorems shows that the price of stability is always smaller than 5/4, and this is tight. This should be contrasted with load balancing games in which a job's cost is its machine's load, where there exists a social optimum that is a NE [24] and thus the price of stability is 1. The following proof, in addition to bounding the PoS, also provides a polynomial time algorithm for finding a NE with cost less than factor 5/4 from the social optimum.

Theorem 4.3. *In any resource allocation game under the proportional sharing rule, $PoS < \frac{5}{4}$.*

Proof. Let $p = \alpha B$ be the length of the longest job in the instance, for $\alpha < 1$. If $\alpha > 1/4$, then we show that the price of *Anarchy* is less than $\frac{5}{4}$. By Observation 2.3 the cost of the longest job is at least $2\sqrt{pB}$, thus $OPT \geq 2\sqrt{pB}$. On the other hand, by Observation 2.4, $c_j(s) \leq p_j + B \leq p + B$ for every job j and NE profile s. It follows that $PoA \leq \frac{B+p}{2\sqrt{pB}} = \frac{1+\alpha}{2\sqrt{\alpha}}$. It is easy to verify that the last expression is smaller than 5/4 for $\alpha > 1/4$. The assertion follows since $PoS \leq PoA$.

Otherwise, $\alpha \leq 1/4$. Let m be the minimal number of machines such that algorithm LPT on m machines produces a schedule whose makespan is at most $2(\sqrt{\alpha} - \alpha^2)B$, and let s be a profile obtained by LPT on m machines. We first provide a lower bound for the load of any machine.

Claim 4.4. *The load on any machine in s is greater than $(\sqrt{\alpha} - \alpha/2 - \alpha^2)B$.*

Proof. By Lemma 2.7 (applied with $C = 2(\sqrt{\alpha} - \alpha^2)B$), the total load on any two machines must be greater than $2(\sqrt{\alpha} - \alpha^2)B$. Assume towards contradiction that the load on some machine is at most $(\sqrt{\alpha} - \alpha/2 - \alpha^2)B$. Since s is produced by LPT, the gap in the load between any two machines cannot exceed the length of the longest job, which is αB. Therefore, the load of any other machine is at most $(\sqrt{\alpha} - \alpha/2 - \alpha^2)B + \alpha B$. We get that the total load on these two machines is at most $2(\sqrt{\alpha} - \alpha^2)B$, in contradiction to Lemma 2.7. □

Next, we show that the profile s is a NE.

Claim 4.5. *The profile s is a NE.*

Proof. Observe that for any $\alpha \leq 1/4$, $2(\sqrt{\alpha} - \alpha^2) < 1$, thus the makespan is less than B. Therefore, by Observation 2.5(i), no job will migrate to a dedicated machine. Since in schedules obtained by LPT unilateral migrations cannot decrease the load, and since a job is indifferent if its load does not change, it suffices to show that no load-increasing deviation is profitable. It also follows from LPT that the gap between any two machines cannot exceed the length of the longest job, i.e., αB, and by Lemma 2.7 (applied with $C = 2(\sqrt{\alpha} - \alpha^2)B$), the total load on any two machines is greater than $2(\sqrt{\alpha} - \alpha^2)B$. Given a lower bound on the sum of of the two loads, and an upper bound on the difference between the two loads, one can obtain the following bound on their multiplication. specifically, for any two machines $M_i, M_{i'}$,

$$L_i(s)L_{i'}(s) \geq (\sqrt{\alpha} - \alpha^2 - \frac{\alpha}{2})B(\sqrt{\alpha} - \alpha^2 + \frac{\alpha}{2})B = (\alpha - 2\sqrt{\alpha}\alpha^2 + \alpha^4 - \frac{\alpha^2}{4})B^2 \tag{4}$$

By Claim 2.2 a load-increasing migration from load $L_i(s)$ into load $L_{i'}(s)$ is profitable for a job of length p_j only if $B > L_i(s)(L_{i'}(s) + p_j)/p_j$. However,

$$\begin{aligned}
\frac{L_i(s)(L_{i'}(s) + p_j)}{p_j} &= \frac{L_i(s)L_{i'}(S)}{p_j} + L_i(s) \\
&> (1 - 2\sqrt{\alpha}\alpha + \alpha^3 - \frac{\alpha}{4})B + (\sqrt{\alpha} - \frac{\alpha}{2} - \alpha^2)B = \\
&= (1 - \sqrt{\alpha}(2\alpha - 1) - \frac{3\alpha}{4} + \alpha^3 - \alpha^2)B,
\end{aligned}$$

where the inequality follows from Equation (4)and Claim 4.4. The last expression is greater than B for every $\alpha \leq 1/4$. □

Let M_i be the machine on which the maximal cost is achieved. Since $p_j \leq \alpha B$ for every job j,

$$\max_j c_j(s) = \max_{j:s_j=M_i} c_j(s) \leq L_i(s) + \frac{\alpha B^2}{L_i(s)}.$$

Let $f(x) = x + \frac{\alpha B^2}{x}$. The function $f(x)$ decreases in x for every $x < \sqrt{\alpha}B$ and increases in x for every $x > \sqrt{\alpha}B$. Since for every i, $L_i(s) \leq 2(\sqrt{\alpha} - \alpha^2)B$ (by the construction of s) and $L_i(s) > (\sqrt{\alpha} - \alpha/2 - \alpha^2)B$ (by Claim 4.4), it suffices to bound the maximal cost at these two points. Since $OPT \geq 2\sqrt{\alpha}B$, it follows that:

$$PoS \leq \max\left(\frac{f(2(\sqrt{\alpha}-\alpha^2)B)}{2\sqrt{\alpha}B}, \frac{f((\sqrt{\alpha}-\alpha/2-\alpha^2)B)}{2\sqrt{\alpha}B}\right).$$

The assertion of the theorem follows by observing that for every $\alpha \leq 1/4$ it holds that $\frac{f(2(\sqrt{\alpha}-\alpha^2)B)}{2\sqrt{\alpha}B} < \frac{5}{4}$, and $\frac{f((\sqrt{\alpha}-\alpha/2-\alpha^2)B)}{2\sqrt{\alpha}B} \leq 1.1125 < \frac{5}{4}$.

□

We next show that the above upper bound is tight. That is, for every $\varepsilon > 0$, there exists an instance for which the ratio between the best NE and the social optimum is greater than $\frac{5}{4} - \varepsilon$. Interestingly, this ratio can be achieved with unit-length jobs.

Theorem 4.6. *For every $\varepsilon > 0$ there exists a resource allocation game with unit-length jobs for which $PoS > \frac{5}{4} - \varepsilon$.*

Proof. Let b be a sufficiently large integer (to be determined later) and let $B = b^2$. Consider an instance with $n = 2b - 2$ unit-length jobs. We claim that the game induced by this instance has a unique NE in which all the jobs are assigned to a single machine. Since $n < B$ it follows from Observation 2.6 that this is a NE. To prove uniqueness, assume towards contradiction that there is a NE schedule s in which the number of activated machines is at least 2, and let n_i denote the number of jobs assigned to machine M_i in s. Having unit-length jobs, the cost of a job that is scheduled on a machine with load k is $c(k) = k + B/k$. If $m = 2$ then for some $r > 0$, $n_1 = b - r$ and $n_2 = n - n_1 = b - 2 + r$. It is easy to verify that for any $r > 0$, $c(b-r) > c(b-1+r)$, thus, jobs on the first machine benefit from migrating to the second machine, contradicting the assumption that this is a NE. If $m > 2$ then since $n = 2b - 2$, there must exist at least two machines with at most $b - 1$ jobs, denote them i, i'. It is easy to verify that for any $0 < n_i \leq n_{i'} \leq b - 1$ it holds that $c(n_i) > c(n_{i'} + 1)$. Therefore, a unilateral migration from machine M_i to machine $M_{i'}$ is beneficial, in contradiction to s being a Nash equilibrium. It follows that scheduling all the jobs on a single machine is the unique NE. The cost of each job in this schedule is $c(2b-2) = 2b - 2 + \frac{B}{2b-2}$.

Consider next a schedule of this instance with two activated machines, each holding $b - 1$ jobs. The cost of each job in this schedule is $c(b-1) = b - 1 + \frac{B}{b-1}$. It follows that:

$$PoS \geq \frac{2b - 2 + B/(2b-2)}{b - 1 + B/(b-1)} = \frac{5b^2 - 8b + 4}{4b^2 - 4b + 2}.$$

One can easily verify that the last expression is greater than $\frac{5}{4} - \varepsilon$ for every $b > \frac{2}{\varepsilon}$. This establishes the assertion of the theorem. □

5 Conclusions and Open Problems

We study settings with conflicting congestion effects. While most of the literature focused on cases with either positive or negative congestion effects, we are interested to study equilibrium properties in settings that admit both effects simultaneously. While the induced game is not a congestion game [20], we find that it always admits a Nash equilibrium in pure strategies. This result motivated the evaluation of the inefficiency introduced due to selfish behavior, as quantified by the price of anarchy and price of stability measures, for which we provide tight bounds.

Our analysis and results suggest several intriguing open questions and future work. First of all, it is desirable to generalize our results to settings with different sharing rules, different cost structures, or different social choice functions. In addition, we believe that conflicting congestion effects should be studied in additional congestion models, such as network creation and routing games. Finally, it would be interesting to provide a characterization of instances for which best-response dynamics or some other types of dynamics are guaranteed to converge to a Nash equilibrium.

References

[1] Susanne Albers, Stefan Elits, Eyal Even-Dar, Yishay Mansour, and Liam Roditty. On Nash Equilibria for a Network Creation Game. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2006.

[2] Elliot Anshelevich, Anirban Dasgupta, Jon M. Kleinberg, Éva Tardos, Tom Wexler, and Tim Roughgarden. The price of stability for network design with fair cost allocation. In *FOCS*, pages 295–304, 2004.

[3] Yossi Bukchin and Eran Hanany. Decentralization Cost in Scheduling: A Game-Theoretic Approach. *MANUFACTURING SERVICE OPERATIONS MANAGEMENT*, 9(3):263–275, 2007.

[4] Roberto Cominetti, Jose R. Correa, and Nicolas E. Stier-Moses. The Impact of Oligopolistic Competition in Networks. *OPERATIONS RESEARCH*, page opre.1080.0653, 2009.

[5] Artur Czumaj and Berthold Vöcking. Tight bounds for worst-case equilibria. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 413–420, 2002.

[6] Amir Epstein, Michal Feldman, and Yishay Mansour. Strong Equilibrium in Cost Sharing Connection Games. In *ACM Conference on Electronic Commerce (ACMEC)*, 2007.

[7] E. Even-Dar and Y. Mansour. Fast convergence of selfish rerouting. In *Sixteenth ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 772–781, 2005.

[8] Alex Fabrikant, Ankur Luthra, Elitza Maneva, Christos Papadimitriou, and Scott Shenker. On a network creation game. In *ACM Symposium on Principles of Distriubted Computing (PODC)*, 2003.

[9] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the Cost of Multicast Transmissions. In *Journal of Computer and System Sciences*, volume 63, pages 21–41, 2001.

[10] D. Fotakis, M. Mavronicolas S. Kontogiannis, and P. Spiraklis. The Structure and Complexity of Nash Equilibria for a Selfish Routing Game. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 510–519, 2002.

[11] R.L. Graham. Bounds on multiprocessing timing anomalies. SIAM J. Appl. Math., 17:263–269, 1969.

[12] Shai Herzog, Scott Shenker, and Deborah Estrin. Sharing the "Cost" of Multicast Trees: An Axiomatic Analysis. In *IEEE/ACM Transactions on Networking*, 1997.

[13] Kamal Jain and Mohammad Mahdian. *Algorithmic Game Theory*, chapter Cost Sharing. Cambridge University Press, 2007.

[14] Elias Koutsoupias and Christos H. Papadimitriou. Worst-case equilibria. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 404–413, 1999.

[15] Dov Monderer and Llyod S. Shapley. Potential Games. *Games and Economic Behavior*, 14:124–143, 1996.

[16] Herve Moulin and Scott Shenker. Serial cost sharing. *Econometrica*, 60:1009–1037, 1992.

[17] Herve Moulin and Scott Shenker. Strategyproof sharing of submodular costs: Budget balance versus efficiency. *Journal of Economic Theory*, 18:511–533, 2001.

[18] Christos Papadimitriou. Algorithms, Games, and the Internet. In *Proceedings of 33rd STOC*, pages 749–753, 2001.

[19] Ali K. Parlakturk and Sunil Kumar. Self-Interested Routing in Queueing Networks. *MANAGEMENT SCIENCE*, 50(7):949–966, 2004.

[20] Robert W. Rosenthal. A Class of Games Possessing Pure-Strategy Nash Equilibria. *Internation Journal of Game Theory*, 2:65–67, 1973.

[21] T. Roughgarden and Eva Tardos. How bad is selfish routing? *Journal of the ACM*, 49(2):236 – 259, 2002.

[22] Tim Roughgarden. The Price of Anarchy is Independent of the Network Topology. In *STOC'02*, pages 428–437, 2002.

[23] Adrian Roshan Vetta. Nash equilibria in competitive societies with applications to facility location, traffic routing and auctions. In *Symposium on the Foundations of Computer Science (FOCS)*, pages 416–425, 2002.

[24] B. Vöcking. *In Noam Nisan, Tim Roughgarden, Eva Tardos and Vijay Vazirani, eds., Algorithmic Game Theory. Chapter 20: Selfish Load Balancing.* Cambridge University Press, 2007.

A Equilibrium Existence

We show a particular profile which is a pure Nash equilibrium. We first define a complete order on the profiles.

Definition A.1. A vector $(L_1, L_2, \dots L_m)$ is smaller than $(\hat{L}_1, \hat{L}_2, \dots \hat{L}_m)$ *lexicographically if for some i, $L_i < \hat{L}_i$ and $L_k = \hat{L}_k$ for all $k < i$. A profile s is smaller than s' lexicographically if the vector of machine loads $L(s) = (L_1(s), \dots, L_m(s))$, sorted in non- increasing order, is smaller lexicographically than $L(s')$, sorted in non-increasing order.*

We use the following property of the lexicographically minimal assignment.

Claim A.2. *Let s be a lexicographically minimal assignment, and suppose that job j is assigned to machine M_i. Then, $L_{i'}(s) + p_j \geq L_i(s)$ for every $i' \neq i$.*

Proof. If $L_{i'}(s) \geq L_i(s)$, the statement holds trivially. Otherwise (i.e., $L_{i'}(s) < L_i(s)$), suppose by way of contradiction that $L_{i'}(s) + p_j < L_i(s)$, then, the schedule that assigns the job of length p_j to the machine of load $L_{i'}(s)$ produces a lexicographically-smaller assignment, in contradiction to the minimality of s. □

Given an instance of jobs I, Let $I_{short} \subseteq I$ be the subset of jobs having length less than B. 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. Note that m is well defined, since all the participating jobs are shorter than B.

Let $\hat{s}$ be the profile in which: (i) every j s.t. $p_j \geq B$ is assigned to a dedicated machine, and (ii) The jobs of I_{short} are assigned according to $\hat{s}_m$.

Theorem A.3. *The profile $\hat{s}$ is a NE.*

Proof. We show that none of the possible migrations is beneficial. By Observation 2.3, no long job can benefit from migration, and by Observation 2.5(i) no short job can benefit from joining a long job or from activating a new machine. It remains to show that no short job can benefit from migrating to another machine of $\hat{s}_m$. Let j be a short job assigned to M_i. By Claim A.2, $L_{i'}(\hat{s}) + p_j \geq L_i(\hat{s})$ for every $i' \neq i$. In other words, only load-increasing migration can be performed. By Claim 2.2, such migrations are beneficial for job j if and only if $B > \frac{L_i(\hat{s})(L_{i'}(\hat{s})+p_j)}{p_j}$. Assume $p_j = \alpha L_i(\hat{s})$ for some $0 < \alpha \leq 1$. Then the migration condition can be written as $B > \frac{L_i(\hat{s})(L_{i'}(\hat{s})+\alpha L_i(\hat{s}))}{\alpha L_i(\hat{s})} = \frac{L_{i'}(\hat{s})}{\alpha} + L_i(\hat{s}) \geq L_{i'}(\hat{s}) + L_i(\hat{s})$. Therefore, to obtain a contradiction, it is sufficient to show that for every i, i', $L_i(\hat{s}) + L_{i'}(\hat{s}) \geq B$. Assume by way of contradiction that there exist i, i', s.t. $L_i(\hat{s}) + L_{i'}(\hat{s}) < B$. Consider the schedule which is identical to $\hat{s}$ except the jobs assigned to machines i and i' are jointly assigned to a single machine. This schedule produces a makespan smaller than B on $m - 1$ machines, which is a contradiction to the choice of m. □