**The Open University of Israel**
**Department of Mathematics and Computer Science**

# Positive and Negative Congestion Effects in Resource Allocation Games

Final paper submitted as partial fulfillment of the requirements
towards an M.Sc. degree in Computer Science
The Open University of Israel
Computer Science Division

by
**Ilia Katz**

Prepared under the supervision of **Dr. Leah Epstein**

August 2010

# Contents

# List of Figures

# Acknowledgments

I am gratefully thankful to my advisor, Dr. Leah Epstein. It was Leah who introduced me to game theory. Her encouragement, guidance and support from the preliminary to the concluding level enabled me to develop an understanding of the subject and made this work possible. I am especially thankful to Leah for her readiness to supervise my work as an external advisor.

Special thanks to my wife and parents, for their help and support during the entire time of my graduate studies.

## Abstract

Most of the previous work on resource allocation problems, studied from the perspective of algorithmic game theory, has considered cost structures with either negative congestion effects (e.g., job scheduling and network routing games), where a job prefers to be assigned to a lightly loaded resource, or positive congestion effects (e.g., network design and bin packing games), where a job prefers to share a resource with many jobs. In practice, both effects take place simultaneously.

We study resource allocation games with a cost function that encompasses both the positive and the negative congestion effects. We present a model, previously studied by Feldman and Tamir [8], under which both effects take place simultaneously. We show that under this model, Strictly Pareto optimal Nash Equilibria always exist for game instances with number of players smaller than or equal to three. In contrast, we show that for game instances with number of players greater than three, Pareto Optimal Nash Equilibria might not exist. We improve the lower bound on the Price of Stability (PoS) of [8]. Finally, we introduce a new model that is based on the aforementioned model and show that under the new model, even though a Nash Equilibrium must exist, $PoS$ is unbounded.

# 1 Introduction

## 1.1 Background and Motivation

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. The computer science literature has traditionally treated these problems as combinatorial optimization problems. In the last decade, algorithmic game theory has introduced game theoretic considerations to many of these problems [2, 1, 11, 12]. 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. 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 each job chooses a resource instead of being assigned to a resource by a central designer. 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 assumptions on the cost functions lie at the heart 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 [9] and selfish routing belong [13, 7] 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, a user usually wishes to share its resource with as many 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. Network design games [1, 4] and bin packing games [5] belong to the second class of models.

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.

## 1.2 Existing model

A game instance $G = \langle I, B \rangle$, consists of a set of $n$ jobs, each job has a length $p_j$ (processing time, bandwidth requirement, etc.) associated with it. Let $I = \{p_1, \ldots, p_n\}$ denote the jobs lengths (also called weights). A set of identical resources $M = \{M_1, M_2, \ldots\}$ (machines, links, etc.) is given as well, each having an activation cost $B$. We use the terminology of job scheduling for simplicity of presentation though the model is general and can be used in a vast variety of games.

The action space $S_j$ of a 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 $j$ select 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 lengths of the players that chose machine $M_i$. In particular, a player can choose to be assigned to 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. 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 of $B$. In [8], $c_j(s)$ is assumed to be equal to $L_i(s) + b_j(s)$.

The resources activation cost may be shared among its users according to different sharing rules, two of which are considered in [8]. The sharing rules are: the *uniform* sharing rule and the *proportional* sharing rule.

**Uniform sharing rule:** Under the uniform sharing rule, all jobs assigned to a particular resource share its activation cost equally. Formally, a job $j$ assigned to $M_i$ under joint action $s$ pays $b_j(s) = \dfrac{B}{|R_i^s|}$. For example, let $G = \langle I = \{1, 2\}, B = 12 \rangle$ and let $s$ be an assignment in which all jobs are assigned to the same machine. Then, under the uniform sharing rule the jobs' costs will be as follows: $c_1(s) = c_2(s) = 3 + 12/2 = 9$.

**Proportional sharing rule:** Under the proportional rule, all jobs assigned to a particular resource share its activation cost proportionally to their sizes. Formally, a job $j$ assigned to $M_i$ under joint action $s$ pays $b_j(s) = \dfrac{p_j B}{L_i(s)}$. Under the proportional sharing rule, the jobs from the previous example will now pay $c_1(s) = 3+1\cdot 12/3 = 7$ and $c_2(s) = 3+2\cdot 12/3 = 11$.

It was shown in [8] that under the *uniform sharing rule* equilibria might not exist. Therefore, in the subsequent sections we focus on the *proportional sharing rule.*

## 1.3 Definitions

**Definition 1.** *(Nash Equilibrium) A joint action is said to be a pure Nash Equilibrium (NE) if no player can benefit from unilaterally switching from his action to a different action. Formally, a joint action $s \in S$ is said to be pure Nash Equilibrium if for all players $i$, such that $i \in N$ and each alternative joint action $\bar{s} \in S$*

$$c_i(\bar{s}) \geq c_i(s)$$

**Definition 2.** *(Price of Stability) The Price of Stability (PoS) is a measure of inefficiency designed to differentiate between games in which all equilibria are inefficient and those in which some equilibrium is inefficient. Formally, the Price of Stability of a game instance is the ratio between the value of best Nash Equilibrium and the value of optimal solution.*

$$PoS = \frac{\textit{value of best Nash Equilibrium}}{\textit{value of optimal solution}}$$

**Definition 3.** *(Weakly Pareto optimal Nash Equilibrium) A Nash Equilibrium joint action is said to be Weakly Pareto optimal Nash Equilibrium (wPNE) if there is no joint action in the entire action space in which every player is strictly better off than he was as a result of the current joint action. Formally, a Nash Equilibrium $s \in S$ is said to be Weakly Pareto optimal Nash Equilibrium if there is no joint action $\bar{s} \in S$ such that for all $i \in N$*

$$c_i(\bar{s}) < c_i(s)$$

**Definition 4.** *(Strictly Pareto optimal Nash Equilibrium) A Nash Equilibrium joint action is said to be Strictly Pareto optimal Nash Equilibrium (sPNE) if there is no other joint action in the entire action space in which at least one player is strictly better off and no player is worse off than he was as a result of the current joint action. Formally, a Nash Equilibrium $s \in S$ is said to be Strictly Pareto optimal Nash Equilibrium if there is no joint action $\bar{s} \in S$ and $i^* \in N$ such that for all $i \in N \setminus i^*$*

$$c_i(\bar{s}) \leq c_i(s) \quad \textit{and} \quad c_{i^*}(\bar{s}) < c_{i^*}(s)$$

A Weakly Pareto optimal Nash Equilibrium satisfies a less stringent requirement than Strictly Pareto optimal Nash Equilibrium, thus every Strictly Pareto optimal Nash Equilibrium is also a Weakly Pareto optimal Nash Equilibrium. The opposite is not true, since a Weakly Pareto optimal Nash Equilibrium is not necessarily Strictly Pareto optimal Nash Equilibrium.

**Definition 5.** *A vector $(L_1, L_2, \ldots, L_m)$ is smaller than vector $(\bar{L}_1, \bar{L}_2, \ldots, \bar{L}_m)$ lexicographically if for some $i$, $L_i < \bar{L}_i$ and $L_k = \bar{L}_k$ for all $k < i$. A schedule $s$ is smaller than schedule $s'$ lexicographically if the vector of machine loads $L(s) = (L_1(s), L_2(s), \ldots, L_m(s))$ sorted in non-increasing order, is smaller lexicographically than $L(s')$, sorted in non-increasing order.*

## 1.4 Previous work

Models which emphasize either the negative congestion effect ([9, 13, 7]) or the positive congestion effect ([1, 4, 5]), but not both, are being utilized in a vast variety of games (job scheduling, selfish routing, network design, bin packing, etc.). In practice, in many of this games, both the positive and the negative effects take place simultaneously.

The model in which both effects take place simultaneously is thoroughly examined in [8]. In the remainder of this section we refer to the aspects studied in [8]. Two different sharing rules are introduces in [8] according to which the resource activation cost is shared among its users: uniform sharing rule and proportional sharing rule. It is shown that in the uniform sharing rule model, a pure *NE* might not exist, while in the proportional sharing model, a pure *NE* always exists. The *NE* achieved in the proportional sharing rule model may not be socially optimal. Its inefficiency is quantified in [8] according to the well-established measurements, the Price of Anarchy (*PoA*) and the Price of Stability (*PoS*). It is shown that the *PoA* is not bounded, while the *PoS* is bounded by an upper bound of 5/4 and by a lower bound of 18/17.

## 1.5 New results

In our work all the new results refer to resource allocation games under the proportional sharing rule and unlimited supply of resources. In Section 2.1 we show that Strictly Pareto optimal Nash Equilibrium [10, 6, 3] exists for games with number of players smaller than or equal to three. In Section 2.2 we show that neither Strictly nor Weakly Pareto Optimal Nash Equilibria exist for games with number of player greater than three. In Section 2.3 we provide an improved lower bound on the $PoS$ over the bound presented in Theorem 10 in [8]. Improved lower bound was found with the assistance of a program we developed (the program is presented in Appendix A). In Section 3 we introduce a new model that is based on the model described in [8] and show that for this model $PoS$ is unbounded. In addition, we prove a number of observations, some of which are new and some are adapted versions of observations from [8].

## 1.6 Preliminaries

**Lemma 6** (Lemma 1 from [8])**.** *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 $j$ under schedule $s$ is $c_j(s) = L_i(s) + \dfrac{p_j B}{L_i(s)}$. Let $s'$ be the obtained schedule after $j$'s migration to machine $M_{i'}$. It holds that $c_j(s') = L_{i'}(s) + p_j + \dfrac{p_j B}{L_{i'}(s) + p_j}$. Assume that job $j$ reduces its cost by a migration to machine $i'$ and prove that $L_{i'}(s) + p_j > L_i(s)$ and $B > \rho$ or $L_{i'}(s) + p_j < L_i(s)$ and $B < \rho$.

$$L_i(s) + \frac{p_j B}{L_i(s)} > L_{i'}(s) + p_j + \frac{p_j B}{L_{i'}(s) + p_j}$$

Multiplying both sides by $\rho$ yields

$$\rho L_i(s) + B(L_{i'}(s) + p_j) > \rho(L_{i'}(s) + p_j) + BL_i(s)$$

Rearranging yields

$$\begin{aligned}
\rho L_i(s) - \rho(L_{i'}(s) + p_j) + B(L_{i'}(s) + p_j) - BL_i(s) &> 0 \\
\rho(L_i(s) - (L_{i'}(s) + p_j)) - B(L_i(s) - (L_{i'}(s) + p_j)) &> 0 \\
(L_i(s) - (L_{i'}(s) + p_j))(\rho - B) &> 0
\end{aligned}$$

The inequity holds when $L_i(s) > (L_{i'}(s) + p_j)$ and $\rho > B$ or when $L_i(s) < (L_{i'}(s) + p_j)$ and $\rho < B$. One can easily verify that the opposite direction (assume that $L_{i'}(s) + p_j > L_i(s)$ and $B > \rho$ or $L_{i'}(s) + p_j < L_i(s)$ and $B < \rho$ and prove that job $j$ reduces its cost by a migration to machine $i'$) holds as well using the same inequity. $\square$

**Observation 7** (Observation 1 from [8])**.** *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 (possibly empty) set of jobs with total length $z$ is $c_j(s) = p_j + z + \frac{p_j B}{p_j + z}$. This term gets its minimal value for $z = \sqrt{p_j B} - p_j$, for which $c_j(s) = 2\sqrt{p_j B}$. Hence, for every job $j$,$c_j(s) \geq 2\sqrt{p_j B}$. If $p_j \geq B$ then the cost is minimized for $z = 0$, for which, $c_j(s) = p_j + B$. Hence for every $j$, such that $p_j \geq B$, $c_j(s) \geq p_j + B$. $\square$

**Observation 8.** *(i) A job $j$ of length $p_j < B$ which is assigned to a machine of load smaller than $B$ together with a (possibly empty) set of jobs, will not decrease its cost by migrating to a machine of load greater than or equal to $B - p_j > 0$. (ii) A job $j$ of length $p_j < B$ which is assigned to a machine of load smaller than $B$ together with a non empty set of jobs will increase its cost by migrating to a dedicated machine.*

*Proof.* Suppose that $v$ is the total length of a (possible empty) set of jobs assigned together with job $j$ to machine with load smaller than $B$. $0 \leq v$ and $v + p_j < B$, thus $0 \leq v < B - p_j$. The cost of job $j$ before migration is $c_j(s) = p_j + v + \frac{p_j B}{p_j + v}$. Suppose that $z$ is the total length of non empty set of jobs assigned to machine with load greater than or equal to $B - p_j$ (the machine $j$ prefers to migrate to). Thus, $z \geq B - p_j > 0$. The cost of job $j$ after migration is $c_j(s') = p_j + z + \frac{p_j B}{p_j + z}$. $v < B - p_j$ and $z \geq B - p_j$ resulting in $z > v$. We show that $c_j(s') \geq c_j(s)$.

$$p_j + z + \frac{p_j B}{p_j + z} \geq p_j + v + \frac{p_j B}{p_j + v}$$

Multiplying both sides by $(p_j + z)(p_j + v)$ yields

$$\begin{aligned} z(p_j + z)(p_j + v) + p_j B(p_j + v) &\geq v(p_j + z)(p_j + v) + p_j B(p_j + z) \\ {p_j}^2 z + p_j z^2 + p_j v z + v z^2 + p_j v B &\geq {p_j}^2 v + p_j v^2 + p_j v z + v^2 z + p_j z B \end{aligned}$$

Rearranging yields

$$\begin{aligned}
{p_j}^2 z - {p_j}^2 v + p_j z^2 - p_j v^2 + v z^2 - v^2 z + p_j v B - p_j z B &\geq 0 \\
{p_j}^2 (z - v) + p_j (z^2 - v^2) + vz(z - v) - p_j B(z - v) &\geq 0 \\
{p_j}^2 (z - v) + p_j (z - v)(z + v) + vz(z - v) - p_j B(z - v) &\geq 0 \\
(z - v)(p_j^2 + p_j(z + v) + vz - p_j B) &\geq 0
\end{aligned}$$

We showed that $z > v$, thus we only need to prove that $p_j^2 + p_j(z + v) + vz - p_j B \geq 0$

$$\begin{aligned}
p_j^2 + p_j(z + v) + vz - p_j B &\geq 0 \\
p_j(p_j + z + v - B) + vz &\geq 0
\end{aligned}$$

By definition $z \geq B - p_j$, hence $z + p_j \geq B$. In addition $v \geq 0$ and $z > 0$, resulting in $p_j + z + v - B \geq 0$ and $p_j(p_j + z + v - B) + vz \geq 0$.

(ii) Let $L = \alpha B$ be the load of the machine to which $p_j$ is assigned, for $0 < \alpha < 1$. The cost of $j$ when assigned to a machine of load smaller than $B$ together with a non empty set of jobs is $c_j(s) = L + \frac{p_j B}{L} = \alpha B + \frac{p_j B}{\alpha B}$. The cost of $j$ when assigned to a dedicated machine is $c_j(s') = p_j + B$. We show that $c_j(s') > c_j(s)$.

$$\begin{aligned}
p_j + B &> \alpha B + \frac{p_j B}{\alpha B} \\
p_j + B &> \alpha B + \frac{p_j}{\alpha}
\end{aligned}$$

Multiplying both sides by $\alpha$ yields

$$\begin{aligned}
\alpha p_j + \alpha B &> \alpha^2 B + p_j \\
\alpha p_j - p_j &> \alpha^2 B - \alpha B \\
p_j(\alpha - 1) &> \alpha B(\alpha - 1)
\end{aligned}$$

Multiplying both sides by $\left(\dfrac{1}{\alpha - 1}\right)$ yields

$$p_j < \alpha B$$

By definition $L$ is equal to $\alpha B$, hence we obtain

$$p_j < L$$

$\square$

**Observation 9.** *Given an assignment $s$ such that job $j$ is assigned to machine $M_i$ and for every $i'$, such that $i' \neq i$, $L_{i'}(s) + p_j \geq L_i(s)$, if $L_i(s) + L_{i'}(s) \geq B$ then a migration of job $j$ from machine $i$ to machine $i'$ is not beneficial.*

*Proof.* First we show that the Observation holds for $L_{i'}(s) + p_j > L_i(s)$. Assume that $L_i(s) + L_{i'}(s) \geq B$. By Lemma 6, it is beneficial for $j$ to migrate to machine $i'$ if and only if $B > \rho = \frac{L_i(s)(L_{i'}(s)+p_j)}{p_j}$. Job $j$ is assigned to machine $i$, hence $p_j = \alpha L_i(s)$ for some $0 < \alpha \leq 1$. The migration condition can be rewritten as $B > \frac{L_i(s)(L_{i'}(s)+\alpha L_i(s))}{\alpha L_i(s)} = \frac{L_{i'}(s)}{\alpha} + L_i(s) \geq L_{i'}(s) + L_i(s)$. It is a contradiction to the assumption that $L_i(s) + L_{i'}(s) \geq B$.

Next we show that the Observation holds for $L_{i'}(s) + p_j = L_i(s)$. The cost of job $j$ under schedule $s$ is $c_j(s) = L_i(s) + \frac{p_j B}{L_i(s)}$. Let $s'$ denote the schedule obtained after $j$'s migration to machine $i'$. The cost of job $j$ under schedule $s'$ is $c_j(s') = L_{i'}(s) + p_j + \frac{p_j B}{L_{i'}(s)+p_j} = L_i(s) + \frac{p_j B}{L_i(s)}$. The cost of job $j$ before the migration is identical to the cost after the migration ($c_j(s) = c_j(s')$), therefore the migration is not beneficial. □

**Observation 10.** *If $B \geq \sum_j p_j$, then the schedule $s$ in which all jobs are assigned to a single machine is a NE.*

*Proof.* All jobs are assigned to a single machine, therefore an arbitrary job $j$ can migrate to a dedicated machine only. We show that a migration is not beneficial and therefore schedule $s$ is a *NE*. If a game instance consists of a single job, this job cannot benefit from migration (from a dedicated machine) to a dedicated machine, therefore the Observation holds trivially. Otherwise, the game instance consists of at least two jobs. First, we show that the Observation holds for $B > \sum_j p_j$. Assume that $B > \sum_j p_j$. By definition $B > p_j$ for every $j$. By Observation 8(ii), a job $j$ of length $p_j < B$ which is assigned to a machine of load smaller than $B$ will increase its cost by migrating to a dedicated machine.

Next we now show that the Observation holds for $B = \sum_j p_j$. Assume that $B = \sum_j p_j$. By definition $B \geq p_j$. The cost of job $j$ under schedule $s$ is $c_j(s) = B + \dfrac{p_j B}{B} = B + p_j$. Let $s'$ denote the schedule obtained after $j$'s migration to a dedicated machine. The cost of job $j$ under schedule $s'$ is $c_j(s') = p_j + B$ and is equal to the cost before the migration, hence the migration is not beneficial. □

In the subsequent sections, job's lengths are assumed to be positive in all the game instances we present.

# 2 Equilibria existence

In the following section we would like to analyze Equalibria existence under the proportional sharing rule.

## 2.1 Existence of Strictly Pareto optimal Nash Equilibria

In this section we prove that under the proportional sharing rule and unlimited supply of resources a *sPNE* always exists for game instances with number of players smaller than or equal to three. In the Preliminaries section, we showed that every *sPNE* is also a *wPNE*, thus existence of *sPNE* results in existence of *wPNE*. We prove that *sPNE* always exists by presenting a particular joint action which is a *sPNE*.

**Claim 11** (Claim 2 from [8])**.** *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 $L_{i'}(s) \geq L_i(s)$, then $L_{i'}(s) + p_j \geq L_i(s)$ (the statement holds trivially). Otherwise, $L_{i'}(s) < L_i(s)$. Suppose by way of contradiction that $L_{i'}(s) + p_j < L_i(s)$, then the schedule that assigns the job $j$ to machine of load $L_{i'}(s)$ produces an assignment which is lexicographically smaller than assignment $s$, in contradiction to minimality of $s$. □

Given an instance of jobs $I$, let $I_{short} \subseteq I$ be a subset of jobs having length smaller than $B$. Let $s_k$ be the lexicographically minimal assignment of $I_{short}$ on $k$ machines. Let $m$ be the minimal integer such that the makespan under $s_m$ is smaller than $B$. Note that $m$ is well defined, since all participating jobs are smaller than $B$.
Let $s$ be the schedule in which: (i) Every $j$ such that $p_j \geq B$ is assigned to a dedicated machine (ii) The jobs of $I_{short}$ are assigned according to $s_m$.

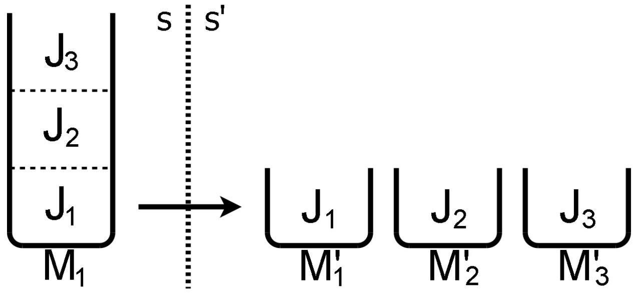
**Theorem 12** (Theorem 3 from [8])**.** *Schedule $s$ is a NE.*

*Proof.* If $m = 1$, then all jobs are jointly assigned to a single machine of load smaller than $B$. By Observation 10 schedule $s$ is a *NE*. Otherwise, $m \geq 2$. We prove that schedule $s$ is *NE* by showing that none of the possible migrations is beneficial. By Observation 7, no long job (a job $j$ such that $p_j \geq B$) can benefit from migration. By Observation 8(i), no short job (a job $j$ such that $p_j < B$) can benefit from joining a long job. By Observation 8(ii), no short job can benefit from activating a new machine. It remains to show that no short job can benefit from migrating to another machine of $s$ of load smaller than $B$. Let $j$ be a short job assigned to machine $M_i$. By Claim 11, $L_{i'}(s) + p_j \geq L_i(s)$ for every $i'$, such that $i' \neq i$. Assume by way of contradiction that there exist $i'$, such that $i' \neq i$, for which $L_i(s) + L_{i'}(s) < B$. Consider the schedule which is identical to $s$ 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. It is a contradiction to the choice of $m$. Thus, $L_i(s) + L_{i'}(s) \geq B$ for every $i'$, such that $i \neq i'$. We already showed that $L_{i'}(s) + p_j \geq L_i(s)$ for every $i'$, such that $i' \neq i$. Therefore, by Observation 9, migration of job $j$ from machine $i$ to any other machine $i'$ of $s$ is not beneficial. □

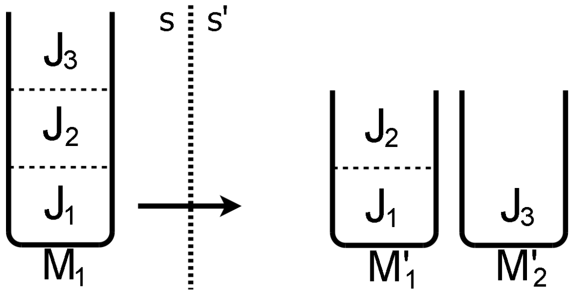
We now show that schedule $s$ is a *sPNE*. In schedule $s$, for every long job $j$, $c_j(s) = p_j + B$. By Observation 7, for job $j$ such that $p_j \geq B$, $c_j(s) \geq p_j + B$. Hence, for every schedule $\bar{s}$, such that $\bar{s} \neq s$, $c_j(\bar{s}) \geq c_j(s)$. Therefore, we focus on games with jobs of length smaller than $B$. We prove that schedule $s$ is a *sPNE* by showing that there is no schedule $\bar{s}$, such that $\bar{s} \neq s$, in which at least one player is strictly better off and no player is worse off.

We start with a game instance with three players. Consider schedule $\bar{s}$, such that $\bar{s} \neq s$. Distinguish between three cases:

**1.** Under schedule $s$, all jobs are assigned to a single machine of load smaller than $B$ (figure 1). Under schedule $\bar{s}$ (Figure 1a, Figure 1b), at least one of the jobs must be assigned to a dedicated machine. We denote this job by $j$. By Observation 8(ii), $c_j(\bar{s}) > c_j(s)$. Thus, schedule $\bar{s}$ does not contradict the claim that schedule $s$ is a *sPNE*.



(a) Jobs are assigned to three machines in $\bar{s}$



(b) Jobs are assigned to two machines in $\bar{s}$

Figure 1: All jobs are assigned to single machine in $s$

**2.** Under schedule $s$, jobs are assigned to two machines, each of load smaller than $B$ (figure 2). Assume without loss of generality that jobs $J_1$ and $J_2$ are assigned to machine $M_1$ and job $J_3$ is assigned to machine $M_2$. We distinguish between three sub cases:

a. Under schedule $\bar{s}$ (Figure 2a), all jobs are assigned to dedicated machines (particularly job $J_1$ and job $J_2$). By Observation 8(ii), $c_{J_1}(\bar{s}) > c_{J_1}(s)$ and $c_{J_2}(\bar{s}) > c_{J_2}(s)$. Thus, schedule $\bar{s}$ does not contradict the claim that schedule $s$ is a *sPNE*.

b. Under schedule $\bar{s}$ (Figure 2b), the number of machines remains unchanged. One of the jobs that are assigned together in schedule $s$ (job $J_1$ or job $J_2$) is assigned to a dedicated machine in schedule $\bar{s}$. Assume without loss of generality that under schedule $\bar{s}$, job $J_2$ is assigned to a dedicated machine. By Observation 8(ii), $c_{J_2}(\bar{s}) > c_{J_2}(s)$. Thus, schedule $\bar{s}$ does not contradict the claim that schedule $s$ is a *sPNE*.

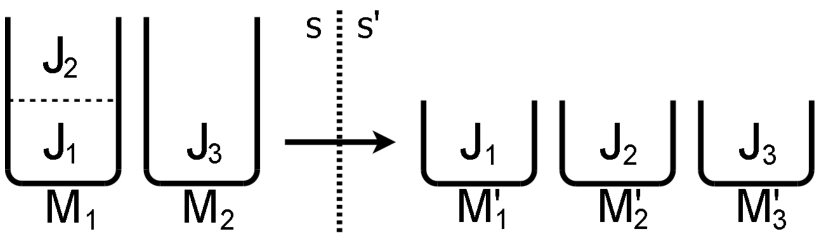
c. Under schedule $\bar{s}$ (Figure 2c), jobs $J_1$, $J_2$ and $J_3$ are assigned together to a single machine $M_1'$. If the load of the machine is smaller than $B$ it is a contradiction to the choice of $m$. We now examine the case where the load of $M_1'$ is equal to or greater than $B$. We assumed that under schedule $s$, jobs $J_1$ and $J_2$ are assigned together (to machine $M_1$) and job $J_3$ is assigned to a dedicated machine (to machine $M_2$). Let $z$ denote $p_{J_1} + p_{J_2}$. Under schedule $s$, job $J_1$ pays $c_{J_1}(s) = z + (p_{J_1}B)/z$. Under schedule $\bar{s}$, job $J_1$ pays $c_{J_1}(\bar{s}) = z + p_{J_3} + p_{J_1}B/(z + p_{J_3})$. We now show that $c_{J_1}(s) < c_{J_1}(\bar{s})$.

$$z + \frac{p_{J_1}B}{z} < z + p_{J_3} + \frac{p_{J_1}B}{z + p_{J_3}}$$
$$\frac{p_{J_1}B}{z} < p_{J_3} + \frac{p_{J_1}B}{z + p_{J_3}}$$

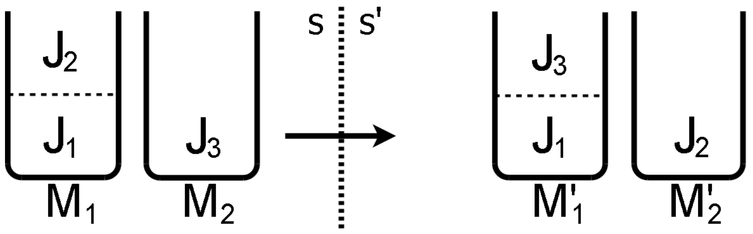
Multiplying both sides by $z(z + p_{J_3})$ yields

$$zp_{J_1}B + p_{J_1}p_{J_3}B < z(z + p_{J_3})p_{J_3} + zp_{J_1}B$$
$$p_{J_1}p_{J_3}B < z(z + p_{J_3})p_{J_3}$$
$$p_{J_1}B < z(z + p_{J_3})$$

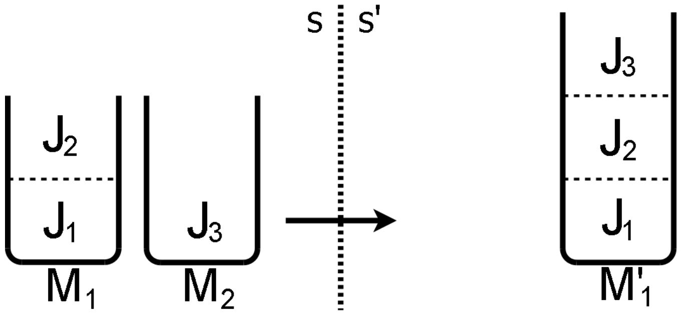
We assumed that $p_{J_1} + p_{J_2} + p_{J_3} \geq B$, i.e, $z + p_{J_3} \geq B$. In addition, $z = p_{J_1} + p_{J_2} > p_{J_1}$. Hence, the inequality holds. We showed that schedule $\bar{s}$ does not contradict the claim that schedule $s$ is a *sPNE*.



(a) Jobs are assigned to three machines in $\bar{s}$



(b) Jobs are assigned to two machines in $\bar{s}$



(c) Jobs are assigned to single machine in $\bar{s}$

Figure 2: Jobs are assigned to two machines in $s$

**3.** Jobs are assigned to three (dedicated) machines each of load smaller than $B$ in $s$ (Figure 3). We distinguish between two sub cases:

a. In schedule $\bar{s}$ (Figure 3a) jobs are assigned to two machines. Assume without loss of generality that under schedule $\bar{s}$, jobs $J_1$ and $J_2$ are assigned to a single machine (to machine $M_1'$) and job $J_3$ is assigned to another, dedicated machine (to machine $M_2'$). If the makespan under $\bar{s}$ is smaller than $B$, it is a contradiction to the choice of $m$. Consider that the makespan under $\bar{s}$ is equal to or greater than $B$. The load of machine $M_2'$ is smaller than $B$ because only job $J_3$ (which length is smaller than $B$) is assigned to it. The makespan under $\bar{s}$ is equal to or greater than $B$ therefore the load of machine $M_1'$ must be equal to or greater than $B$. Under schedule $s$, job $J_1$ pays $c_{J_1}(s) = p_{J_1} + B$. Under schedule $\bar{s}$, job $J_1$ is assigned together with job $J_2$ and pays $c_{J_1}(\bar{s}) = p_{J_1} + p_{J_2} + (p_{J_1}B)/(p_{J_1} + p_{J_2})$.

We now show that $c_{J_1}(s) \leq c_{J_1}(\bar{s})$.

$$p_{J_1} + B \leq p_{J_1} + p_{J_2} + p_{J_1}B/(p_{J_1} + p_{J_2})$$
$$B \leq p_{J_2} + p_{J_1}B/(p_{J_1} + p_{J_2})$$

Multiplying both sides by $(p_{J_1} + p_{J_2})$ yields

$$p_{J_1}B + p_{J_2}B \leq (p_{J_1} + p_{J_2})p_{J_2} + p_{J_1}B$$
$$p_{J_2}B \leq (p_{J_1} + p_{J_2})p_{J_2}$$
$$B \leq p_{J_1} + p_{J_2}$$

We assumed that $p_{J_1} + p_{J_2} \geq B$, hence, the inequality holds. It can be shown in a similar way that $c_{J_2}(s) \leq c_{J_2}(\bar{s})$. Job $J_3$ is assigned to a dedicated machine under schedule $\bar{s}$, which means that its cost remains unchanged. We showed that under schedule $\bar{s}$, there is no job that is strictly better off, thus schedule $\bar{s}$ does not contradict the claim that schedule $s$ is a *sPNE*.

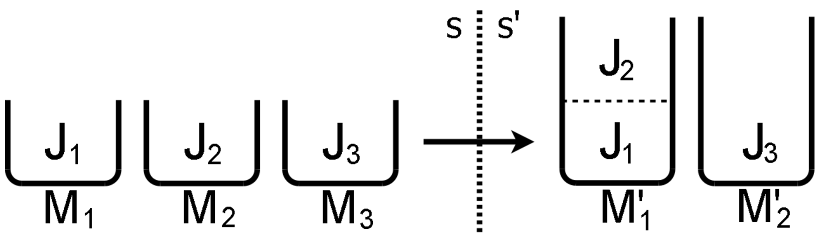
b. In schedule $\bar{s}$ (Figure 3b) jobs $J_1$, $J_2$ and $J_3$ are assigned together to a single machine $M'_1$. If the load of the machine is smaller than $B$ it is a contradiction to the choice of $m$. Consider that the load of the machine is equal to or greater than $B$. Under schedule $s$, job $J_1$ pays $c_{J_1}(s) = p_{J_1} + B$. Under schedule $\bar{s}$, all jobs are jointly assigned to a single machine and job $J_1$ pays $c_{J_1}(\bar{s}) = p_{J_1} + p_{J_2} + p_{J_3} + p_{J_1}B/(p_{J_1} + p_{J_2} + p_{J_3})$. We now show that $c_{J_1}(s) \leq c_{J_1}(\bar{s})$.

$$p_{J_1} + B \leq p_{J_1} + p_{J_2} + p_{J_3} + p_{J_1}B/(p_{J_1} + p_{J_2} + p_{J_3})$$
$$B \leq p_{J_2} + p_{J_3} + p_{J_1}B/(p_{J_1} + p_{J_2} + p_{J_3})$$

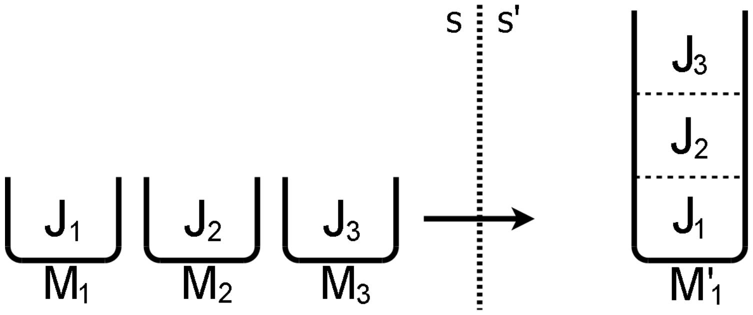
Multiplying both sides by $(p_{J_1} + p_{J_2} + p_{J_3})$ yields

$$p_{J_1}B + p_{J_2}B + p_{J_3}B \leq p_{J_2}(p_{J_1} + p_{J_2} + p_{J_3}) + p_{J_3}(p_{J_1} + p_{J_2} + p_{J_3}) + p_{J_1}B$$
$$p_{J_2}B + p_{J_3}B \leq p_{J_2}(p_{J_1} + p_{J_2} + p_{J_3}) + p_{J_3}(p_{J_1} + p_{J_2} + p_{J_3})$$

We assumed that $p_{J_1} + p_{J_2} + p_{J_3} \geq B$, hence, the inequality holds. It can be shown in a similar way that $c_{J_2}(s) \leq c_{J_2}(\bar{s})$ and $c_{J_3}(s) \leq c_{J_3}(\bar{s})$. We showed that under schedule $\bar{s}$, there is no job that is strictly better off, thus schedule $\bar{s}$ does not contradict the claim that schedule $s$ is a *sPNE*.



(a) Jobs are assigned to two machines in $\bar{s}$



(b) Jobs are assigned to single machine in $\bar{s}$

Figure 3: Jobs are assigned to three machines in $s$

If a game instance with three players consists of long jobs only, then the claim that schedule $s$ is a *sPNE* holds trivially. Otherwise, short jobs participate in the game as well. For such a game instance, we showed that there is no schedule $\bar{s}$, such that $\bar{s} \neq s$, in which at least one player is strictly better off and no player is worse off. Therefore, schedule $s$ is a *sPNE*. It remains to show that in a game instance with two players, schedule $s$ is a *sPNE* as well. In a game instance with two players, under schedule $s$, jobs are either jointly assigned to a single machine or separately assigned to two dedicated machines. Consider the schedule in which jobs are assigned to a single machine. The proof that this schedule is a *sPNE* is very similar to the proof for three players that are jointly assigned to single machine in schedule $s$ (**Case 1**). Consider the schedule in which jobs are assigned separately to two dedicated machines. The proof that this schedule is a *sPNE* is very similar to the proof for three players that are assigned separately to dedicated machines in schedule $s$ (**Case 3**). In a game instance with single player there is a unique schedule and there is nothing to prove.

## 2.2 Nonexistence of Pareto optimal Nash Equilibria

In this section we prove that under the proportional sharing rule and unlimited supply of resources, Pareto Optimal Nash Equilibria (neither strict nor weak) may not exist for games with number of players greater than three. We show this by introducing a game instance with number of player greater than three and for which weak Pareto Optimal Nash Equilibria do not exist (Nonexistence of weak Pareto Optimal Nash Equilibria implies nonexistence of strict Pareto Optimal Nash Equilibria).

Consider a game instance $G$ with one long job and a number of short jobs (at least 3 short jobs). We denote the long job by $X$, an arbitrary short job by $S$ and the number of short jobs by $N$ ($N \geq 3$). We denote by $\delta$ the length of the long job and by $\varepsilon$ the length of an arbitrary short job. In our game instance we choose $\delta \geq 2N$ ($\delta \geq 6$) and $\varepsilon = 1/N$ ($\varepsilon \leq 1/3$). $B$ is the activation cost and it is equal $\delta + N\varepsilon + \varepsilon = \delta + 1 + \varepsilon$. In Section 3.2 we show that there is a unique *NE* schedule in the described game instance, a schedule in which all jobs are assigned to a single machine. We denote this schedule by $s$. We now calculate the costs all jobs in schedule $s$.

$$c_j(s) = L_i(s) + \frac{p_j B}{L_i(s)}$$
$$c_X(s) = \delta + N\varepsilon + \frac{\delta(\delta + N\varepsilon + \varepsilon)}{\delta + N\varepsilon} = \delta + 1 + \frac{\delta(\delta+1) + \delta\varepsilon}{\delta + 1} = 2\delta + 1 + \frac{\delta\varepsilon}{\delta + 1}$$
$$c_S(s) = \delta + N\varepsilon + \frac{\varepsilon(\delta + N\varepsilon + \varepsilon)}{\delta + N\varepsilon} = \delta + 1 + \frac{\varepsilon(\delta+1) + \varepsilon^2}{\delta + 1} = \delta + 1 + \varepsilon + \frac{\varepsilon^2}{\delta + 1} > \delta + 1 + \varepsilon$$

We show that schedule $s$ is not a weak Pareto Optimal Nash Equilibrium (*wPNE*) by showing that there is a schedule $\bar{s}$, such that $\bar{s} \neq s$, under which for every job $j$, $c_i(\bar{s}) < c_i(s)$. Consider an assignment under which the long job is jointly assigned with one short job to single machine. The remaining $N-1$ short jobs are assigned in pairs (two short jobs assigned to single machine), each pair on different machine, when the number of short jobs is odd. When the number of short jobs is even, the remaining short jobs are assigned in pairs except for the last three short jobs that are jointly assigned in a triple (three jobs are assigned to a single machine) (See Figure 4). We denote this schedule by $\bar{s}$. In $\bar{s}$, we denote an arbitrary short job assigned together with the long job by $S_\ell$, an arbitrary short job assigned in a pair by $S_p$ and an arbitrary short job assigned in a triple by $S_t$. In schedule $\bar{s}$, the notation of the long job, $X$, remains unchanged. It is important to note that $p_S = p_{S_\ell} = p_{S_p} = p_{S_t} = \varepsilon = 1/N$.

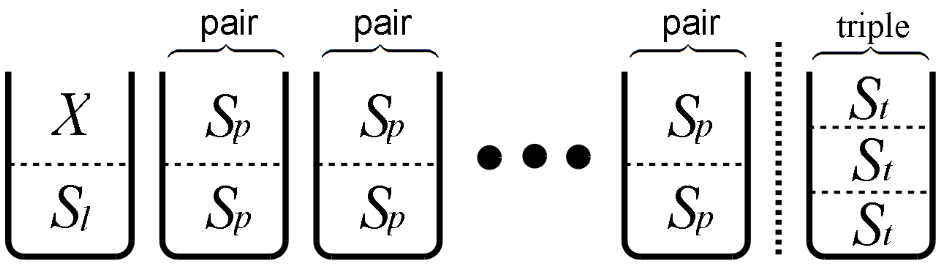


Figure 4: Schedule $\bar{s}$ under which for every job $j$, $c_j(\bar{s}) < c_j(s)$

We now calculate the costs of all jobs in schedule $\bar{s}$

$$\begin{aligned}
c_X(\bar{s}) &= \delta + \varepsilon + \frac{\delta(\delta + N\varepsilon + \varepsilon)}{\delta + \varepsilon} = \delta + \varepsilon + \frac{\delta + \delta(\delta + \varepsilon)}{\delta + \varepsilon} = 2\delta + \varepsilon + \frac{\delta}{\delta + \varepsilon} \\
c_{S_\ell}(\bar{s}) &= \delta + \varepsilon + \frac{\varepsilon(\delta + N\varepsilon + \varepsilon)}{\delta + \varepsilon} = \delta + \varepsilon + \frac{\varepsilon + \varepsilon(\delta + \varepsilon)}{\delta + \varepsilon} = \delta + 2\varepsilon + \frac{\varepsilon}{\delta + \varepsilon} < \delta + 3\varepsilon \\
c_{S_p}(\bar{s}) &= 2\varepsilon + \frac{\varepsilon(\delta + N\varepsilon + \varepsilon)}{2\varepsilon} = 2\varepsilon + \frac{\delta + N\varepsilon + \varepsilon}{2} = 2\varepsilon + \frac{\delta + 1 + \varepsilon}{2} < \delta + 3\varepsilon \\
c_{S_t}(\bar{s}) &= 3\varepsilon + \frac{\varepsilon(\delta + N\varepsilon + \varepsilon)}{3\varepsilon} = 3\varepsilon + \frac{\delta + N\varepsilon + \varepsilon}{3} = 3\varepsilon + \frac{\delta + 1 + \varepsilon}{3} < \delta + 3\varepsilon
\end{aligned}$$

We now show that for every job $j$ under schedule $\bar{s}$, $c_j(s) > c_j(\bar{s})$. We start with the long job (X) and show that $c_{\mathrm{X}}(s) > c_{\mathrm{X}}(\bar{s})$.

$$\begin{aligned}
2\delta + 1 + \frac{\delta\varepsilon}{\delta + 1} &> 2\delta + \varepsilon + \frac{\delta}{\delta + \varepsilon} \\
1 + \frac{\delta\varepsilon}{\delta + 1} &> \varepsilon + \frac{\delta}{\delta + \varepsilon}
\end{aligned}$$

Multiplying both sides by $(\delta + 1)(\delta + \varepsilon)$ yields

$$\begin{aligned}
(\delta + 1)(\delta + \varepsilon) + \delta\varepsilon(\delta + \varepsilon) &> \varepsilon(\delta + 1)(\delta + \varepsilon) + \delta(\delta + 1) \\
\delta^2 + \delta\varepsilon + \delta + \varepsilon + \delta^2\varepsilon + \delta\varepsilon^2 &> \delta^2\varepsilon + \delta\varepsilon^2 + \delta\varepsilon + \varepsilon^2 + \delta^2 + \delta
\end{aligned}$$

Rearranging both sides yields $\varepsilon > \varepsilon^2$. By definition $\varepsilon \leq 1/3$, hence the inequality holds. Therefore $c_j(\bar{s}) < c_j(s)$. It remains to show that $c_S(s) > c_{S_\ell}(\bar{s}), c_{S_p}(\bar{s}), c_{S_t}(\bar{s})$. We already showed that $c_{S_\ell}(\bar{s}), c_{S_p}(\bar{s}), c_{S_t}(\bar{s}) < \delta + 3\varepsilon$ and $c_S(s) > \delta + 1 + \varepsilon$. It is easy to verify that this inequality holds, therefore the statement holds as well.

We showed that in this particular game instance, under schedule $\bar{s}$, for every job $j$, $c_j(\bar{s}) < c_j(s)$. Therefore, neither strict nor weak Pareto Optimal Nash Equilibria exist in the game instance we introduced.

## 2.3 An improved lower bound on the *PoS*

In order to examine whether the lower bound of [8] on the *PoS* can be improved, we developed a computer program that assisted us with this task. The program receives a game instance (number of jobs, jobs lengths and activation cost) as input and calculates the *PoS* for all possible assignments. Number of machines is assumed to be unlimited. The output of the program is the highest *PoS* from among the calculated ones and the its corresponding assignment. The program can be found in Appendix A.

With the assistance of the program we have found better *PoS* lower bound (1.149) which is much higher than provided in [8](1.058). Consider the game instance $G = \langle I = \{1.15, 1, 1, 1, 1\}, B = 7.8\rangle$. The social optimum schedule for this game instance is achieved by assigning the jobs (1.15,1,1) to a single machine and the remaining jobs (1,1) to another machine. We now calculate the cost of the social optimum schedule. For a machine with (1,1), each job pays $2 + \frac{7.8}{2} = 5.9$. In machine with (1.15,1,1), the job of length 1.15 pays $3.15 + \frac{1.15 \cdot 7.8}{3.15} = 5.997$ and job of length 1 pays $3.15 + \frac{7.8}{3.15} = 5.626$. Therefore, the cost of the optimum social schedule is 5.997 (incurred by the job of length 1.15). By Observation 10, the schedule under which all jobs are assigned to a single machine is a *NE*. We denote this schedule by $s$. We now calculate the cost of $s$ and show that it is the only *NE* schedule in the game instance. Job of length 1.15 pays $5.15 + \frac{1.15 \cdot 7.8}{5.15} = 6.891$. Job of length 1 pays $5.15 + \frac{7.8}{5.15} = 6.664$. Therefore, the cost of schedule $s$ is 6.891. We now show that any arbitrary schedule $\bar{s}$, such that $\bar{s} \neq s$ is not a *NE*.

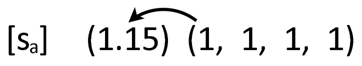


Figure 5: Schedule $s_a$

Under schedule $s_a$, for a machine with (1,1,1,1), a job of length 1 pays $4 + \frac{7.8}{4} = 5.95$. If it migrates to machine with (1.15) it will pay $2.15 + \frac{7.8}{2.15} = 5.777$ resulting in joint action (1.15,1);(1,1,1). Therefore, migration is beneficial. Hence schedule $s_a$ is not a *NE*.

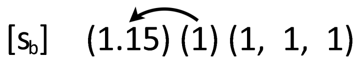


Figure 6: Schedule $s_b$

Under schedule $s_b$, for a machine with (1), the job of length 1 pays $1 + 7.8 = 8.8$. It can reduce its cost by migrating to machine with (1.15) resulting in joint action (1.15,1);(1,1,1). Hence schedule $s_b$ is not a *NE*.

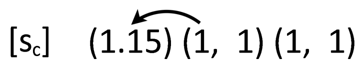


Figure 7: Schedule $s_c$

Under schedule $s_c$, for a machine with (1,1), a job of length 1 pays $2 + \frac{7.8}{2} = 5.9$. It can reduce its cost by migrating to machine with (1.15) resulting in joint action (1.15,1);(1;1,1). Hence schedule $s_c$ is not a *NE*.

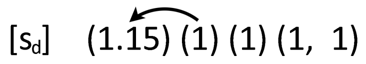


Figure 8: Schedule $s_d$

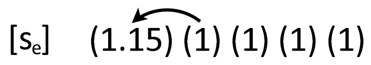


Figure 9: Schedule $s_e$

Schedule $s_d$ and schedule $s_e$ are not *NE* schedules. The proof is similar as for schedule $s_b$.

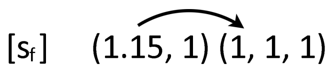


Figure 10: Schedule $s_f$

Under schedule $s_f$, for a machine with (1.15,1), the job of length 1.15 pays $2.15 + \frac{1.15 \cdot 7.8}{2.15} = 6.322$. If it migrates to machine with (1,1,1) it will pay $4.15 + \frac{1.15 \cdot 7.8}{4.15} = 6.311$ resulting in joint action (1);(1.15,1,1,1). Therefore, migration is beneficial. Hence schedule $s_f$ is not a *NE*.

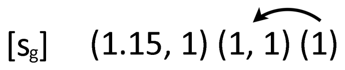


Figure 11: Schedule $s_g$

Under schedule $s_g$, for a machine with (1), the job of length 1 pays $1 + 7.8 = 8.8$. If it migrates to machine with (1,1) it will pay $3 + \frac{7.8}{3} = 5.6$ resulting in joint action (1.15,1);(1,1,1). Therefore, migration is beneficial. Hence schedule $s_g$ is not a *NE*.

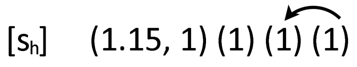


Figure 12: Schedule $s_h$

Under schedule $s_h$, for a machine with (1), the job of length 1 pays $1 + 7.8 = 8.8$. If it migrates to machine with (1) it will pay $2 + \frac{7.8}{2} = 5.9$ resulting in joint action (1.15,1);(1);(1,1). Therefore, migration is beneficial. Hence schedule $s_h$ is not a *NE*.

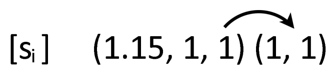


Figure 13: Schedule $s_i$

Under schedule $s_i$, for a machine with (1.15,1,1), a job of length 1 pays $3.15 + \frac{7.8}{3.15} = 5.626$. If it migrates to machine with (1,1) it will pay $3 + \frac{7.8}{3} = 5.6$ resulting in joint action (1.15,1);(1,1,1). Therefore, migration is beneficial. Hence schedule $s_i$ is not a *NE*.

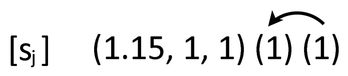


Figure 14: Schedule $s_j$

Under schedule $s_j$, for a machine with (1), a job of length 1 pays $1 + 7.8 = 8.8$. If it migrates to machine with (1) it will pay $2 + \frac{7.8}{2} = 5.9$ resulting in joint action (1.15,1,1);(1,1). Therefore, migration is beneficial. Hence schedule $s_j$ is not a *NE*.

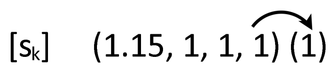


Figure 15: Schedule $s_k$

Under schedule $s_k$, for a machine with (1.15,1,1,1), a job of length 1 pays $4.15 + \frac{7.8}{4.15} = 6.029$. If it migrates to machine with (1) it will pay $2 + \frac{7.8}{2} = 5.9$ resulting in joint action (1.15,1,1);(1,1). Therefore, migration is beneficial. Hence schedule $s_k$ is not a *NE*.

We showed that any arbitrary schedule $\bar{s}$, such that $\bar{s} \neq s$ is not a *NE*. Hence there is a unique *NE* schedule for this game instance and it is achieved when all jobs are assigned to a single machine. We showed that the cost of the *NE* schedule is 6.891 and the cost of the social optimum schedule is 5.997. Therefore, $PoS = 6.891/5.997 = 1.149$.

# 3 New model

In this section we introduce a different model than the one provided in [8] and examine the *PoS* under it.

## 3.1 Model definition

Our model is identical to the model described in Section 1.2 except for the social cost function. In our model, the social cost function is the sum of all jobs' costs and not the biggest job's cost. Formally, let $g(s)$ denote the social cost function under the joint action $s$. $g(s)$ is defined as follows $g(s) = \sum_j c_j(s)$. The optimal social cost function is defined as $OPT = min_{s \in S} g(s)$.

## 3.2 *PoS* is unbounded

In [8], it was shown that *NE* always exists under the proportional sharing rule and unlimited supply of resources. This motivates us to examine the best *NE* and quantify its inefficiency under the model we introduced. In this section, we show that *PoS* is unbounded under the new model. Consider a game instance $G$ with one long job and a number of short jobs as in Section 2.2. We denote the long job by $X$, an arbitrary short job by $S$ and the number of short jobs by $N$. We denote by $\delta$ the length of the long job and by $\varepsilon$ the length of an arbitrary short job. In our game instance we choose $\delta \geq 1$ and $\varepsilon = \frac{1}{N}$. $B$ is the activation cost and it is equal to $\delta + N\varepsilon + \varepsilon = \delta + 1 + \varepsilon$. Let $s$ be a schedule in which all jobs are assigned to a single machine. In schedule $s$, $B \geq \sum_j p_j$ (results from the definition of $B$ and the definition of the game instance). By Observation 10 schedule $s$ is a *NE*.

**Claim 13.** *Schedule $s$ in which all jobs are assigned to a single machine is the only NE assignment in the given game instance.*

*Proof.* We divide all possible assignments into two cases and examine them.

**Case 1.** Consider an assignment $s$ under which the long job is assigned to a dedicated machine. The remaining short jobs are assigned to one or more machines (Figure 16). We show that under this schedule it is always beneficial for the long job to migrate to any other machine in the schedule.

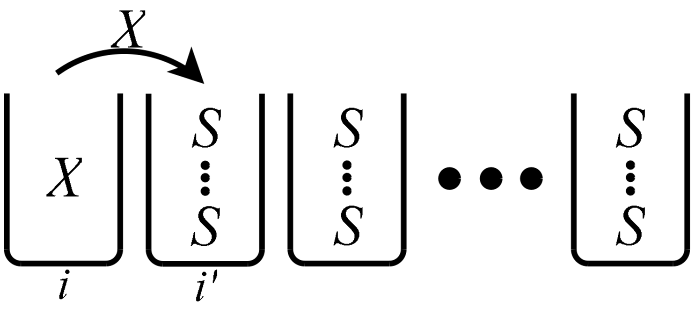


Figure 16:
$X$ is assigned to machine $i$ and prefers to migrate to machine $i'$

By Lemma 6, job $j$ reduces its cost by a migration from machine $i$ 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 $B < \rho$. We show that the migration condition $L_{i'}(s) + p_j > L_i(s)$ and $B > \rho$ is always satisfied for the long job $X$.

$$\rho = \frac{L_i(s)(L_{i'}(s) + p_j)}{p_j} = \frac{\delta(L_{i'}(s) + \delta)}{\delta} = L_{i'}(s) + \delta$$

At least one short job and at most $N$ short jobs are assigned to machine $i'$. Hence, $\varepsilon \leq L_{i'}(s) \leq N\varepsilon$. The above equation can be rewritten as $\rho = L_{i'}(s)+\delta \leq N\varepsilon+\delta < \delta+N\varepsilon+\varepsilon = B$. We obtained that $B > \rho$. On the other hand $L_{i'}(s)+p_j \geq \varepsilon+\delta > \delta = L_i(s)$. We obtained that $L_{i'}(s) + p_j > L_i(s)$. Hence the migration condition is satisfied and the long job will always prefer to migrate to machine $i'$ (any other machine in schedule $s$) no matter how many short jobs are assigned to it. Therefore, the examined schedule is not an *NE*.

**Case 2.** Consider an assignment $s$ under which the long job is jointly assigned together with $n$ short jobs, such that $1 \leq n < N$ (if $n = N$ then all jobs would be assigned to a single machine). The rest of the short jobs are assigned to one or more (other) machines (Figure 17). Assume without loss of generality that $m$ short jobs are jointly assigned to machine $i'$, such that $1 \leq m < N$. We show that in the described assignment it is always beneficial for a short job that is assigned together with the long job to migrate to a machine that has only short jobs assigned to it.

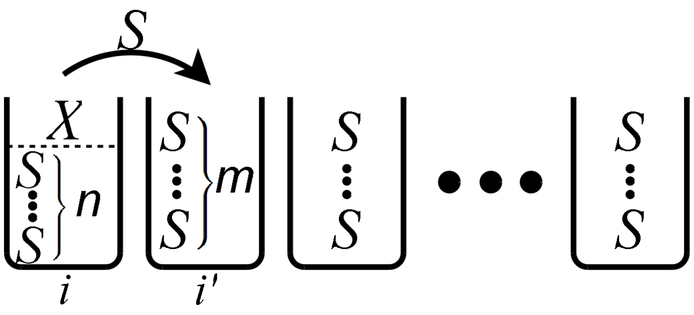


Figure 17: $1 \leq n, m < N$

Arbitrary short job $S$, that is assigned to machine $i$, prefers to migrate to machine $i'$

We show that the migration condition $L_{i'}(s) + p_j < L_i(s)$ and $B < \rho$ is always satisfied for the short job.

$$\begin{aligned}\rho &= \frac{L_i(s)(L_{i'}(s) + p_j)}{p_j} = \frac{(\delta + n\varepsilon)(m\varepsilon + \varepsilon)}{\varepsilon} \\ &= \frac{\varepsilon(\delta + n\varepsilon)(m+1)}{\varepsilon} = (\delta + n\varepsilon)(m+1)\end{aligned}$$

According to the game instance's definition, $\delta \geq 1$ and $N\varepsilon = 1$. We also assumed that under schedule $s$, $n, m \geq 1$. The above equation can be rewritten as

$$\begin{aligned}\rho &= (\delta + n\varepsilon)(m+1) = \delta m + \delta + nm\varepsilon + n\varepsilon \\ &> \delta + 1 + n\varepsilon = \delta + N\varepsilon + n\varepsilon > \delta + N\varepsilon + \varepsilon = B\end{aligned}$$

We obtained that $B < \rho$. It remains to show that $L_{i'}(s) + p_j < L_i(s)$. We assumed that $m < N$, resulting in $m\varepsilon < N\varepsilon = 1 \leq \delta$. We also assumed that $1 \leq n$, resulting in $\varepsilon \leq n\varepsilon$. Therefore, $L_{i'}(s) + p_j = m\varepsilon + \varepsilon < \delta + \varepsilon \leq \delta + n\varepsilon = L_i(s)$. Hence, the migration condition is satisfied and the short job will always prefer to migrate to machine $i'$ (i.e to any other machine in schedule $s$) no matter how many short jobs are assigned to it. Therefore, the examined schedule is not an *NE*.

Therefore, the only *NE* schedule is the schedule under which all jobs are assigned to a single machine

$\square$

We now show that for the examined game instance, *PoS* is unbounded. In Section 2.2 we introduced a game instance for which Strict Pareto Optimal Nash Equilibrium might not always exist. The game instance from section 2.2 ($\delta \geq 6$) is a special case of the game instance we introduced here ($\delta \geq 1$). We already calculated the costs of all the involved jobs in Section 2.2, which remains the same.

$$\begin{aligned} c_X(s) &= 2\delta + 1 + \frac{\delta\varepsilon}{\delta+1} \\ c_S(s) &= \delta + 1 + \varepsilon + \frac{\varepsilon^2}{\delta+1} \end{aligned}$$

The social cost under schedule $s$ is

$$\begin{aligned} g(\mathrm{s}) &= \sum_j c_j(s) = c_X(s) + N \cdot c_S(s) \\ &= 2\delta + 1 + \frac{\delta\varepsilon}{\delta+1} + N\left(\delta + 1 + \varepsilon + \frac{\varepsilon^2}{\delta+1}\right) \\ &= 2\delta + 1 + \frac{\delta\varepsilon}{\delta+1} + N\delta + N + 1 + \frac{N\varepsilon\cdot\varepsilon}{\delta+1} \\ &= \delta(2+N) + (2+N) + \frac{\delta\varepsilon}{\delta+1} + \frac{\varepsilon}{\delta+1} \\ &= (\delta+1)(N+2) + \frac{\varepsilon(\delta+1)}{\delta+1} = (N+2)(\delta+1) + \varepsilon \end{aligned}$$

In our proof we will refer to the schedule under which the long job is assigned to a dedicated machine and all the remaining short jobs are assigned to a different (single) machine as the social optimum schedule. We denote this schedule by $\bar{s}$. This schedule may not be the best social optimum schedule. The cost of the actual social optimum schedule may be smaller than or equal to the one we chose. Therefore, if we show that $PoS$ is unbounded for this social optimum schedule, it will be true for the actual social optimum schedule as well.

We now calculate the costs of all jobs under the optimum schedule $\bar{s}$.

$$\begin{aligned} c_X(\bar{s}) &= \delta + B = \delta + \delta + N\varepsilon + \varepsilon = 2\delta + 1 + \varepsilon \\ c_S(\bar{s}) &= N\varepsilon + \frac{\varepsilon(\delta + N\varepsilon + \varepsilon)}{N\varepsilon} = 1 + \delta\varepsilon + \varepsilon + \varepsilon^2 \end{aligned}$$

The social cost under the optimum schedule $\bar{s}$ is

$$\begin{aligned} OPT &= \sum_j c_j(\bar{s}) = c_X(\bar{s}) + Nc_S(\bar{s}) \\ &= 2\delta + 1 + \varepsilon + N(1 + \delta\varepsilon + \varepsilon + \varepsilon^2) \\ &= 2\delta + 1 + \varepsilon + N + N\varepsilon\delta + N\varepsilon + N\varepsilon^2 \\ &= 2\delta + 1 + \varepsilon + N + \delta + 1 + \varepsilon = 3\delta + N + 2 + 2\varepsilon \end{aligned}$$

We now calculate the $PoS$.

$$PoS = \frac{g(s)}{OPT} = \frac{(N+2)(\delta+1)+\varepsilon}{3\delta+N+2+2\varepsilon} = \frac{(N+2)(\delta+1)+\frac{1}{N}}{3\delta+N+2+\frac{2}{N}} \tag{1}$$

The values $N$ and $\delta$ are arbitrary and any value within the game instance's boundaries ($\delta \geq 1$ and $N \geq 1$) can be assigned to them. First, let $\delta$ to be equal $N+2$. Equation (1) can be rewritten as

$$PoS = \frac{(N+2)(\delta+1)+\frac{1}{N}}{3\delta+N+2+\frac{2}{N}} = \frac{(N+2)(N+2+1)+\frac{1}{N}}{4(N+2)+\frac{2}{N}} \tag{2}$$

Next, we let $N$ to be big enough so that $\frac{1}{N}$ will be negligible. Let $N'$ denote $N+2$. Equation (2) can be rewritten once again as

$$PoS = \frac{(N+2)(N+2+1)+\frac{1}{N}}{4(N+2)+\frac{2}{N}} \rightarrow \frac{N'(N'+1)}{4N'} = \frac{N'+1}{4} \tag{3}$$

From Equation (3) it is clear that for any given $r$, there exist instances for which $PoS > r$. Therefore, $PoS$ is unbounded.

# References

[1] E. Anshelevich, A. Dasgupta, J. M. Kleinberg, É.Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. In *Proc. of the 45st Annual Symposium on Foundations of Computer Science (FOCS'04)*, pages 295–304, 2004.

[2] S. Albers, S. Elits, E. Even-Dar, Y. Mansour, and L. Roditty On Nash Equilibria for a Network Creation Game. In *Proc. of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA'06)*, pages 89–98, 2006.

[3] S. Chien, A. Sinclair. Strong and pareto price of anarchy in congestion games. In . *Proc. of the 36th International Colloquium on Automata, Languages and Programming (ICALP'09)*, pages 279–291, 2009.

[4] A. Epstein, M. Feldman, and Y. Mansour. Strong Equilibrium in Cost Sharing Connection Games. In *Proc. of the 8th ACM Conference on Electronic Commerce (EC'07)*, pages 84–92, 2007.

[5] L. Epstein and E. Kleiman. Selfish bin packing. In *Proc. of the 16th Annual European Symposium on Algorithms (ESA'08)*, pages 368–380, 2008.

[6] L. Epstein, S. O. Krumke, A. Levin and H. Sperber. Selfish bin coloring. *Journal of Combinatorial Optimization (JCO)*, 2010.

[7] D. Fotakis, S. Kontogiannis, M. Mavronicolas, and P. Spiraklis. The Structure and Complexity of Nash Equilibria for a Selfish Routing Game. *Theoretical Computer Science (TCS)*, 410(36):3305-3326, 2009.

[8] M. Feldman and T. Tamir. Conflicting Congestion Effects in Resource Allocation Games. In *Proc. of the 4th International Workshop on Internet and Network Economics (WINE'08)*, pages 109–117, 2008.

[9] R. Graham. Bounds on Multiprocessing Timing Anomalies. *SIAM Journal of Applied Mathematics (SIAMAM)*, 17(2):416-429, 1969.

[10] R. Holzman, N. Law-Yone. Strong equilibrium in congestion games. *Games and Economic Behaviour*, 21(1-2), pages 85–101, 1997

[11] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *Proc. of the 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99)*, pages 404–413, 1999.

[12] N. Nisan, T. Roughgarden, É. Tardos, and V. V. Vazirani. Algorithmic Game Theory. *Cambridge University Press*, 2007.

[13] T. Roughgarden and É. Tardos. How bad is selfish routing? In *Proc. of the 41st Annual Symposium on Foundations of Computer Science (FOCS'00)*, pages 93–102, 2000.

# Appendix A - Computer program

```
import sys

def assignmentsGenerator(rJobLengths):
  """ Assignments generator

  This is not a recursive function but a generator. Each time it
  is called it returns a unique assignment that was not yet
  returned.  Same results  can be achieved with the help of
  recursion, but it will much more memory and time consuming.

  \param  rJobLengths         jobs' lengths
  """

  if 2 == len(rJobLengths):
    yield [[rJobLengths[0]], [rJobLengths[1]]]
    yield [[rJobLengths[0], rJobLengths[1]]]
  else:
    for assignment in assignmentsGenerator(rJobLengths[1:]):
      retAssignment = [[rJobLengths[0]]]
      retAssignment.extend(assignment[:])
      yield retAssignment

      for element in assignment:
        retAssignment = assignment[:]
        retAssignment.remove(element)
        yield [[rJobLengths[0]] + element] + retAssignment
      #endfor
    #endfor
  #endif
#enddef


def calculateCost(rActivationCost, rAssignment):
  """ Calculate cost of an assignment and check if it is a NE

  \param  rActivationCost       activation cost
  \param  rAssignment       assignment
  \return List containing two element. If the given assignment
          is a NE, first element will be set to True,otherwise
          it will be set to False. Second element is the
          assignment cost.
```

```
"""

isNE = True

# calculate the load on (all) machines
machinesLoad = []
for machine in rAssignment:
  singleMachineLoad = 0.0
  for job in machine:
    singleMachineLoad += job
  #endfor
  machinesLoad += [singleMachineLoad]
#endfor

assignmentCost = 0
for i in range(len(rAssignment)):
  for job in rAssignment[i]:
    # rAssignmetn[i] is de facto machine i under the examined
    # assignment
    cost = 0.0

    # calculate the cost of of a signle job assigned to machine i
    cost = machinesLoad[i] + \
      ((job * rActivationCost) / machinesLoad[i])

    # calculate the assignment cost (makespan)
    assignmentCost = max(assignmentCost, cost)

    # check if it worth  migrating to a  dedicated machine. if
    # the migration  is beneficial, then the currectassignment
    # is not NE.
    if (cost > (job + rActivationCost)):
      isNE = False
      continue
    #endif

    # check if it worth migrating to any other machine.if the
    # migration is beneficial, then the currect assignment is
    # not NE.
    for k in range(len(rAssignment)):
      if i == k:  # skip current machine
        continue
      #endif

      L1 = machinesLoad[k]  # other machine's load
```

```
        L   = machinesLoad[i]  # current machine's load
        rho = 0.0
        rho = (L * (L1 + job)) / job

        # we use Lemma 6 to check if it worth migrating.
        if (((L1 + job) > L) and (rActivationCost > rho)) or \
          (((L1 + job) < L) and (rActivationCost < rho)):
          isNE = False
        #endif
      #endfor
    #endfor
  #endfor

  if True == isNE:
    return  [True, assignmentCost]
  else:
    return [False, assignmentCost]
  #endif
#enddef


def calculatePOS(rActivationCost, rJobLengths):
  """ Calculate Price Of Stability (PoS) for the given game

  \param  rActivationCost     activation cost
  \param  rJobLengths         jobs' lengths
  \return PoS value
  """

  socialOpt      = 0.0
  socialSet      = None
  neAssignments = []

  for schedule in assignmentsGenerator(rJobLengths):
    # calculate assignment cost
    retVal = calculateCost(rActivationCost, schedule)

    # if assignment cost is smaller than social optimum let it be
    # the new social optimum
    if (0 == socialOpt):
      socialOpt = retVal[1]
      socialSet = schedule
    elif (retVal[1] < socialOpt):
      socialOpt = retVal[1]
      socialSet = schedule
    #endif

    # if assignment is NE, append it to NE assignments list
    if True == retVal[0]:
     neAssignments.append([schedule, retVal[1]])
    #endif
  #endfor

  pos = 0.0

  # by Theorem 12, NE always exists for the examined model under
  # the proportional sharing rule. thus, PoS is well defined.find
  # best NE and calculate PoS.
  if neAssignments:
    #find the best NE value
    bestNE = reduce(lambda x, y: [[], min(x[1], y[1])], \
                    neAssignments)[1]

  # filter NE assignments. Leave only these which cost equal to
  # bestNE
  neAssignments = filter(lambda x: x[1] == bestNE, \
                         neAssignments)

  # socialOpt never equals 0
  pos = bestNE / socialOpt

  # list all best nash equilibria assignments
  print 'best nash equilibrium cost=%f' % bestNE
  print 'best nash equilibrium schedules'
  for ne in neAssignments:
    print ne[0]
  #endfor
  print ''

  print 'social optimum=%f' % socialOpt
  print 'social schedule'
  print socialSet
  print ''

  return pos
#enddef
```

```
if __name__ == "__main__":
  jobsLengths = [1.15,1,1,1,1]
  activationCost = 7.8

  print "activation cost=%f\n" % activationCost
  for i in range(len(jobsLengths)):
    print 'job %d length %f' % (i + 1, jobsLengths[i])
  #endfor
  print ''

  pos = calculatePOS(activationCost, jobsLengths)

  print 'PoS %f'

  sys.exit(0)
#endif
```

# תקציר

רוב העבודות הקודמות שנעשו בתחום הקצאת משאבים, והתמקדו בפרספקטיבה האלגוריתמית של תורת המשחקים, נחלקות לשני סוגים. בבעיות מהסוג הראשון, פונקצית העלות מתחשבת בהשפעות שליליות של צפיפות כגון בעיות תזמון על מכונות מרובות (job scheduling) ומשחקי ניתוב ( network routing), כאשר בהינתן פונקצית עלות כזו, תעדיף משימה להיות מוקצית למשאב לא עמוס. בבעיות מהסוג השני, ההתמקדות היא בהשפעות חיוביות של צפיפות, כגון תכנון רשתות (network design) ומשחקי  הקצאה לתאים (bin packing), בהם תעדיף משימה לחלוק את המשאב עם מספר רב של משימות אחרות.

אנו חוקרים משחקי הקצאות משאבים עם פונקצית עלות שמתחשבת הן בהשפעות החיוביות של צפיפות והן בהשפעותיה השליליות. נציג מודל שנחקר בעבר על ידי Feldman ו-Tamir [8], ובו שתי ההשפעות מתקיימות בו זמנית. נראה שבמודל זה, שווי משקל מהסוג Strictly Pareto optimal Nash Equilibria תמיד קיים עבור משחקים עם מספר משתתפים שלא עולה על 3. מנגד, נראה שכאשר מספר המשתתפים עולה על 3, Strictly Pareto optimal Nash Equilibria לא בהכרח קיים. בנוסף נציג חסם תחתון טוב גבוה על מחיר היציבות (Price of Stability) המשפר את החסם התחתון של [8]. לבסוף נציג מודל חדש שמבוסס על המודל הנזכר לעיל ונראה כי במודל החדש למרות ששווי משקל טהור מסוג - Nash Equilibrium תמיד קיים,  מחיר היציבות איננו חסום.

# תוכן עניינים

**האוניברסיטה הפתוחה**

**המחלקה למתמטיקה ולמדעי המחשב**

# השפעות חיוביות ושליליות של צפיפות במשחקי הקצאת משאבים

עבודה מסכמת זו הוגשה כחלק מהדרישות לקבלת תואר

"מוסמך למדעים" M.Sc. במדעי המחשב

באוניברסיטה הפתוחה

החטיבה למדעי המחשב

על-ידי

**איליה כץ**

העבודה הוכנה בהדרכתה של **ד"ר לאה אפשטיין**

אוגוסט 2010