IDC HERZLIYA | Efi Arazi School of Computer Science

The Interdisciplinary Center, Herzlia

Efi Arazi School of Computer Science

LOAD REBALANCING GAMES IN DYNAMIC SYSTEMS

M.Sc. Dissertation

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science (M.Sc.) Research Track in Computer Science

Submitted by Sofia Belikovetsky

Under Supervision of Prof. Tami Tamir

January, 2013

Acknowledgements

First and foremost, I express my sincerest gratitude to my supervisor, Prof. Tami Tamir who has supported me throughout my thesis with her patience and immense knowledge. Without Her guidance, motivation and enthusiasm, this thesis would not have been completed. I could not have wished for a better supervisor!

I would also like to thank my thesis committee: Prof. Yishay Mansour and Prof. Leah Epstein for reading this work and for their important remarks and questions.

Abstract

We consider a dynamic variant of the classic load balancing game, in which selfish jobs need to be assigned on a set of identical parallel machines, and each job's cost is the load on the machine it is assigned to. Given an initial assignment, the system is modified; specifically, some machines are added or removed. When machines are added, jobs naturally have an incentive to migrate to the new unloaded machines. When machines are removed, the jobs assigned to them must be reassigned. As a result of these migrations, other jobs might also benefit from migrations. The goal is to find a pure *Nash Equilibrium* (NE) assignment in the modified system. A deviation from the initial assignment is associated with a penalty. We introduce and study the job-extension penalty model. In this model, we are given *an extension parameter* $\delta \geq 0$. If the machine on which a job is assigned to is different from its initial machine, then the job's processing time is extended by δ.

We provide answers to the basic questions arising in this model. Namely, the existence and calculation of a Nash equilibrium and a strong Nash equilibrium, and their inefficiency compared to an optimal schedule. We show that, for any extension parameter, a NE exists and BRD converges to a NE. The convergence time is linear when the jobs are activated in a certain way. We prove lower and upper bounds for the price of stability (PoS), the price of anarchy (PoA), the strong price of stability (SPoS) and the strong price of anarchy (SPoA).

We show that in general, the price of anarchy is unbounded when machines are either added or removed. The PoA can be bounded if the modified schedule is achieved by performing improvement steps. Specifically, let m_0, m' denote the number of initial machines and added/removed machines respectively. For NEs that are achieved by performing improvement steps, we show that the PoA is (i) $2 + \frac{m'-1}{m_0}$ when machines are added, (ii) $m_0 - m'$ when machines are removed, and (iii) $2 - \frac{1}{m_0 - m'}$ when machines are removed, and jobs are activated in a specific order, denoted *two-phase better-response*.

We show that the SPoA is 3 for the both adding and removing machines scenarios. We also provide a closer analysis of the strong price of anarchy, and bound it as a function of the ratio between δ and OPT. Specifically, we show that the SPoA is $2 + \frac{\delta}{OPT}$.

Our work adds two realistic considerations to the study of job scheduling using game theoretic approach: the analysis of the common situation in which systems are upgraded or suffer from failures, and the practical fact according to which job migrations are associated with a cost.

Contents

1 Introduction

The well-studied *load balancing* problem considers a scenario in which a set of jobs needs to be assigned on a set of identical parallel machines. Each job j, is associated with a processing time p_j and the goal is to balance the load on the machines. In contrast to the traditional load balancing problem, where a central designer determines the allocation of jobs to machines and all the participating entities are assumed to obey the protocol, in the *load balancing game*, each job is owned by a selfish agent who wishes to optimize its own objective.

Given an assignment, each job incurs a cost which is equal to the total load on the machine it is assigned to. This cost function characterizes systems in which jobs are processed in parallel, or when all jobs on a particular machine have the same single pick-up time, or need to share some resource simultaneously. This problem have been widely studied in recent years from a game theoretic perspective, see [21, 2, 6, 8, 12], and a survey in [23].

In this work, we consider a dynamic variant of this game. Specifically, we are given an assignment, s_0, of n jobs on m_0 machines. The system is modified, namely, m' machines are added or removed. When machines are added, jobs will naturally have an incentive to migrate to the new unloaded machines. When machines are removed, the jobs assigned to the removed machine must be reassigned. As a result of these migrations, other jobs might also benefit from migrations. The goal is to find a pure *Nash Equilibrium* (NE) assignment, s, in the modified system. In such an assignment, no job can reduce its cost by migrating to a different machine. Apparently, this can be viewed as a new instance of the load balancing game. However, in the model we consider, a deviation from the initial assignment is associated with a penalty. We introduce and study the job-extension penalty model. In this model, we are given *an extension parameter* $\delta \geq 0$. If the machine on which job j is scheduled in s is different from its initial machine in s_0, then the processing time of j is extended to be $p_j + \delta$. Practically, this penalty is justified by the fact the reassignment of j causes some extra work on the system, for example, if some preprocessing or configuration set-up was already performed according to the initial assignment.

Note that this model can be seen as a restricted case of scheduling on related machines. For every job j and machine M_i, the processing time of j on M_i is p_j if j was scheduled on machine

M_i in s_0 and $p_j + \delta$ otherwise. Our analysis provides tighter results than those known for related machines [8]. Also note that if s is produced from s_0 by a sequence of improvements steps, then the extension penalty is independent of the number of steps and only the final assignment of j in s matters. In particular, if j leaves its original machine and returns to it latter as a part of the improvement steps sequence then j is not extended.

For the above load rebalancing game, we study the problem of equilibrium existence, calculation, and inefficiency. We quantify the inefficiency incurred due to self-interested behavior according to the *price of anarchy* (PoA) [21, 22] and *price of stability* (PoS) [1] measures, which quantify to what extent a system can benefit from a central coordinator or regulator.

The measures of PoA and PoS are quantified according to a well-defined objective function. In this work, we consider the egalitarian objective function, i.e., we wish to minimize the cost of the job with the highest cost. In scheduling terms, this is equivalent to minimizing the maximal load on some machine (also known as *makespan*).

We distinguish between the following scenarios:

1. The initial schedule s_0 might be a pure NE or not.
2. The system's modification might be addition or removal of machines.
3. The modified schedule is achieved by performing a sequence of improvement steps, a sequence of best-improvement steps, or arbitrarily.

In addition, we study the existence and quality of *strong equilibria* in our setting. A strong equilibrium (SE) [3] is a strategy profile from which no *coalition* of agents can deviate in a way that strictly benefits each one of its members. For the static load balancing game, it is known that a SE exists and can be computed efficiently [2]. We explore the existence of a SE in our dynamic setting, its computation and inefficiency (also known as the strong price of stability and strong spice of anarchy).

Applications:

Traditional analysis of job scheduling setting, assume a central utility that determines the allocation of jobs to machines and all the participating entities are assumed to obey the protocol.

However, in practice, many systems are used by heterogeneous, autonomous agents, which often display selfish behavior and attempt to optimize their own objective rather than the global objective. Game theoretic analysis provides us with the mathematical tools to study such situations, and indeed has been extensively used recently to analyze multiagent systems. This trend is motivated in part by the emergence of the Internet, which is composed of distributed computer networks managed by multiple administrative authorities and shared by users with competing interests [22].

Our work adds two realistic consideration to the study of job scheduling using game theoretic analysis. First, we assume that the system is dynamic and resources might be added or removed - this reflects the common situation in which systems are upgraded or suffer from failures. Second, we assume that job migrations are associated with a cost. Indeed, in real systems, migrations do incur some cost.

The added cost might be due to the transferring overhead or due to set-up time that should be added to the job's processing time on its new location. Consider for example an initial allocation of clients to download servers. Assume that some preprocessing is done at the time a client is assigned to a server, before the download actually started (e.g., locating the required file, format conversion, etc.). Clients might choose to switch to a mirror server. Such a change would require repeating the preprocessing work on the new server. In our model, this added work is represented by the extension penalty.

Another example of a system in which extension penalty occurs is of an RPC (Remote Procedure Call) service. In this service, a cloud of servers enables service to simultaneous users. When the system is upgraded, more virtual servers are added. Users might switch to the new servers and get a better service (with less congestion), however, some set-up time and configuration tuning is required for each new user.

1.1 Model and Preliminaries

A job rescheduling setting is define by the tuple $G = \langle M_0, M', N, p_j, \delta \rangle$, where M_0 is a set of the initial identical machines and M' is a set of added or removed machines. If the modification is machines' addition, then M' is a set of new machines, all identical to the machines in M_0.

If the modification is machines' removal then $M' \subseteq M_0$. We denote by m_0, m' the number of machines in M_0, M', respectively. $N = 1, \ldots, n$ is the set of jobs, for each job $j \in N$, p_j denotes the processing time of job j. $\delta > 0$ is the extension parameter, i.e, the time penalty that is added to the processing time of a migrating job. An assignment method produces an assignment $s = (s(1), \ldots, s(n))$ of jobs into machines, where $s(j)$ is the machine to which job j is assigned. The assignment is referred to as a schedule. The schedule s_0 denotes the initial schedule of the jobs before the systems' modification. Let s denote the assignment after the modification. In s, the processing time of a job $j \in N$ on machine $i \in M_0 \bigcup M'$ is p_j if $i = s_0(j)$ and $p_j + \delta$ otherwise. The load on a machine i in a schedule s is the sum of the processing times (including the extension penalty) of the jobs assigned to i, that is, $L_i(s) = \sum_{j:s(j)=i} p_j + \delta_{i,j}$ where $\delta_{i,j} = 0$ if $s_0(j) = i$ and δ otherwise. For a job $j \in N$, let $c_j(s)$ be the cost of job j in the schedule s, then $c_j(s) = L_{s(j)}$. In our job scheduling game, the objective function is minimizing the *makespan*, which is the load on the most loaded machine (or equivalently, the highest cost of some job). Formally, for a schedule s, $makespan(s) = max_j c_j(s) = max_i L_i(s) = L_{max}(s)$.

An assignment s is a *pure Nash equilibrium* (NE) if no job $j \in N$ can benefit from unilaterally deviating from its machine to another machine; i.e., for every $j \in N$ and every machined $i \neq s(j)$, $L_i + p_j + \delta_{i,j} \geq L_{s(j)}$.

The following definitions are used to evaluate the inefficiency of the NE schedule.

Definition 1.1 *Let $\mathcal{G}$ be a family of games, and let $G \in \mathcal{G}$ be some game in this family. Let $\Phi(G)$ be the set of Nash equilibria of the game G. If $\Phi(G) \neq \emptyset$:*

- *The* price of anarchy *of the game G is the ratio between the* maximal *cost of a Nash equilibrium and the social optimum of G:*

$$PoA(G) = \max_{s \in \Phi(G)} L_{max}(s)/OPT(G),$$

and the price of anarchy *of the family of games $\mathcal{G}$ is*

$$PoA(\mathcal{G}) = Sup_{G \in \mathcal{G}} PoA(G).$$

- *The* price of stability *of the game G is the ratio between the* minimal *cost of a Nash equi-*

librium and the social optimum of G:

$$PoS(G) = \min_{s \in \Phi(G)} L_{max}(s)/OPT(G),$$

and the price of stability *of the family of games $\mathcal{G}$ is:*

$$PoS(\mathcal{G}) = Sup_{G \in \mathcal{G}} PoS(G).$$

In section 4 we study coordinated deviations. A set of players $\Gamma \subseteq N$ forms a *coalition*, if there exists a move where each job $j \in \Gamma$ strictly reduces its cost. An assignment s is a *strong equilibrium* (SE) if there is no coalition $\Gamma \subseteq N$ that has a beneficial move from s. Clearly, a strong equilibrium is a refinement of the notion of Nash equilibrium (in particular, if s is a strong equilibrium, it is resilient to migration of coalitions of size 1, which coincides with the definition of NE).

Definition 1.2 *Let $\mathcal{G}$ be a family of games, and let $G \in \mathcal{G}$ be some game in this family. Let $\Phi(G)$ be the set of strong equilibria of the game G. If $\Phi(G) \neq \emptyset$:*

- *The* strong price of anarchy *of the game G is the ratio between the* maximal *cost of a strong equilibrium and the social optimum of G:*

  $$SPoA(G) = \max_{s \in \Phi(G)} L_{max}(s)/OPT(G),$$

  and the strong price of anarchy *of the family of games $\mathcal{G}$ is*

  $$SPoA(\mathcal{G}) = Sup_{G \in \mathcal{G}} SPoA(G).$$

- *The* strong price of stability *of the game G is the ratio between the* minimal *cost of a strong equilibrium and the social optimum of G:*

  $$SPoS(G) = \min_{s \in \Phi(G)} L_{max}(s)/OPT(G),$$

  and the strong price of stability *of the family of games $\mathcal{G}$ is:*

  $$SPoS(\mathcal{G}) = Sup_{G \in \mathcal{G}} SPoS(G).$$

1.2 Related Work

The Minimum Makespan Scheduling Problem: The minimum makespan problem corresponds to the centralized version of our game in which all jobs obey the decisions of one utility. The goal is to assign the jobs on m identical machines in a balanced way – that minimizes the last completion time. A simple reduction from the *Partition* problem implies that this problem is NP-hard even for two identical machines [16]. The simple List-scheduling algorithm [17] is a greedy algorithm that assigns the jobs in arbitrary order, each job to a machine that would minimize its completion time, given the current schedule. List-scheduling provides a $(2-\frac{1}{m})$-approximation to the minimum makespan problem. A bit better approximation ratio is guaranteed by the Longest Processing Time (LPT) algorithm [18]. LPT algorithm applies List-scheduling on the jobs in non-increasing order of their lengths. The approximation ratio of LPT is $(\frac{4}{3}-\frac{1}{3m})$. A PTAS for the minimum makespan problem on identical machines is given in [19].

Load Balancing Games: In the associated game, each job is controlled by a selfish agent who aims to minimize its cost - given by the load on the machine it is assigned to. While List-scheduling is not guaranteed to provide a NE schedule, Fotakis et al. show that schedules that result from LPT algorithm are NE schedules [14]. In [9], Even-dar et al. show that a load balancing game with unrelated machines always converges in a Nash equilibrium and determine the convergence time to be linear, specifically reached after at most n steps by performing BRD in a specific order.

The concept of the price of anarchy (PoA) was introduced by Koutsoupias and Papadimitriou in [21]. They studied a model of a routing game consisting of a source and a sink connected by m parallel edges with possibly different speeds. Each agent has an amount of traffic that he seeks to map to one of the edges such that the total load on this edge is as small as possible. They proved that in this model, the price of anarchy is $2-\frac{1}{m}$. Their results are in fact valid in our model, where the parallel edges between the source and the sink correspond to the machines, and the routing requests correspond to jobs. In [13], Finn and Horowitz presented an upper bound of $2-\frac{2}{m+1}$ for the price of anarchy in load balancing games with identical machines. Note that in this game, the PoA is equivalent to the makespan approximation. The upper and lower bounds of the price of anarchy in load balancing games with uniformly related machines was studied by Czumaj and Vocking in [8] and is bounded by $\frac{\log m}{\log\log\log m}$. The price of stability (PoS), measures

the best-case inefficiency of a Nash equilibrium, and is defined as the ratio between the best NE and the optimal solution. This measure was introduced by Anshelevich et al. in [1], for cost-sharing network formation games. It is shown that the price of stability with respect to the total-cost objective is $H(k) = 1 + \frac{1}{2} + \ldots + \frac{1}{k}$.

Other related work deals with cost functions that depend on the internal order of jobs, e.g., in [5, 4, 20]. A different cost function, in which the job's cost is affected by both the congestion on the machine and the machines' activation cost is studied by Feldman and Tamir in [11]. In [5], Caragiannis et al. study the price of stability of load balancing games with respect to the objective of minimizing the total completion time, and shows that it is at most 4/3. Other definitions of the social cost are considered, e.g., by Gairing et al. in [15].

The strong price of anarchy with respect to coalitional moves was studied by Andelman et al. in [2]. The paper proves the existence of a strong equilibrium and shows that for unrelated machines the strong price of anarchy can be bounded as a function of the number of machines and the size of the coalition. Specifically, for m unrelated machines and n players the worst-case k-SPoA, that considers coalitions of size at most k, is at most $O(nm^2/k)$ and at least $\Omega(n/k)$. A survey of results on selfish load balancing and routing on parallel links appear in [7, 23].

1.3 Our Results

We study the problem of equilibrium existence, calculation, and inefficiency in the load rebalancing game with uniform extension penalty. We show that any job scheduling game with adding or removing machines obeys at least one Nash equilibrium schedule. Moreover, some optimal solution is also a Nash equilibrium, and thus, the price of stability is 1. We show that in general, the price of anarchy is unbounded when machines are either added or removed. The PoA can be bounded if the modified schedule is achieved by performing improvement steps. Specifically, for NE's that are achieved by performing improvement steps, we show that the PoA is

1. $2 + \frac{m'-1}{m_0}$ when machines are added.

2. $m_0 - m'$ when machines are removed.

3. $2 - \frac{1}{m_0 - m'}$ when machines are removed, and jobs are activated in a specific order, denoted

two-phase better-response.

For all the above cases we prove the upper bound and provide matching lower bounds. The lower bounds are tight for some values of m_0, m' and almost tight for other values.

We also analyze the load rebalancing game assuming coordinated deviations are allowed. We prove that a strong equilibrium exists for all system modifications and that the SPoS is 1. We show that the SPoA is 3 for the both adding and removing machines scenarios and that this bound is tight. Moreover, we provide a closer analysis of the strong price of anarchy, and bound this value as a function of the ratio between δ and OPT. Specifically, we show that the strong price of anarchy is $2 + \frac{\delta}{OPT}$. We also show that for any value of $\delta > 0$ it is NP-hard to determine whether a given modified schedule is a SE.

The work is organized as follows: In Section 2 we examine the scenario where m' identical machines are added. In Section 3 we examine the scenario in which m' machines are removed. In Section 4 we consider coordinated deviation of jobs. For each scenario we consider the problem of equilibrium existence, calculation, and inefficiency, distinguishing between various initial states and convergence methods.

We note that in the load rebalancing game with no migration penalty (i.e., when $\delta = 0$) the analysis for a fixed number of machines is valid also when machines are added. Specifically, in a dynamic setting in which machines are added or removed and migrations are free of cost, then the results known for classic load balancing games applied. In particular, the PoA assuming $\delta = 0$ is $2 - \frac{2}{m_0+m'+1}$ for a game with m' added machines and $2 - \frac{2}{m_1+1}$ for a game with removed machines and m_1 remaining machines. The proofs are identical to the proofs for a fixed number of machines. Thus, the difference between our results and the results for the classic load balancing game are due to the migration penalty.

2 Machines' Addition

In this section we study the scenario in which the systems' modification involves an addition of machines and uniform extension penalty is applied. Specifically, for a given parameter $\delta > 0$, if a job is assigned to a machine different than its original machine then its processing time is extended to be $p_j + \delta$. Recall that m_0, m' denote the initial and added number of machines, respectively. Let $m = M_0 + m'$ denote the resulting number of machines.

2.1 Equilibrium Existence and Computation

We first prove the existence of a NE in our model and suggest a specific form of best-response dynamics with linear convergence time.

Theorem 2.1 *Every instance of the load rebalancing game with added machines and uniform extension penalty admits at least one pure Nash equilibrium.*

Proof: The proof follows the proof for the load balancing game. For a given schedule s, let $(L_1, \ldots, L_m)$ be the sorted load vector corresponding to s. That is, L_i is the load on the machine that has the i-th highest load. If s is not a NE, then there exists a beneficial move to some job. We show that the sorted load vector obtained after performing a beneficial move is lexicographically smaller. This implies that a pure NE is reached after a finite number of beneficial moves.

Assume that job j can benefit by migrating from M_a to M_b. Clearly, $L_b < L_a$. The move decreases the load on M_a and increases the load on M_b. Before the move $L_b + p_j + \delta < L_a$, as otherwise, j would not benefit from the move. Combining this with the fact that the load on machines other than M_a, M_b is unchanged, we get that the number of machines with load at least L_a is decreasing. Therefore, the improvement step yields a sorted load vector that is lexicographically smaller than $(L_1, \ldots, L_m)$. ∎

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. As there is a finite number of possible configurations, the above proof implies that any best-response-dynamics is guaranteed to converge to a NE. In fact, our proof implies that even better response dynamics (in

which each player's move is beneficial, though not-necessarily the most beneficial), must converge to a NE.

The next question we consider is how many moves are required to reach a NE. The following result shows that, for any given initial assignment, there exists a short sequence of beneficial moves that leads to a NE. Assume that the jobs are sorted according to their processing length, that is, $p_1 \geq p_2 \geq \ldots \geq p_n$. The *max-length best response* policy activates the jobs one after the other according to the sorted order. An activated job j plays a best response, i.e., it moves to a machine that minimizes its cost (or remain on $s_0(j)$ if no beneficial move exists).

We show that after a single pass of *max-length best response* policy on the jobs, the system reaches a NE. While this result is valid also for the classic load balancing game [23], its proof for the load rebalancing game is more involved. We begin with the following observation.

Claim 2.2 *As long as each job moves at most once, the minimal load does not decrease.*

Proof: We show that the minimum load does not decrease as a result of any first move of a job j. Assume that j moves from M_a to M_b. Since this is the first move of job j, it holds that $M_a = s_0(j)$. Denote by L_i^0, L_i^1 the loads on machine M_i before and after the move of job j respectively. Denote by L_{min}^0 the minimal load before the move of job j. The move is beneficial for j, thus, $L_b^0 + p_j + \delta < L_a^0$. We show that $\min\{L_a^0, L_b^0\} \leq \min\{L_a^1, L_b^1\}$. Clearly, the load on any machine other than a,b does not change. Since M_b is the best response of j, $L_b^0 = L_{min}^0 = \min\{L_a^0, L_b^0\}$. In addition, $L_a^1 = L_a^0 - p_j$, $L_b^1 = L_b^0 + p_j + \delta < L_a^0$ and $L_a^0 - p_j > L_b^0 + \delta$. Thus, $\min\{L_a^0 - p_j, L_b^0 + p_j + \delta\} \geq \min\{L_b^0 + \delta, L_b^0 + p_j + \delta\} \geq \min\{L_a^0, L_b^0\}$. ∎

Theorem 2.3 *Let s_0 be any initial schedule of n jobs on m_0 machines. Assume that m' machines are added. Starting from s_0, the max-length best response policy reaches a pure Nash equilibrium after each job is activated at most once.*

Proof: A job j is said to be *satisfied* if it cannot reduce its cost by migrating to a different machine. By the definition of the cost function with migration penalty, j is satisfied if it is assigned to $s_0(j)$ and $L_{s_0(j)} \leq L_{i'} + p_j + \delta$ for every $i' \neq s_0(j)$, or if it is assigned to M_i, for some $i \neq s_0(j)$, $L_i \leq L_{i'} + p_j + \delta$ for every $i' \neq s_0(j)$, and $L_i \leq L_{s_0(j)} + p_j$. We show that once a job j was activated and played its best response, it never gets unsatisfied again.

Assume by contradiction that the claim is false and let j be the first job for which a *second* beneficial move exists. Let $M_a = s_0(j)$. Assume that on its first move j migrated from M_a to M_b. Job j might leave M_b only if one of the following conditions holds:

C_1 : For some machine $M_c \neq M_a$ it holds that $L_c \leq L_b - p_j - \delta$.

C_2 : $L_a \leq L_b - p_j$.

We show that none of these conditions hold. We first prove it assuming that the load on M_b does not increase, and then consider also the possibility that the load on M_b increases after j joins it. Denote by t the time in which job j moves from M_a to M_b, and let L^t_{min}, L^t_i denote the minimal load and the load on machine i, respectively, at time t.

Claim 2.4 *Conditions C_1 and C_2 do not hold as long as the load on M_b does not increase.*

Proof: Since j performs a best move, it must be that $L^t_{min} = L^t_b$. According to Claim 2.2, the minimal load does not decrease during the game, therefore, the load on each machine is at least L^t_{min}. This implies that condition C_1 does not hold and the only machine to which j might move to is M_a. Job j might move back to M_a if $L_a < L^t_{min} + \delta$. We show that this never happens. In order for the load on machine M_a to decrease, there must be a job j' that leaves it after the move of j. Since we assume that j is the first job that is migrating twice, it must be that $s_0(j') = M_a$. Let M_d be the machine to which j' moves, and let t' be the migration time. Before the move of j', $L^{t'}_a > L^{t'}_d + p_{j'} + \delta$. Since $L^{t'}_d \geq L^t_{min}$, $L^{t'}_a > L^t_{min} + p_{j'} + \delta$. After the move of j' (at time t'+1), $L^{t'+1}_a = L^{t'}_a - p_{j'} > L^t_{min} + \delta = L^t_b + \delta$. Since we assume that no job is added to M_b, it holds that $L^{t'+1}_b = L^t_b$. Thus, $L^{t'+1}_a > L^{t'+1}_b + \delta$ and condition C_2 does not hold. ∎

Claim 2.5 *Conditions C_1 and C_2 do not hold after any job k joins M_b.*

Proof: Denote by t' the time after the move of job k to M_b. $M_b \neq s_0(k)$ since we assume that j is the first job that is migrating twice. We know that $p_j \geq p_k$ because job j was activated before job k and jobs are activated in non-increasing order of their length. Therefore, for any machine M_i,

$$L^{t'}_b \leq L^{t'}_i + p_k + \delta \leq L^{t'}_i + p_j + \delta.$$

Thus, $L_b^{t'} - p_j - \delta \leq L_i^{t'}$ and condition C_1 does not hold. For M_a we show that condition C_2 does not hold, that is $L_a^{t'} \geq L_b^{t'} - p_j$. Recall that t'-1, t' are the times before and after the move of job k to M_b, respectively.

If k moves from M_a to M_b, then

$$L_a^{t'-1} > L_b^{t'-1} + p_k + \delta \tag{1}$$

After the move $L_a^{t'} = L_a^{t'-1} - p_k$ and $L_b^{t'} = L_b^{t'-1} + p_k + \delta$. Job j would benefit from migrating to M_a if $L_a^{t'} \leq L_b^{t'} - p_j$ which is equivalent to $L_a^{t'-1} - p_k + p_j < L_b^{t'-1} + p_k + \delta$. Since job j was activated before job k, we have $p_k \leq p_j$. Thus, $L_a^{t'-1} < L_a^{t'-1} - p_k + p_j$, and $L_a^{t'-1} < L_b^{t'-1} + p_k + \delta$, contradicting (1).

Next we consider the case in which job k moves to M_b from $M_c \neq M_a$. Job k prefers M_b over M_a, therefore, $L_a^{t'} + p_k + \delta \geq L_b^{t'}$. This implies that $L_a^{t'} \geq L_b^{t'} - p_k - \delta$. Therefore, condition C_2 does not hold and job j will not benefit from migrating back to M_a. ∎

We conclude that the max-length best response policy reaches a pure Nash equilibrium after each agent is activated at most once. ∎

2.2 Equilibrium Inefficiency

In this section we bound the *price of stability* and the *price of anarchy* of our game, distinguishing between various initial states and convergence methods. For the classic load balancing game, with no extension penalty, it is known that $PoA = 2 - \frac{2}{m+1}$. We show that in our model $PoS = 1$ and the PoA is not bounded by a constant. It can be arbitrary large if the schedule is not achieved by a sequence of improvement steps and bounded by $2 + \frac{m'+1}{m_0}$ if the schedule is achieved by a sequence of improvement steps. We also show that if the initial schedule is not a NE but the schedule is achieved by performing a sequence of improvement steps, the PoA is bounded by $m_0 + m'$.

Theorem 2.6 *The price of stability of the selfish load rebalancing game with job extension penalty is* 1.

Proof: It is easy to see that a beneficial move does not increase the makespan. Therefore, by preforming best-response starting from any optimal assignment, we reach a NE whose makespan is equal to the optimum. ∎

We turn to consider the price of anarchy (PoA). We distinguish between a modified schedule that is an arbitrary NE, and a NE that is reached by performing a sequence of beneficial moves. We denote by L^0_{max}, L_{max}, OPT the makespan of the initial schedule s_0, the makespan of the final NE s, and the minimal possible makespan, respectively. Let $P = \sum_j p_j$ denote the total initial length of the jobs. We first show that for an arbitrary NE, the PoA is unbounded. The bound is valid even if $m' = 0$ and the initial schedule is a NE.

Theorem 2.7 *When the NE is not necessary achieved by a sequence of beneficial moves, the PoA is unbounded.*

Proof: Given m', m_0, δ and r, we construct an instance for which the PoA is r. Let ε be a small constant such that $r = \frac{\varepsilon+\delta}{\varepsilon}$. Assume that in the initial schedule, s_0, there is a single job of length ε on each machine in M_0 (see Figure 1(a)). Independent of the number of added machines,

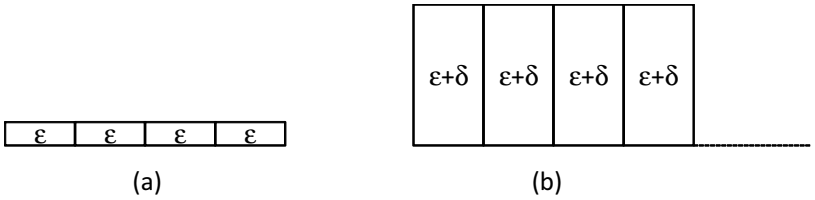

Figure 1: An instance achieving unbounded PoA. (a) the initial assignment, (b) the worst NE.

m', a schedule in which each job is assigned to a machine in M_0 different from $s_0(j)$ and each machine in M_0 is assigned a single job, is a NE (see Figure 1(b)). In this schedule, all jobs have the same cost of $\varepsilon + \delta$. It is easy to verify that this schedule is a NE. The optimal schedule is identical to s_0, where all jobs have the same cost of ε. The PoA is $\frac{\varepsilon+\delta}{\varepsilon} = r$. ∎

We turn to consider the more realistic scenario in which the NE is reached by performing beneficial moves, starting from a NE schedule s_0. Our analysis depends on the parameters m_0 and m'. We provide an upper bound that is tight when $m' \bmod m_0 = 1$, and almost tight for any other case.

Theorem 2.8 *The price of anarchy in the selfish rebalancing game with job extension penalty is at most* $2 + \frac{m'-1}{m_0}$.

Proof: Recall that $P = \sum_j p_j$ denotes the total initial length of the jobs. Let j be the shortest job on the most loaded machine in s_0. Since s_0 is a NE, it holds that the gap between the maximal and minimal load is at most p_j. Therefore, $m_0 L^0_{max} \leq P + (m_0 - 1)p_j$. Implying, $L^0_{max} \leq \frac{P}{m_0} + \frac{m_0-1}{m_0} p_j$.

Also, since s is achieved by performing beneficial moves, it must be that $L_{max} \leq L^0_{max}$. Clearly, even with no migration penalties, for the minimal possible penalty it holds that $OPT \geq \frac{P}{m_0+m'}$. Also, $OPT \geq p_j$. We get that the price of anarchy is bounded by

$$\frac{L_{max}}{OPT} \leq \frac{\frac{P}{m_0} + \frac{m_0-1}{m_0} p_j}{OPT} \leq \frac{\frac{P}{m_0}}{\frac{P}{m_0+m'}} + \frac{\frac{m_0-1}{m_0} p_j}{p_j} \leq \frac{m_0 + m'}{m_0} + \frac{m_0 - 1}{m_0} = 2 + \frac{m' - 1}{m_0}.$$

■

We show that this bound is tight for several combinations of m_0, m', and almost tight for any other combination.

Theorem 2.9 *For any number of machines m_0, for any integer $k > 0$, and for any $\rho > 0$, there exists an input with $m' = km_0 + 1$ added machines, for which $PoA > 2 + \frac{m'-1}{m_0} - \rho$.*

Proof: Given ρ, m_0, k, let $m' = km_0 + 1$. Let B be an integer such that $\rho \geq \frac{k+2}{B+1}$. In addition, let $\varepsilon = \frac{1}{(k+1)m'B}$ and $\delta = 1 - \varepsilon$.

The set of jobs includes $m' + m_0 = (k+1)m_0 + 1$ jobs of length B, and $1/\varepsilon = (k+1)m'B$ jobs of length ε. In the initial assignment, a single machine is assigned $k+2$ jobs of length B and each of the other $m_0 - 1$ machines is assigned $k+1$ jobs of length B, as well as some jobs of length ε, such that the ε-jobs are assigned in a balanced way and the assignment is a NE. Note that the load on the first machine is $(k+2)B$ and the load on each of the other $m_0 - 1$ machines is between $(k+1)B$ and $(k+1)B + 1$.

We demonstrate the construction of the lower bound in Figure 2. In this instance $m_0 = 3$ and $k = 1$ (implying $m' = 4$). The initial assignment is given in Figure 2(a).

Assume that m' machines are added and improving steps are performed. A possible NE (see Figure 2(b)) is a one in which the long jobs remain on M_0 and every new machine is assigned $(k+1)B$ jobs of length ε. The load on the first machine remains $(k+2)B$. The load on each of the other $m_0 - 1$ machines of M_0 is $(k+1)B$. The load on every new machine is $(k+1)B(\delta + \varepsilon) =$

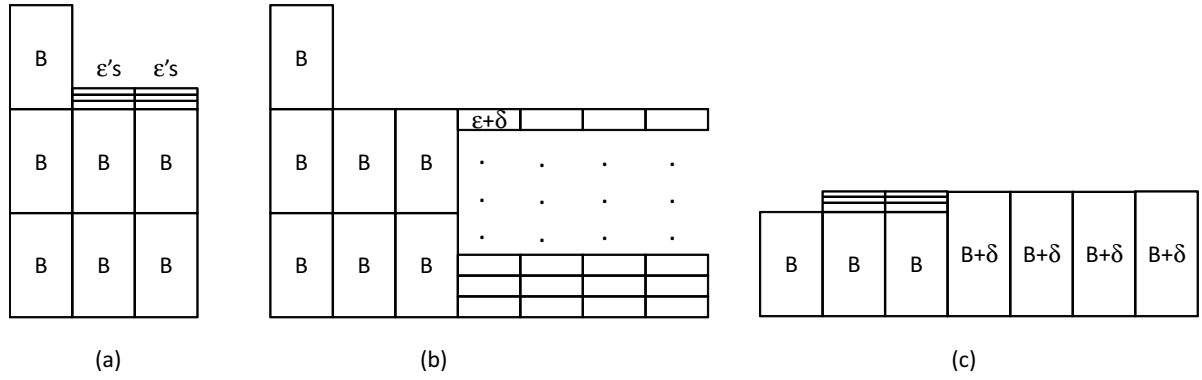

Figure 2: An instance achieving the maximal possible PoA. (a) the initial assignment, (b) the worst NE, and (c) the best NE.

$(k+1)B$. The maximum load is $(k+2)B$ - achieved on the first machine. This assignment is a NE as the shortest job on the most loaded machine has length B - which is exactly the gap from the load on all other machines. Also, the other machines are perfectly balanced, therefore no migrations are beneficial.

On the other hand, the following is an optimal assignment (see Figure 2(c)): One job of length B migrates to each of the new machines. The other m_0 jobs of length B as well as all jobs of length ε remain on the original machines M_0. The maximal load on M_0 is at most $B+1$. The load on every new machine is $B+\delta < B+1$.

The ratio between the maximal loads of the two assignments is $\frac{(k+2)B}{B+1}$. The value of B was selected such that this is more than $2+k-\rho = 2+\frac{m'-1}{m_0}-\rho$. ∎

Let $m' = km_0+\alpha$ for integers k and $\alpha < m_0$. By Theorem 2.8, we have that the PoA is at most $2+\frac{m'-1}{m_0} = 2+\frac{m_0k+\alpha-1}{m_0} = k+2+\frac{\alpha-1}{m_0}$. For $\alpha = 1$ and any k, by theorem 2.9, the bound is tight. We turn to consider other values of α and k. Assume first that $k=0$, that is, the number of added machines is smaller than the number of initial machines.

Theorem 2.10 *For any $1 < m' < m_0$ and $\rho > 0$, there exists an input such that PoA $> 2+\frac{3m'-m_0-2}{2m_0+1}-\rho$.*

Proof: Given $\rho, 1 < m' < m_0$, let B be an integer such that $\rho \geq \frac{6(m_0+m')}{(B+1)(2m_0+1)}$. In addition, let $\delta = 1-\varepsilon$ and $\varepsilon = \frac{1}{2Xm'}$ where $X = B \cdot \frac{m_0+m'}{2m_0+1}$.

The input consists of m_0+m' long jobs of size X, m_0+m' medium jobs of size $Y = X \cdot \frac{m_0-m'+1}{m_0+m'}$, and $\frac{1}{\varepsilon}$ tiny jobs of size ε. In the initial assignment, one machine, M_1, is assigned three jobs of length X. Additional $m'-2$ machines are assigned two jobs of length X each, and each of the remaining $m_0-m'+1$ machines is assigned a single job of length X. In additions, each of these $m_0-m'+1$ machines is assigned $\frac{m_0+m'}{m_0-m'+1}$ medium jobs[1]. Note that $Y \cdot \frac{m_0+m'}{m_0-m'+1} = X$, therefore, the total load on every machine $M_2, \ldots, M_{m_0}$ is $2X$. Finally, the tiny jobs are assigned in a balanced way on $M_2, \ldots, M_{m_0}$.

We demonstrate the construction of the lower bound in Figure 3. In this instance $m_0 = 4$ and $m' = 3$. The initial assignment is given in Figure 3(a). It is easy to verify that this assignment is a NE. The most loaded machine, M_1, has load $3X$. All other machines are balanced and have load at least $2X$. Since the shortest job on M_1 has length X, no job has a beneficial move.

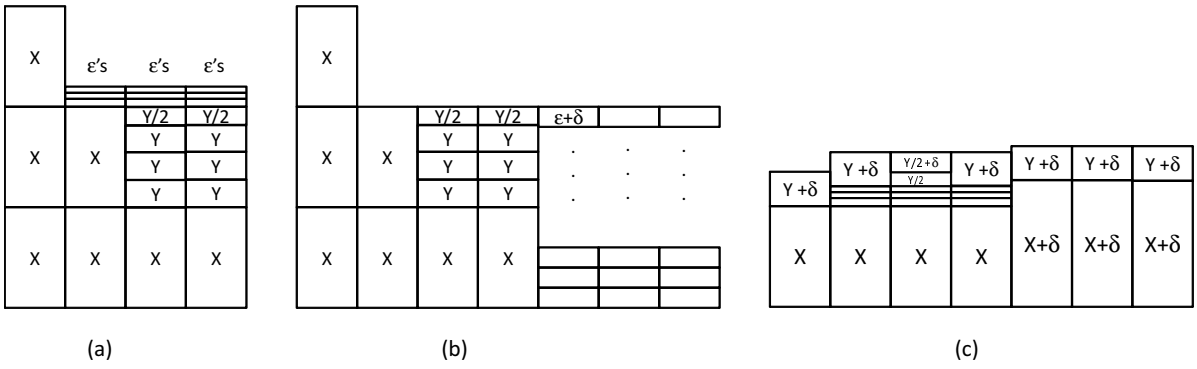

Figure 3: An instance achieving a high PoA for $m' < m_0$. (a) the initial assignment, (b) the worst NE, and (c) the best NE.

Assume that m' machines are added and improving steps are performed. A possible NE is a one in which the long and the medium-size jobs remain on M_0 and every new machine is assigned $\frac{1}{m'\varepsilon} = 2X$ tiny jobs. The load on the first machine is $3X$. The load on each of the other m_0-1 machines of M_0 is $2X$. The load on every new machine is $2X(\delta+\varepsilon) = 2X$. The maximum load is $3X$ - achieved on the first machine. This assignment is a NE as the shortest job on the most loaded machine has length X - which is exactly the gap from the load on all other machines. Also,

[1]If $\frac{m_0+m'}{m_0-m'+1}$ is not an integer, it is possible to replace at most $m_0-m'+1$ medium jobs each by two jobs whose total size is Y in a way that the load on the machines is balanced (see in Figure 3(a)).

the other machines are perfectly balanced, therefore no migrations are beneficial.

On the other hand, a better possible assignment results from the following migrations: The tiny jobs remain on M_0. Every new and original machine is assigned one long job and one medium job. The total load on every new machine is $X + Y + 2\delta = B + 2\delta < B + 2$. The total load on every original machine is at most $X + Y + \delta + 1 < B + 2$. The addition of δ is due to the migration of a medium job, the addition of 1 is due to the tiny jobs[2].

The ratio between the maximal loads of the two assignments is $\frac{3X}{B+2}$. The value of B was selected such that this is more than $2 + \frac{3m'-m_0-2}{2m_0+1} - \rho$. ∎

For $k \geq 1$ and $\alpha > 1$, the worst PoA we were able to get is the following:

Theorem 2.11 *For $m' > m_0$ and every $\rho > 0$, there exists an input such that $PoA > k + 2 + \frac{(\alpha-1)(k+3)-m_0+1}{m_0(k+2)+1} - \rho$.*

Proof: Given $\rho, m_0, m' = km_0 + \alpha, k > 0, \alpha < m_0$, let B be an integer such that $\rho \geq \frac{2(k+3)(m_0+m')}{(B+2)(m_0(k+2)+1)}$. In addition, let $\delta = 1 - \varepsilon$ and $\varepsilon = \frac{1}{(k+2)Xm'}$ where $X = B \cdot \frac{m_0+m'}{(k+2)m_0+1}$.

The input consists of $m_0 + m'$ long jobs of size X, $m_0 + m'$ medium jobs of size $Y = X \cdot \frac{1-\alpha}{m_0+m'}$, and $\frac{1}{\varepsilon}$ tiny jobs of size ε. In the initial assignment, one machine, M_1 is assigned $k + 3$ jobs of length X. Additional $\alpha - 1$ machines are assigned $k + 2$ jobs of length X each, and each of the remaining $m_0 - \alpha$ machines is assigned a single job of length X. In additions, each of these $m_0 - \alpha$ machines is assigned $\frac{m_0+m'}{m_0-\alpha}$ medium jobs[3] Note that $Y \cdot \frac{m_0+m'}{m_0-\alpha} = X$, therefore, the total load on every machine $M_2, \ldots, M_{m_0}$ due to long and medium jobs is $(k + 2)X$. Finally, the tiny jobs are assigned in a balanced way on $M_2, \ldots, M_{m_0}$.

We demonstrate the construction of the lower bound in Figure 4. In this instance $m_0 = 3$ and $m' = 5$, thus, $k = 1$ and $\alpha = 2$. The initial assignment is given in Figure 4(a). It is easy to verify that this assignment is a NE. The most loaded machine, M_1, has load $(k + 3)X$. All other machines are balanced and have load at least $(k + 2)X$. Since the shortest job on M_1 has length X, no job has a beneficial move.

[2]If some medium jobs were replaced by two jobs, it is possible to 'unite' these two parts in the assignment. It would still hold that a single migrating job is assigned on each initial machine.

[3]If $\frac{m_0+m'}{m_0-\alpha}$ is not an integer, it is possible to replace at most $m_0 - \alpha$ medium jobs each by two jobs whose total size is Y, in a way that the load on the machines is balanced.

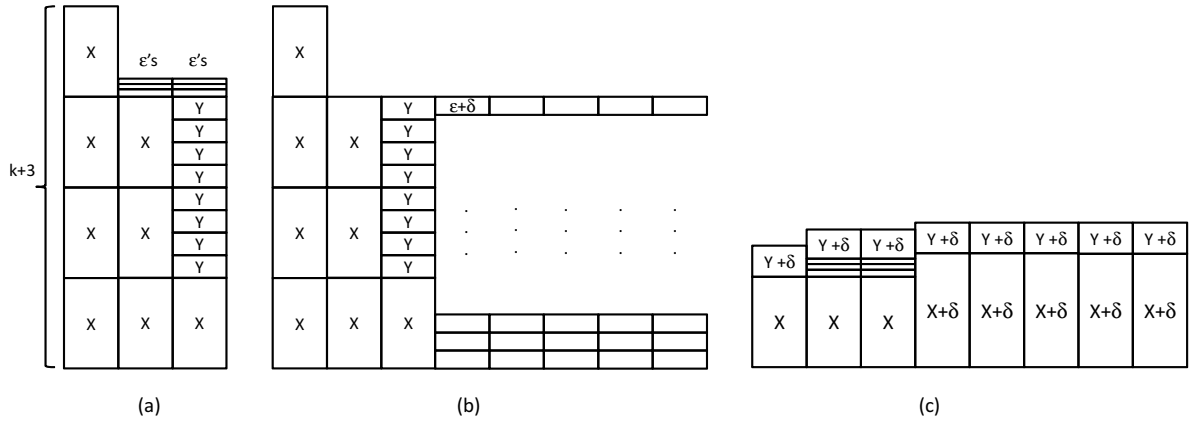

Figure 4: An instance achieving a high PoA for $k = 1$, $\alpha 21$. (a) the initial assignment, (b) the worst NE, and (c) the best NE.

Assume that m' machines are added and improving steps are performed. A possible NE is a one in which the long and the medium-size jobs remain on M_0 and every new machine is assigned $\frac{1}{m'\varepsilon} = (k+2)X$ tiny jobs. The load on the first machine is $(k+3)X$. The load on each of the other $m_0 - 1$ machines of M_0 is $2X$. The load on every new machine is $(k+2)X(\delta+\varepsilon) = (k+2)X$. The maximum load is $(k+3)X$ - achieved on the first machine. This assignment is a NE as the shortest job on the most loaded machine has length X - which is exactly the gap from the load on all other machines. Also, the other machines are perfectly balanced, therefore no migrations are beneficial.

On the other hand, a better possible assignment results from the following migrations: The tiny jobs remain on M_0. Every new and original machine is assigned one long job and one medium job. The total load on every new machine is $X + Y + 2\delta = B + 2\delta < B + 2$. The total load on every original machine is at most $X+Y+\delta+1 < B+2$. The addition of δ is due to the migration of a medium job, the addition of 1 is due to the tiny jobs[4].

The ratio between the maximal loads of the two assignments is $\frac{(k+3)X}{B+2}$. The value of B was selected such that this is more than $k + 2 + \frac{(\alpha-1)(k+3)-m_0+1}{m_0(k+2)+1} - \rho$. ∎

[4]If some medium jobs were replaced by two jobs, it is possible to 'unite' these two parts in the assignment. It would still hold that a single migrating job is assigned on each initial machine.

The lower bounds in Theorems 2.10 and 2.11 do not match the upper bound in Theorem 2.8. We believe that the upper bound can be reduced when $\alpha \neq 1$.

Finally, we analyze the PoA for arbitrary initial assignment.

Theorem 2.12 *If the initial assignment is not necessary a NE, and the modified schedule is reached by performing improvement steps, then the PoA is at most* $m_0 + m'$.

Proof: Clearly, in the initial assignment, $L^0_{max} \leq \sum_j p_j$. Since improvement steps are preformed, we have $L_{max} \leq L^0_{max}$. Also, the makespan of the modified schedule is at least $OPT \geq \frac{\sum_j p_j}{m_0+m'}$. Therefore, $PoA \leq \frac{L_{max}}{OPT} \leq \frac{L^0_{max}}{OPT} \leq \frac{\sum_j p_j}{\sum_j p_j/m_0+m'} \leq m_0 + m'$. ∎

This bound is tight as implied by the following theorem.

Theorem 2.13 *For any number of machines* m_0, m' *and for any* $\rho > 0$, *there exists a non-NE schedule on* m_0 *machines, such that when* m' *machines are added and improvement steps are preformed, the PoA is at least* $m_0 + m' - \rho$.

Proof: Given ρ, m_0, m'. Let B be an integer such that $\rho \geq \frac{m_0+m'-1}{B+1}$. In addition, let $\varepsilon = \frac{1}{(m_0+m')(m_0+m'-1)B}$ and $\delta = 1 - \varepsilon$.

In the initial assignment, a single machine is assigned $m_0 + m'$ jobs of length B and $\frac{1}{\varepsilon}$ jobs of length ε. The other machines in M_0 are empty. Thus, the load on the first machine is $(m_0 + m')B + 1$ and the load on each of the other $m_0 - 1$ machines is 0.

We demonstrate the construction of the lower bound in Figure 5. The initial assignment is given in Figure 5(a). In this instance $m_0 = 3$ and $m' = 2$.

Assume that m' machines are added and improvement steps are performed. A possible NE (see Figure 5(b)) is a one in which the long jobs remain on the first machine and every other machine is assigned $(m_0 + m')B$ jobs of length ε. The load on the first machine is $(m_0 + m')B$. The load on each of the other $m_0 + m' - 1$ machines is also $(m_0 + m')B(\varepsilon + \delta) = (m_0 + m')B$. Since the load on all the machines is balanced, the schedule is a NE.

On the other hand, the following is an optimal assignment (see Figure 5(c)): One job of length B is migrating to each of the empty $m_0 + m' - 1$ machines. One job of length B and all jobs of length ε remain on the original first machine. The load on the first machine M_0 is $B + 1$. The load on every other machine is $B + \delta < B + 1$.

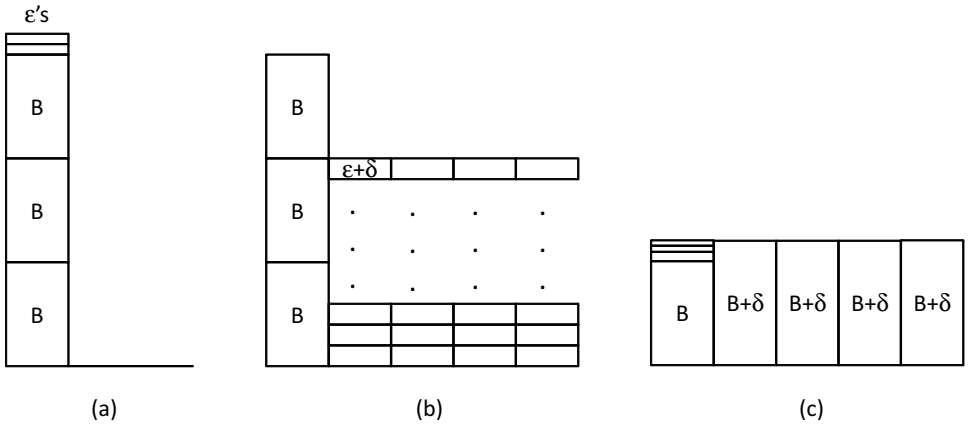

Figure 5: An instance achieving the maximal possible PoA. (a) the initial assignment (Not a NE), (b) the worst NE, and (c) the best NE.

The ratio between the maximal loads of the two assignments is $\frac{(m_0+m')B+1}{B+1}$. The value of B was selected such that this is at least $m_0 + m' - \rho$. ■

3 Machines' Removal

In this section we study the scenario in which the systems' modification involves removal of machines, and migrations are associated with extension penalty. Recall that M_0, M' denote the sets of initial and removed machines, respectively. Let $M_1 = M_0 \setminus M'$ denote the set of remaining machines. Let m_0, m', m_1 denote the corresponding number of machines, that is $m_1 = m_0 - m'$. Every job originally assigned to a machine from M' must be reassigned. As a result, additional jobs might also be interested in migrating.

Throughout this section, we assume that the initial schedule, s_0, is a Nash Equilibrium. The last result in this section, Theorem 3.9, considers the case where s_0 is not a NE.

3.1 Equilibrium Existence and Computation

We prove the existence of a NE and analyze the convergence rate of several policies. We first show that unlike the 'adding machines' scenario (studied in Section 2), when machines are removed, a single phase of max-length best-response might not end up in a NE.

Consider the initial schedule on $m_0 = 4$ machines given in Figure 6(a). Assume that the two right machines are removed and that $\delta = 1$.

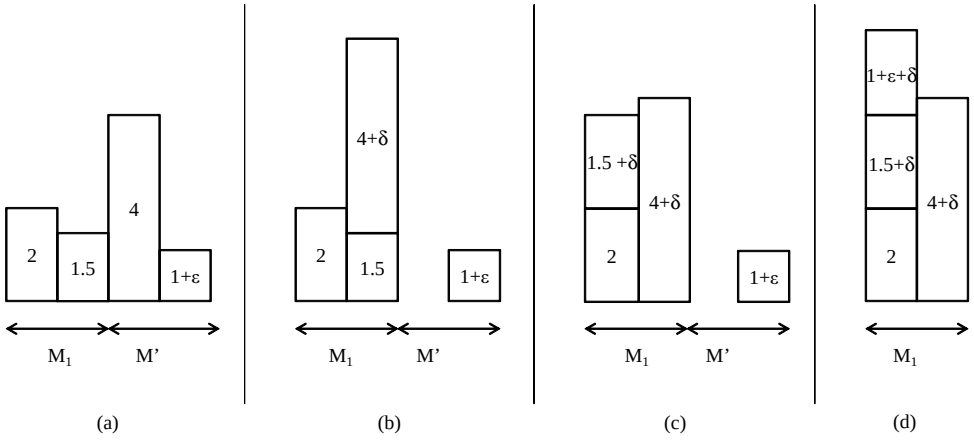

Figure 6: An example of single phase *max-length best-response* that does not converge to a NE.

The job of length 4 is the first to be activated by max-length best-response. It must move to a machine in M_1 and its best-response is to join the machine with load 1.5 (Figure 6(b)). The

job of length 2 is not interested in moving. Next, the job of length 1.5 moves (Figure 6(c)), and finally, the job of length $1+\varepsilon$. Figure 6(d) gives the schedule after one phase of max-length best-response. This schedule is not NE, as the job of length 1.5 would benefit from returning to its initial location. Thus, a single phase of *max-length best-response* is not guaranteed to end up in a NE assignment.

3.1.1 The *better-response* policy

In the *better-response* policy, all jobs are activated in an arbitrary order. When activated, each job migrates if it is on M' or if it can improve its cost. For every job j, if $s_0(j) \in M'$, j must be activated at least once and move to a machine in M_1 and be extended. Other jobs are extended if they leave their original machine. Jobs might be activated several times. Jobs must not migrate into machines in M'.

We first show that *better-response* policy terminates in a NE assignment.

Theorem 3.1 *The* better-response *police leads to a NE assignment for every instance of the load rebalancing game with removed machines and uniform extension penalty.*

Proof: Let s_1 be the schedule at the time after the last job migrated from M'. Thus, in s_1 all the jobs are scheduled on M_1. Let $(L_1, \ldots, L_{m_1})$ be the sorted load vector corresponding to s_1. That is, L_i is the load on the machine that has the i-th highest load. If s_1 is not a NE, then there exists a beneficial move to some job. We show that the sorted load vector obtained after performing a beneficial move is lexicographically smaller. This implies that a pure NE is reached after a finite number of beneficial moves.

Assume that job j can benefit by migrating from M_a to M_b. The move decreases the load on M_a and increases the load on M_b. Before the move $L_b + p_j + \delta < L_a$ if $M_b \neq s_0(j)$ or $L_b + p_j < L_a$ if $M_b = s_0(j)$, as otherwise, j would not benefit from the move. In particular $L_a > L_b$. Combining this with the fact that the load on machines other than M_a, M_b is not changed, we get that the number of machines with load at least L_a is decreasing. Therefore, the improvement step yields a sorted load vector that is lexicographically smaller than $(L_1, \ldots, L_{m_1})$. ∎

3.1.2 The *two-phase better-response* policy

The *two-phase better-response* policy consists of two phases. In the first phase all the jobs that are assigned to machines in M' are activated. Each job is activated once and performs its best move. In the second phase all the jobs (now assigned to M_1) are activated in an arbitrary order. In the first phase each job performs its best move and in the second phase each job performs an improvement step (not necessary the best). The second phase terminates when there are no more beneficial moves. In the first phase, activated jobs must migrate to a machine in M_1 and be extended. In the second phase, a job j migrates if a beneficial move exists and is extended only if the migration is into a machine different from $s_0(j)$. Jobs must not migrate into machines in M'. A special case of this policy is the *two-phase best-response* policy, in which in both phases an activated job performs its best response.

As this is a specific application of the *better-response* policy described above, the *two-phase better-response* policy is guaranteed to terminate in a NE assignment.

3.1.3 The *two-phase max-length best-response* policy

The *two-phase max-length best-response* policy is a special application of the *two-phase better-response* policy. In the first phase all the jobs assigned to machines in M' are activated. In the second phase all the jobs (now assigned to M_1) are activated in a non-increasing order of processing time p_j without taking into account the extension penalty. An activated job performs its best response and is extended if it migrates to a machine it was not assigned on in s_0. Note that each job is activated exactly once in the second phase. Jobs must not migrate into machines in M'.

As demonstrated in Figure 6, a single phase of *max-length best-response* policy might not end up in a NE. As we show below, convergence to a NE is guaranteed by the *two-phase max-length best-response* policy.

Theorem 3.2 *The* two-phase max-length best-response *policy leads to a pure NE schedule.*

Proof: We first show the following claim.

Claim 3.3 *The minimal load does not decrease during the second phase.*

Proof: We show that the minimal load does not decrease as a result of any move of a job j in the second phase. Assume that j moves from M_a to M_b. Denote by L_i^0, L_i^1 the loads on machine M_i before and after the move of job j, respectively. We show that $\min\{L_a^0, L_b^0\} \leq \min\{L_a^1, L_b^1\}$. Clearly, the load on any machine other than a,b does not change.

If $s_0(j) \in M_1$ then $M_a = s_0(j)$. The move is beneficial for j, thus, $L_b^0 + p_j + \delta < L_a^0$. Since M_b is the best response of j, $\min\{L_a^0, L_b^0\} = L_b^0$. In addition, $L_a^1 = L_a^0 - p_j$, $L_b^1 = L_b^0 + p_j + \delta < L_a^0$, thus, $L_a^0 - p_j > L_b^0 + \delta$. We get, $\min\{L_a^1, L_b^1\} = \min\{L_a^0 - p_j, L_b^0 + p_j + \delta\} \geq \min\{L_b^0 + \delta, L_b^0 + p_j + \delta\} \geq \min\{L_a^0, L_b^0\}$.

If $s_0(j) \in M'$, then $L_b^0 + p_j + \delta < L_a^0$. Therefore, $\min\{L_a^0, L_b^0\} = L_b^0 \leq \min\{L_a^0 - p_j - \delta, L_b^0 + p_j + \delta\} = \min\{L_a^1, L_b^1\}$. ∎

We show that during the second phase, once a job j was activated and perform its best response, it never has a beneficial move again. Assume by contradiction that the claim is false and let j be the first job for which a *second* beneficial move exists. We distinguish between two cases:

1. $s_0(j) \in M'$. Assume that on its first move in the second phase, j migrated from M_a to M_b. Job j might leave M_b only if for some machine M_c it holds that $L_c \leq L_b - p_j - \delta$.

   By Claim 3.3, If the load on M_b does not increase after the join of job j, there is no machine M_c for which $L_c \leq L_b - p_j - \delta$.

   We show that even if the load on M_b increases after the join of job j, there is no machine M_c for which $L_c \leq L_b - p_j - \delta$. Assume that the load on M_b increased due to a join of job k. Denote by t' the time after the move of job k to M_b. $M_b \neq s_0(k)$ since we assume that j is the first job that is migrating twice in the second phase. Since jobs are activated in non-increasing order of their length it holds that $p_j \geq p_k$. Therefore, for any machine M_c,

   $$L_b^{t'} \leq L_c^{t'} + p_k + \delta \leq L_c^{t'} + p_j + \delta.$$

   Thus, $L_b^{t'} - p_j - \delta \leq L_c^{t'}$.

2. $s_0(j) \in M_1$. The proof for this case is identical to the proof of Theorem 2.3.

∎

We conclude that the *two-phase max-length best response* policy leads to a pure NE schedule in the case of machine removal. The convergence time is linear - at most $2n$ migrations are preformed.

3.2 Equilibrium Inefficiency

In this section we analyze the *price of stability* and the *price of anarchy* with various initial states and convergence algorithms. We show that the results differ from the classical load balancing game as well as from the machines' addition scenario.

We note that by Theorem 2.6, the *price of stability* of the selfish load rebalancing game with removed machines and any job extension penalty is 1. As shown in Theorem 2.7, for machines' addition, the PoA is unbounded if the NE is not reached by performing beneficial migrations. The same example (or a similar one, if we add a request that the removed machines are non-empty) is valid also when the modification involves machines' removal. On the other hand, by assuming the NE is reached by preforming the *better-response* policy, we can bound the PoA. Let n' be the number of jobs assigned to M' in s_0.

Observation 3.4 *Along the application of better-response policy,* $L_{max} \leq \sum_j p_j + n'\delta$.

Proof: The maximal initial load, L^0_{max}, on machines in M_1 is at most $\sum_{j|s_0(j)\in M_1} p_j$. A beneficial move of jobs that are scheduled on M_1 in s_0 does not increase the maximal load as otherwise the move is not beneficial. A move of a job j originally assigned to a removed machine might increase the maximal load by $p_j + \delta$. Thus, the maximal increase of the load on a single machine is $\sum_{j|s_0(j)\in M'}(p_j + \delta)$, resulting in maximal load at most $\sum_j p_j + n'\delta$. ∎

Theorem 3.5 *The price of anarchy of a NE that is reached by preforming the* better-response *policy is at most* m_1.

Proof: Let $P = \sum_j p_j + n'\delta$. According to Observation 3.4, P is the maximal load that can be reached. Since n' job extensions are inevitable, we have $OPT \geq \frac{P}{m_1}$. Therefore, $PoA \leq \frac{P}{P/m_1} = m_1$. ∎

We show that the above analysis is tight for every $m_1 \leq m'$, such that $m_1|m'$.

Theorem 3.6 *For every $m_1 \leq m'$, $m_1|m'$, and any $\rho > 0$. There exists an instance with m' removed machines and m_1 remaining machines for which the PoA of the game, assuming NE is reached by performing the* better-response *policy is at least $m_1 - \rho$.*

Proof: Given $\rho, m_1 \leq m'$, $m_1|m'$, let $B = \frac{m_1(m_1-1)}{m'\rho} - \frac{m_1}{m'}$. Let $\varepsilon = \frac{1}{Bm'(m_1-1)}$. Also, let $\delta = 1 - \varepsilon$ and M_a be the first machine in M_1.

Consider the schedule s_0 in which there are $\frac{1}{\varepsilon}$ jobs of length ε, forming load 1 on M_a, and a single job of length 1 on every other machine in M_1. On every machine in M' there is a single job of length $B - \delta = B - 1 + \varepsilon$. Note that s_0 is a NE.

We demonstrate the construction of the lower bound in Figure 7. In this instance $m_0 = 3$ and $m' = 3$. The initial assignment is given in Figure 7(a).

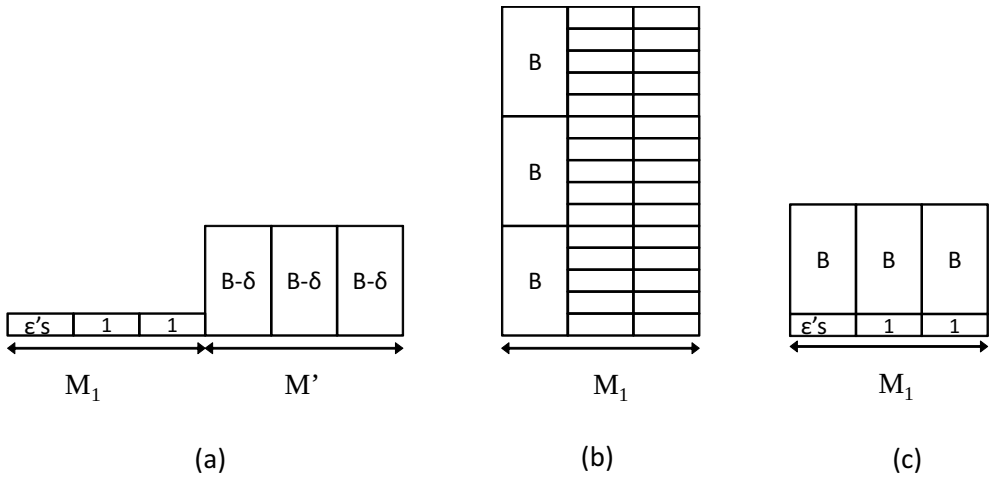

Figure 7: An instance achieving the maximal possible PoA by performing better-response policy. (a) the initial assignment, (b) the worst NE, and (c) the best NE.

In a possible sequence of moves, all jobs from M' move to M_a. After each move of a job j from M' to M_a, some ε-jobs that were assigned on M_a move to the other machines in M_1. The amount of ε-jobs that migrate to each of the remaining machines in M_1 is $B - 1$. After $(B-1)(m_1-1)$ ε-jobs migrate, each machines' load increases by $B-1$ and the machines in M_1 are balanced. After M' such iterations, we reach a NE and the makespan is $L_{max} \leq \varepsilon\frac{1}{\varepsilon} + (B - \delta + \delta)m' = 1 + Bm'$. The final schedule s is shown in Figure 7(b).

In an optimal schedule, since $m_1|m'$ the $B-\delta$ jobs spread equally on M_1. The load on each machine is $OPT = 1 + \frac{m'}{m_1}B = \frac{m_1+Bm'}{m_1}$ (see Figure 7(c)). The ratio between the maximal load in the two schedules is $\frac{1+Bm'}{\frac{m_1+Bm'}{m_1}} = m_1 - \frac{m_1(m_1-1)}{m_1+Bm'}$. The value of B was selected such that this is at least $m_1 - \rho$. ∎

While the PoA for arbitrary better-response is m_1, a better bound can be shown if the NE is reached by the *two-phase better-response* policy.

Theorem 3.7 *The PoA assuming that s_0 is a NE and the modified NE is reached by the* two-phase better-response *policy is at most* $2 - \frac{1}{m_1}$.

Proof: Denote by L^1_{max}, L^2_{max} the maximal load on machines in M_1 after the first and the second phase respectively. Since only beneficial moves are preformed during the second phase, we have $L^2_{max} \leq L^1_{max}$. Thus, $L_{max} = L^2_{max} \leq L^1_{max}$. We bound L^1_{max} as follows. Let M_a be a machine with load L^1_{max} after the first phase. We distinguish between two cases.

L^1_{max} is determined by a single job j.

1. If $L^1_{max} = p_j$ for some job j, then j is the longest job and $s_0(j) \in M_1$. Since $OPT \geq \max_k p_k = p_j$, then the PoA in this case is 1.
2. If $L^1_{max} = p_j + \delta$, then $s_0(j) \in M'$. Clearly, j must migrate in any assignment, thus $OPT \geq p_j + \delta$, implying $PoA = 1$.

L^1_{max} is determined by two or more jobs.

1. If all the jobs on M_a were assigned to M_a also in s_0, then $L^1_{max} \leq L^0_{max}$. By the PoA bound on regular scheduling game $L^0_{max} \leq (2 - \frac{1}{m_1})OPT^0$. Also, $OPT^0 \leq OPT$ because the processing time of some of the jobs increased while the number of machines decreased. Therefore, $L^1_{max} \leq (2 - \frac{1}{m_1})OPT$.
2. If some jobs on M_a are extended, let j be the shortest extended job on M_a. Let $P = \sum_j p_j + n'\delta$. Since we consider the maximal load after the first phase, $s_0(j) \in M'$ and $OPT \geq p_j + \delta$.

   Since j performed its best move from M' in the first phase, then for every machine i, $L_a \leq L_i + p_j + \delta$ at the time of the move of job j. Since the minimal load on machines in M_1 does not decrease during the first phase, j would not have a beneficial move at the end of the first phase. Thus, $L^1_{max}m_1 \leq P + (m_1 - 1)(p_j + \delta)$, then

$L^1_{max} \leq \frac{P}{m_1} + \frac{m_1-1}{m_1}(p_j+\delta) \leq OPT + \frac{m_1-1}{m_1}OPT \leq \frac{2m_1-1}{m_1}OPT$. We conclude that the $PoA \leq \frac{2m_1-1}{m_1} = (2-\frac{1}{m_1})OPT$.

In both cases, we get $L_{max} \leq L^1_{max} \leq (2-\frac{1}{m_1})OPT$. ∎

The above analysis is tight even for the *two-phase best-response* policy.

Theorem 3.8 *For any $m_1 > 1, m' > 2$, $\rho > 0$, there exists an initial schedule for which the PoA of a schedule achieved by the* two-phase best response *policy is at least* $2 - \frac{1}{m_1} - \rho$.

Proof: Given ρ, let $z = \lfloor \frac{m_1-1}{m'-1} \rfloor$. In addition, let $\varepsilon \leq \frac{\rho m_1}{z(2-\rho)}$ so that $z\varepsilon \leq 1$ and $\delta = 1-\varepsilon$.

In the initial assignment, $m_1 - 1$ machines in M_1 are assigned a single job of length $m_1 - 1$ and one machine, M_a, in M_1, is assigned z jobs of length ε. The first machine in M' is assigned a single job of length $m_1 - \delta$ and on each of the other machines in M' there are z or $z+1$ jobs of length ε, such that there are $m_1 - 1$ jobs of length ε on all M' machines. This schedule guarantees that the jobs of length ε are balanced and do not have a beneficial move. The longer jobs are also stable since each is assigned to a dedicated machine. Therefore, the initial schedule is a NE.

We demonstrate the construction of the lower bound in Figure 8. In this instance $m_0 = 3$ and $m' = 2$. The initial assignment is given in Figure 8(a).

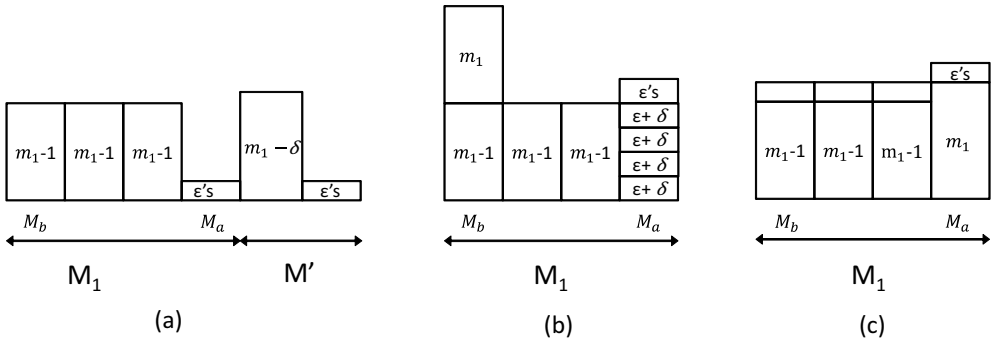

Figure 8: An instance achieving the maximal possible PoA by performing the two-phase best-response policy. (a) the initial assignment, (b) the worst NE, and (c) the best NE.

Assume that m' machines are removed and *two-phase best-response* policy is performed. A possible NE is a one in which, in the first phase, the ε-jobs migrate to M_a and the job of length $m_1 - \delta$ migrates to a different machine M_b in M_1 (see Figure 8(b)). The load on M_a is $(m_1 -$

$1)(\varepsilon+1-\varepsilon)+z\varepsilon = m_1-1+z\varepsilon$. The load on M_b is $m_1-1+(m_1-1+\varepsilon+1-\varepsilon) = 2m_1-1$. In the second phase no job migrates since the load on the other m_1-2 machines of M_1 is m_1-1. Therefore, $L_{max} = 2m_1-1$.

On the other hand, the following is an optimal assignment (see Figure 8(c)): The long job on M' migrates to M_a and each of the ε-jobs migrates to a different machine in M_1. The load on M_a is $(m_1-1+\varepsilon+1-\varepsilon)+z\varepsilon = m_1+z\varepsilon$. The load on each other machine in M_1 is $m_1-1+(\varepsilon+1-\varepsilon) = m_1$. The ε-jobs on M_a do not want to migrate because it will not improve their cost.

The ratio between the maximal loads of the two assignments is $\frac{2m_1-1}{m_1+z\varepsilon} = 2-\frac{1}{m_1+z\varepsilon}-\frac{2z\varepsilon}{m_1+z\varepsilon} \geq 2-\frac{1}{m_1}-\frac{2z\varepsilon}{m_1+z\varepsilon}$. The value of ε was selected such that the PoA is more than $2+\frac{1}{m_1}-\rho$. ■

Finally, we bound the PoA assuming the initial schedule is not be a NE. The upper bound follows from Theorem 2.12. The lower bound follows from Theorem 3.6.

Theorem 3.9 *If we ignore the demand for NE in the initial schedule, the PoA that is reached after performing improvement steps is at most m_1 and this is tight.*

4 Analysis of Coordinated Deviations

In this section we assume that agents can coordinate their strategies and preform a coordinated deviation. Recall that a set of players $\Gamma \subseteq N$ forms a *coalition*, if there exists a move where each job $j \in \Gamma$ strictly reduces its cost. A schedule s is a *strong equilibrium* (SE) if there is no coalition $\Gamma \subseteq N$ that has a beneficial move from s.

We show that a SE always exists and we bound the strong price of stability and the strong price of anarchy in modification scenarios involving an addition or removal of machines with uniform extension penalty. We also show that for any value of $\delta > 0$ it is NP-hard to determine whether a given schedule is a SE.

4.1 Equilibrium Existence and Decision Complexity

We first show the existence of a SE in our model.

Theorem 4.1 *Every instance of the load rebalancing game with added or removed machines admits at least one strong equilibrium.*

Proof: The proof is identical to the proof for the classic load balancing game [2]. For a given schedule s, let $(c_1, \ldots, c_n)$ be the sorted cost vector corresponding to s. That is, c_j is the cost of a job that has the j-th highest cost. If s is not a SE, there is a coalition Γ of size $k \leq n$ that can deviate such that each member of the coalition strictly decreases its cost. It can be seen that the sorted cost vector obtained after performing a beneficial move is lexicographically smaller. This implies that a SE is reached after a finite number of beneficial moves. ∎

Next, we prove that it is *NP-hard* to determine whether a NE schedule s is a SE. Moreover, given a set of jobs, it is NP-hard to determine whether this set has a beneficial coordinated deviation.

Theorem 4.2 *Let s be a NE schedule in a system after a modification took place. For any $\delta \geq 0$, it is NP-hard to determine whether s is a SE.*

Proof: We give a reduction from *Partition*. Given a set A of n integers $a_1, ..., a_n$ with total size $2B$, the goal is to determine whether A can be partitioned into two sets A_1, A_2 each having

total size B. We assume w.l.o.g. that $min_i a_i \geq max\{3, \delta\}$, otherwise, the whole instance can be scaled. Given A, construct an initial schedule on a single machine with $n+5$ jobs. One job has length $2B-1$, two jobs have length $2B-2-\delta$, two jobs have length $2B-1-\delta$ and n jobs have lengths $a_1-\delta, ..., a_n-\delta$. Clearly, as $m_0 = 1$, this schedule is a SE. Assume that $m' = 3$ machines are added. Figure 9(a) presents a possible modified schedule. It is easy to verify that this schedule is a NE.

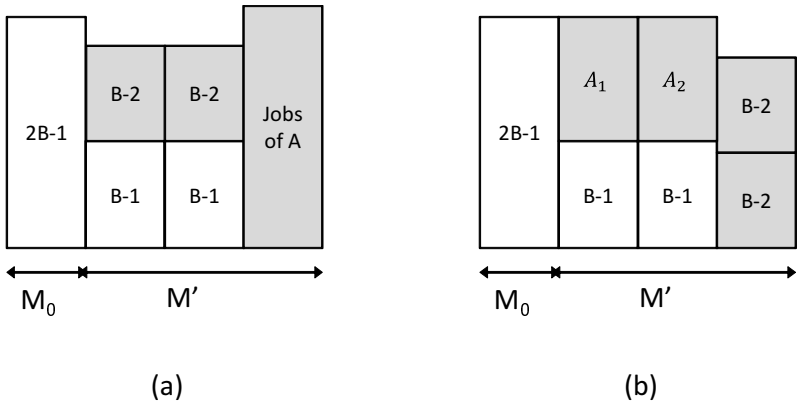

Figure 9: Partition induces a coalition. (a) a SE schedule if no partition exists (b) the schedule after a coalitional move - if a partition exist.

Claim 4.3 *The schedule in Figure 9(a) is an SE if and only if there is no partition.*

Proof: If A has a partition into A_1, A_2, each having total size B, then there is a coalition consisting of all the a_i-jobs and the two jobs of length $B-2-\delta$. Figure 9(b) shows the possible coalitional move. All the partition jobs reduce their cost from $2B$ to $2B-1$, and the $(B-2)$-jobs reduce their cost from $2B-3$ to $2B-4$.

Next, we show that if there is no partition then the schedule in Figure 9(a) is an SE. The job of length $2B-1$ does not participate in any coalition because it currently reaches its minimal cost. Therefore, any possible coalition involves only jobs assigned to machines in M'. Since all the jobs that can participate are extended and no job would return to its original machine, the scenario is identical to the classic load balancing game. Denote by $M'(i)$ the i-th machine in M'. As shown in [10], in any action of a coalition on three machines, jobs must migrate to $M'(3)$ (the

machine with the maximal load) from both $M'(1)$ and $M'(2)$. In order to decrease the load from $2B-3$, the set of jobs migrating to $M'(3)$ must be the set of two jobs of load $B-2$. Also, it must be that all the partition jobs move away from $M'(3)$ - otherwise, the total load on $M'(3)$ will be at least $2B-4+3 = 2B-1$, which is not an improvement for the $(B-2)$-jobs. This implies that the jobs of $M'(3)$ split between $M'(1)$ and $M'(2)$. However, since there is no partition, one of the two subsets is of total load at least $B+1$. These jobs will join a job of load $B-1$ to get a total load of at least $2B$, which is not an improvement over the $2B$ load in the schedule in Figure 9(a). Thus, no coalition exists. ■

This establishes the proof of the Theorem. ■

4.2 Equilibrium Inefficiency

In this section we present tight bounds for the *strong price of anarchy*. We note that by Theorem 2.6, the *strong price of stability* of the selfish load rebalancing game with any job extension penalty is 1.

The following observation will be used in the analysis of the strong price of anarchy (SPoA).

Observation 4.4 *In any SE, at least one job has cost at most OPT.*

Proof: If all jobs have cost more than OPT, then the schedule is not a SE because all jobs form together a coalition that prefers the optimal schedule. ■

Theorem 4.5 *The SPoA in load rebalancing games with uniform penalty with added or removed machines is at most* 3.

Proof: Denote the initial schedule by s_0 and the SE schedule by s. Let $L_{max}(s_0)$, $L_{max}(s)$ be the makespan of schedule s_0 and s respectively. If $\delta > OPT$, we distinguish between two cases.

1. Adding machines: If $\delta > OPT$, then in the optimal solution no job migrates. Thus, $L_{max}(s_0) = OPT$. We show that $s = s_0$, which clearly implies that s is optimal. Assume that $s \neq s_0$. Each of the jobs for which $s(j) \neq s_0(j)$ has cost larger than δ in s. Since $\delta > OPT = L_{max}(s_0)$, all these jobs form a coalition for which returning to s_0 is a beneficial move, contradicting the assumption that s is SE.

2. Removing machines: If no job was assigned on any of the m' removed machines in s_0, then the analysis of "adding machines" is valid. Otherwise, at least one job must migrate in the optimal solution. This job's cost is at least $p_j + \delta$. Contradicting the fact that $OPT < \delta$.

Otherwise, $\delta \leq OPT$. We distinguish between two cases:

1. $L_{max}(s)$ is determined by a single job j. First, is $L_{max}(s) = p_j$ for some job j, then the schedule is optimal since $OPT \geq \max_k p_k = p_j$. Second, if $L_{max}(s) = p_j + \delta$ then since $OPT \geq \max_k p_k = p_j$, we have $L_{max}(s) \leq OPT + \delta \leq 2OPT < 3OPT$.

2. $L_{max}(s)$ is determined by two or more jobs. Let j be the job with the smallest processing time on a machine with load $L_{max}(s)$. Let $L_{min}(s)$ be the machine with the minimal load. It holds that $L_{max}(s) \leq L_{min}(s) + p_j + \delta$, since the schedule is a NE. Since $p_j \leq OPT$, $\delta \leq OPT$ and $L_{min}(s) \leq OPT$ by observation 4.4, we have $L_{max}(s) \leq L_{min}(s) + p_j + \delta \leq OPT + OPT + OPT = 3OPT$.

Therefore, $SPoA \leq 3$. ∎

We show that the above analysis is tight even when the initial schedule is a SE in the cases of adding and removing machines. The example uses specific values of m_0 and m'. It can be generalized for additional values of m_0, m' by scaling and/or adding dummy jobs.

Theorem 4.6 *For any $\rho > 0$, there exists an instance with added machines for which $SPoA \geq 3 - \rho$.*

Proof: Given ρ, let $\varepsilon < 1$ be a small constant and let B be an integer such that $\rho \geq \frac{4\varepsilon}{B+\varepsilon}$. Fix $\delta = B - \varepsilon$. The initial schedule, on $m_0 = 3$ machines, is given in Figure 10(a). Note that each machine accommodates one long job of length B and one tiny job of length ε (the job lengthes' indices in the figure denote $s_0(j)$ - to help us follow the migrations). Since the load is perfectly balanced, s_0 is a strong equilibrium. Assume that $m' = 2$ machines are added. Consider the schedule s given in Figure 10(b). We have $L_{max}(s) = 2B + \delta = 3B - \varepsilon$.

Claim 4.7 *The schedule s is an SE.*

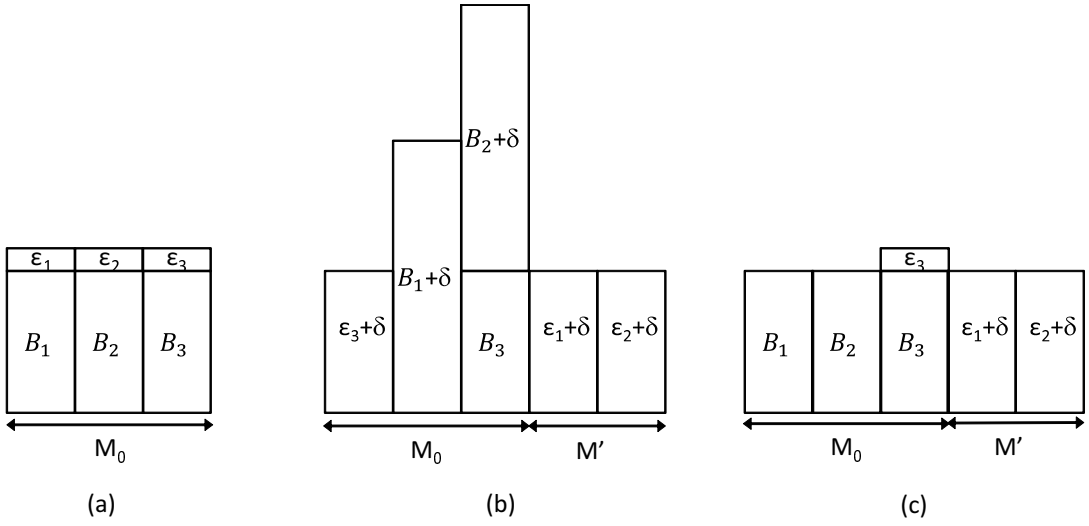

Figure 10: An instance achieving $SPOA = 3$. (a) the initial assignment, (b) a possible SE, and (c) the best SE.

Proof: We show that no job can be part of a coalition. Note that the current cost of each ε-job is $\varepsilon + \delta = B$, therefore, no job will join an ε-job on its current machine. Moreover, an ε-job will participate in a coalition only if it returns to its original machine alone. Therefore, after any coalitional move, three different machines are dedicated to the ε-jobs. Since there are three B_i jobs and two machines without ε-jobs, there is a machine with two B_i jobs on it. At least one of which is extended. Thus, in any coalitional move, one machine has load $2B + \delta$ which is not beneficial for the jobs assigned to it. Thus, no coalition exists. ∎

An optimal schedule for the modified instance is given in Figure 10(c). $L_{max}(OPT) = B + \varepsilon$. Therefore, $SPoA \geq \frac{3B-\varepsilon}{B+\varepsilon} = \frac{3B+3\varepsilon}{B+\varepsilon} - \frac{4\varepsilon}{B+\varepsilon} \geq 3 - \rho$. ∎

Theorem 4.8 *For any $\rho > 0$, there exists an instance with removed machines such that $SPoA \geq 3 - \rho$.*

Proof: Given ρ, let $\varepsilon < 1$ be a small constant and let B be an integer such that $\rho \geq \frac{4}{B+\varepsilon}$. Fix $\delta = B - \varepsilon$. The initial schedule, on $m_1 + m' = 5$ machines, in given in Figure 11(a). Note that the first three machines accommodate one long job of length B and the other two machines accommodate one small job of length ε. Since there is a single job on each machine, s_0 is a strong

equilibrium. Assume that $m' = 1$ machines are removed. Consider the schedule s given in Figure 11(b). We have $L_{max}(s) = 3B - \varepsilon$.

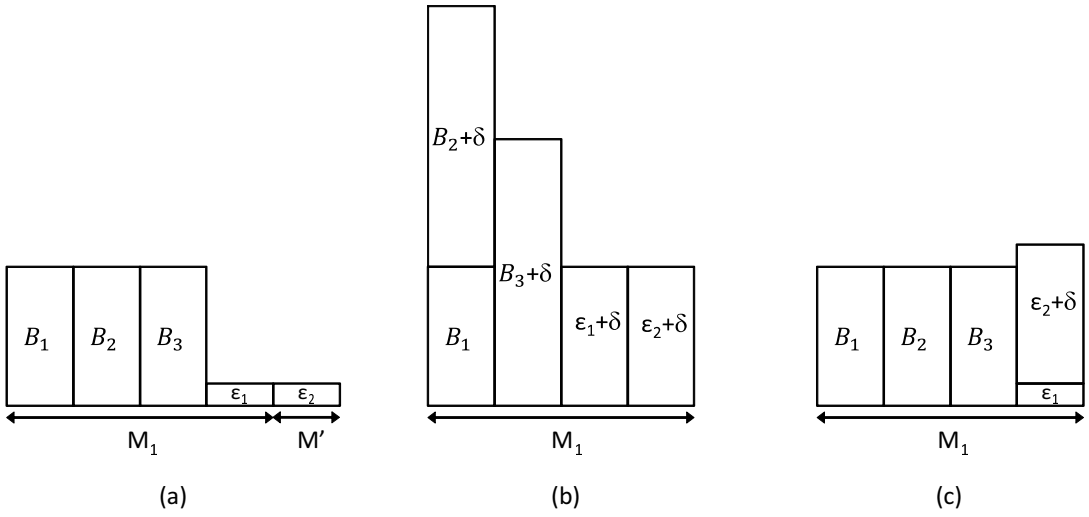

Figure 11: An instance achieving $SPOA = 3$. (a) the initial assignment, (b) a possible SE, and (c) the best SE.

Claim 4.9 *The schedule s is an SE.*

Proof: Clearly, ε_2 does not participate in any coalition because it reaches its minimal cost on $s_0(\varepsilon_1)$. If ε_1 participate in a coalition then it would only move back to its original machine, but it is occupied by ε_2, thus ε_1 does not participate in any coalition. The remaining three jobs have the length of B and can be scheduled on two machines since the other machines reach the load of B. In any schedule there would be at least two jobs of length B on a single machine. No job except B_1 and B_2 will participate in such a coalition. Thus, no coalition exists. ∎

An optimal schedule for the modified instance is given in Figure 11(c). $L_{max}(OPT) = B + \varepsilon$. Therefore, $SPoA \geq \frac{3B-\varepsilon}{B+\varepsilon} = \frac{3B+3\varepsilon}{B+\varepsilon} - \frac{4\varepsilon}{B+\varepsilon} \geq 3 - \rho$.

∎

It is possible to provide a tighter analysis of the strong price of anarchy, by bounding this value as a function of the ratio between δ and OPT. The proof of the following theorem is similar to the proof of Theorem 4.5.

Theorem 4.10 *The SPoA in load rebalancing games with uniform penalty is at most* $2 + \frac{\delta}{OPT}$.

The examples in the proofs of Theorem 4.6 and 4.8 show that this bound is tight. Moreover, as we show below, the above $SPoA$ bound is tight even if the initial schedule is a SE, and the final SE is reached by a sequence of coalitional improvement steps.

Theorem 4.11 *For every* $\rho > 0$*, there exists an instance with added machines such that* $SPoA \geq 2 + \frac{\delta}{OPT} - \rho$.

Proof: We show an instance with $m_0 = 4$ initial machines and $m' = 8$ added machines. By scaling and adding dummy jobs, this example can be generalized to other values of m_0, m'. Given ρ, let B be an integer such that $\rho \geq \frac{2}{B+1}$. Select δ, δ', ε such that $\delta = \delta' - \varepsilon$, $z = \frac{B}{\delta'}$ is an even integer at least 6 and $\varepsilon = \frac{1}{2(z+1)}$. For example, given $\rho = 0.1$, it is possible to select $B = 20$, $\delta' = 2, z = 10$ and $\varepsilon = \frac{1}{22}$. The initial schedule is given in Figure 12(a). Note that $1/\varepsilon$ jobs of length ε are assigned on the fourth machine. The first machine has load $3B - 2\delta + 1 + 2\varepsilon$, the other three machines have load $3B - 2\delta + 1$.

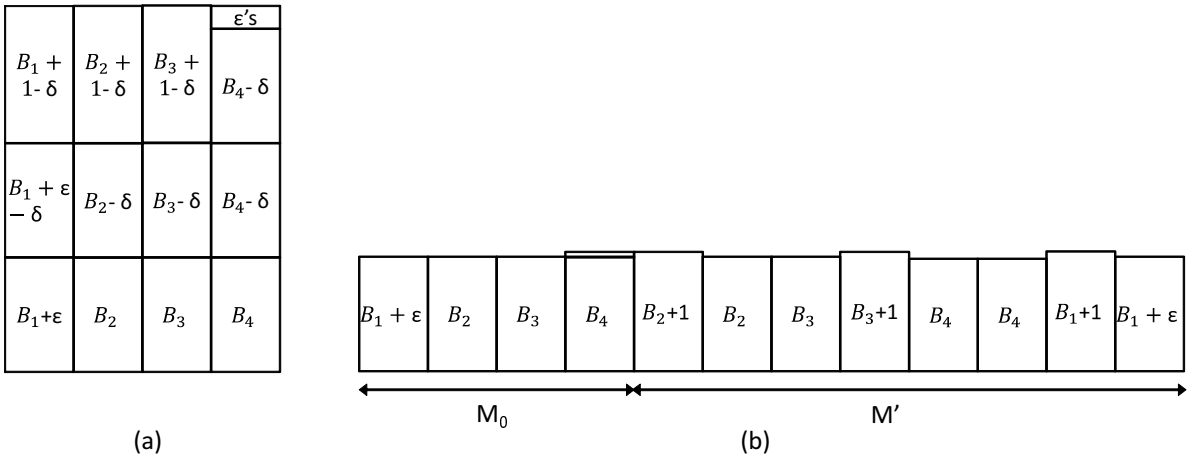

Figure 12: An instance achieving $SPOA = 2 + \frac{\delta}{OPT}$. (a) the initial assignment, (b) the best SE.

Since $L_{max}(s_0) = L_{min}(s_0) + 2\varepsilon$ and the shortest job on $L_{max}(s_0)$ is $B + \varepsilon - \delta$, s_0 is clearly a NE. Moreover, since all the other machines are balances, it is also a SE. Assume that $m' = 8$ machines are added. Figure 12(b) presents an optimal schedule after the modification. $L_{max}(OPT) = B + \varepsilon$. Figure 14 presents a possible sequence of coalitional improvement moves.

First, the ε-jobs migrate to two new machines. After this move, two M' machines have $z+1$ jobs of length $\varepsilon+\delta$ each, forming the load of $(z+1)(\varepsilon+\delta) = (z+1)\delta' = (\frac{B}{\delta'}+1)\delta' = B+\delta'$. Two jobs migrate from machines $M_0(2)$, $M_0(3)$, $M_0(4)$, each to an empty machine in M'. The schedule after the first move is shown in Figure 14(a).

Next, the $B_1+\varepsilon$ job performs an improvement step and moves from $M_0(1)$ to $M_0(2)$, forming the schedule in Figure 14(b). Then, the B_2 job performs an improvement step and moves from $M_0(2)$ to $M_0(3)$, forming the schedule in Figure 14(c). The next move is of the coalition consisting of the two B_1 jobs that are currently scheduled on $M_0(1)$ and z jobs of length ε that are scheduled on the machines in M' ($\frac{z}{2}$ jobs of each machine in M'). The z jobs of length ε migrate to $M_0(1)$ and form the load of $z(\varepsilon+\delta) = z\delta' = B$. The jobs benefit from the move since they previously paid $B+\delta'$. The $B_1+\varepsilon-\delta$ and $B_1+1-\delta$ jobs, each migrate to the a different machine in M' that the ε-jobs left. Since $\frac{z}{2}$ is at least 3, the resulting load on these machines is at most $B+\delta'-3(\varepsilon+\delta) = B-2\delta'$. The $B_1+\varepsilon-\delta$ job benefits from the move since it currently pays at most $B-2\delta'+B+\varepsilon = 2B+\varepsilon-2\delta-2\varepsilon = 2B-\varepsilon-2\delta$ instead of $2B+1+\varepsilon-2\delta$. The $B_1+1-\delta$ job benefits from the move since it currently pays at most $B-2\delta'+B+1 = 2B+1-2\delta-2\varepsilon = 2B+1-2\varepsilon-2\delta$ instead of $2B+1+\varepsilon-2\delta$. The schedule after the move is shown in Figure 14(d).

The final move is of ε-jobs that remained on M'. They prefer to return to their original machine, $M_0(4)$, and pay $B+\frac{z+2}{2(z+1)}$. The final schedule, s', is shown in Figure 14(e).

Claim 4.12 *The schedule s' is an SE.*

Proof: Consider the B_4 job that is scheduled on $M_0(4)$, it does not want to participate in any coalition since it reached its minimal cost on $M_0(4)$ even with the other ε-jobs. Therefore, the ε-jobs that are scheduled on $M_0(1)$ cannot return to $M_0(4)$. If there is a coalition that does not includes the ε-jobs from $M_0(1)$, then there is a load of B on at least two machines. Since there are 11 jobs of length B that may participate in the coalition and 9 other machines, after the move there would be at least one machine with two B_i jobs on it. The only jobs willing to participate in such a coalition are B_2 and B_3 that are scheduled on $M_0(3)$. Thus, no coalition exists.

If there is a coalition that includes the ε-jobs from $M_0(1)$, then they occupy two machines by themselves (without any B_i job). Since there are 11 jobs of length B that may participate in

the coalition and 8 remaining machines, after the move there would be at least one machine with two B_i jobs on it. The only jobs willing to participate in such a coalition are B_2 and B_3 that are scheduled on $m_0(3)$. Thus, no coalition exists. ∎

An optimal schedule for the modified instance is given in Figure 12(b). $L_{max}(OPT) = B + 1$. Therefore,

$$SPoA \geq \frac{2B+\delta}{OPT} = \frac{2B+2}{OPT} + \frac{\delta}{OPT} - \frac{2}{OPT} = \frac{2B+2}{B+1} + \frac{\delta}{OPT} - \frac{2}{B+1} = 2 + \frac{\delta}{OPT} - \rho.$$

∎

This example can be generalized for any $m_0 \geq 4$ and $m_0|m'$.

Next, we show that the bound of $2 + \frac{\delta}{OPT}$ is tight also in the case of machines' removal where the initial schedule is a SE.

Theorem 4.13 *For any $\rho > 0$, there exists an instance with removed machines for which $SPoA \geq 2 + \frac{\delta}{OPT} - \rho$.*

Proof: Let m_0 be an odd integer at least 5. Let $m_1 = \frac{m_0+3}{2}$. Given ρ, let B be an integer such that $\rho \geq \frac{2}{B+1}$. Let $\delta'|B$ and $\delta = \delta' - \varepsilon$ where $\varepsilon = \frac{\delta'}{B}$. The initial schedule, on $m_0 = m_1 + m'$ machines is given in Figure 13(a). Note that there is a single job on each machine except for the fourth machine that is assigned $\frac{1}{\varepsilon}$ jobs of length ε. Each of the machines $5, \ldots, m_1$ is assigned a single job of length 1. And each machine $m_1, \ldots, m_0$, (the m' rightmost machines) is assigned a single job of length $B - \delta$. Note that the fourth machine is the only machine with more than a single job, and it has the minimum load, thus s_0 is a SE. Assume that the rightmost m' machines are removed. Consider the schedule s' given in Figure 13(b). In s', each of the jobs from M' migrates to a different machine from $4, \ldots, m_1$. The ε-jobs are scheduled on the third machines and B_2, B_3 migrate to machines $1, 2$ respectively. The maximal load in the resulting schedule is on the first machine. We have $L_{max}(s') = 2B + \delta$.

Claim 4.14 *The schedule s' is an SE.*

Proof: Assume that s' is not a SE, therefore a coalition exists. Clearly, B'_1 will not participate in any coalition because it reaches its minimal cost when scheduled alone on the fourth machine.

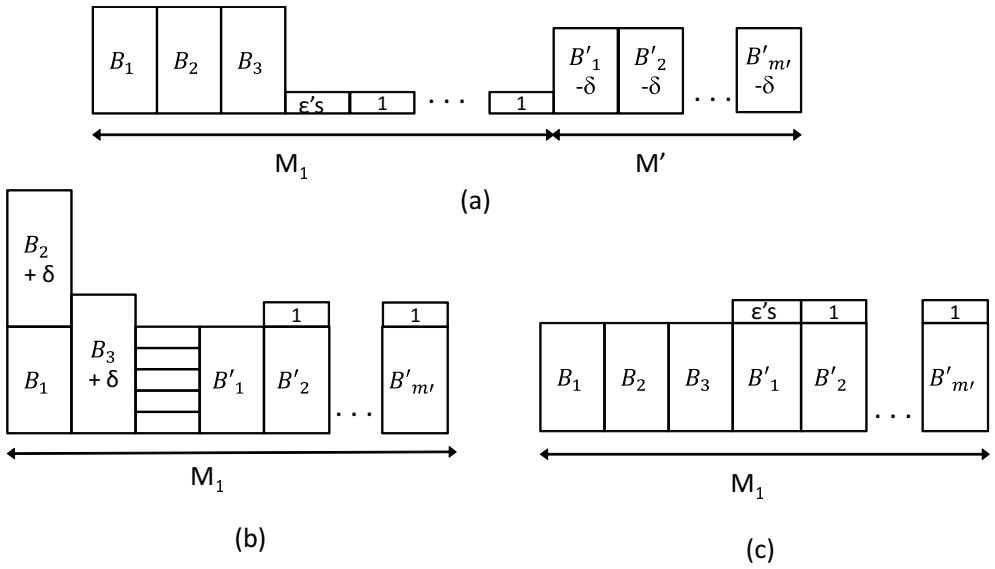

Figure 13: An instance achieving $SPOA = 2B + \delta$. (a) the initial assignment, (b) a possible SE, and (c) the best SE.

This implies that the ε-jobs will not migrate back to machine 4. If the ε-jobs participate in a coalition, then after the move they split among at least two machines without any B-job. In such a schedule the remaining $m_1 - 1$ jobs of length at least B must be assigned on $m_1 - 2$ machines. Therefore, at least one of the machines is assigned two jobs of length at least B. No jobs except for B_1 and B_2 is ready to participate in such a coalition. Clearly, B_1, B_2 alone cannot initiate such a deviation. ■

An optimal schedule for the modified instance is given in Figure 13(c). $L_{max}(OPT) = B + 1$. Therefore,

$$SPoA \geq \frac{2B+\delta}{OPT} = \frac{2B+2}{OPT} + \frac{\delta}{OPT} - \frac{2}{OPT} = \frac{2B+2}{B+1} + \frac{\delta}{OPT} - \frac{2}{B+1} = 2 + \frac{\delta}{OPT} - \rho.$$

■

5 Summary and Future Work

In this work, we consider a dynamic variant of the classic load balancing game, in which selfish jobs need to be assigned on a set of identical parallel machines, and each job's cost is the load on the machine it is assigned to. Given an initial assignment, the system is modified; specifically, some machines are added or removed. When machines are added, jobs naturally have an incentive to migrate to the new unloaded machines. When machines are removed, the jobs assigned to them must be reassigned. As a result of these migrations, other jobs might also benefit from migrations. In this model, we consider *an extension parameter* $\delta \geq 0$ that is associated with migrating from the initial schedule. In the modified schedule, if the machine on which a job is scheduled is different from its initial machine, then the job's processing time is extended by δ.

To the best of our knowledge, these are the first results considering games with migration costs. We provided answers to the basic questions arising in this model. Specifically, the existence of Nash equilibrium and strong equilibrium, the calculation of a Nash equilibrium and the lower and upper bounds of PoS/PoA/SPoS/SPoA. Many important problems remain open. We list below some possible directions for future work.

1. The first and most natural generalization of our work would be to consider heterogenous systems, in particular unrelated machines. For the classic load balancing problem, there is a significant difference between the games induced by identical and related machines. It would be interesting to check if similar differences exist also in our model.

2. Consider settings in which the extension penalty is not uniform. That is, for each i, i', j we are given an extension parameter $\delta_{i,i',j}$ such that job j is extended by $\delta_{i,i',j}$ if it migrates from machine i to machine i'. This work studied the case $\delta_{i,i',j} = \delta$ for all i, i', j.

3. Study proportional extension, i.e., a migration of job j extends its processing time from p_j to $p_j(1 + \delta)$.

4. Our analysis of the PoA is for arbitrary values of δ. Another direction is to analyze instances in which δ is bounded by the instance parameters, e.g., when $\delta \leq p_{min}$.

5. Our assumption is that migrating jobs are extended. Thus, a migration of j affects all

the jobs assigned to j's target machine. Another possible game can be defined by assuming *individual* penalties. Specifically, migrations are associated with cost, but this cost is covered by the job and does not affect other jobs. The cost of a job j assigned to machine i is L_i if $i = s_0(j)$ and $L_i + \delta$ otherwise, where the load is the total processing time of jobs assigned to machine i.

6. Consider jobs with uniform lengths, for which tighter bounds of the PoA/PoS might be found. For uniform jobs it might also be possible to analyze the convergence time of best response dynamics and to calculate the social optimum efficiently.

7. Finally, in this work we considered the social objective function of minimizing the maximal cost. It would also be interesting to study and analyze the total payments.

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 *Symposium on the Foundations of Computer Science (FOCS)*, pages 295–304, 2004.

[2] N. Andelman, M. Feldman, and Y. Mansour. Strong Price of Anarchy. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

[3] R. Aumann. Acceptable points in general cooperative n-person games. In *Contributions to the Theory of Games*, volume 4, 1959.

[4] Y. Azar, K. Jain, and V.S. Mirrokni. (Almost) optimal coordination mechanisms for unrelated machine scheduling. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2008.

[5] I. Caragiannis, M. Flammini, C. Kaklamanis, P. Kanellopoulos, L. Moscardelli. Tight Bounds for Selfish and Greedy Load Balancing. *Algorithmica*, 61(3): 606–637, 2011.

[6] G. Christodoulou, E. Koutsoupias, and A. Nanavati. Coordination mechanisms. *J. Daz, J. Karhumaki, A. Lepisto, and D. Sannella (Eds.), Automata, Languages and Pro- gramming, LNCS Vol. 3142: Springer*, (2):345–357, 2004.

[7] A. Czumaj. Selfish Routing on the Internet. In *Chapter 42 in Handbook of Scheduling: Algorithms, Models, and Performance Analysis, edited by J. Leung, CRC Press, Boca Raton, FL*, 2004

[8] A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. In *ACM Transactions on Algorithms*, vol.3(1), 2007

[9] E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence time to Nash equilibria. *In Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP), pages 502–513*, 2003.

[10] M. Feldman and T. Tamir. Approximate Strong Equilibrium in Job Scheduling Games. *Journal of Artificial Intelligence Research*, 2009.

[11] M. Feldman and T. Tamir. Conflicting congestion effects in resource allocation games. Journal of Operation Research. vol. 60(3), pages 529–540, 2012.

[12] A. Fiat, H. Kaplan, M. Levi, and S. Olonetsky. Strong Price of Anarchy for Machine Load Balancing. In *In Proceedings of the 34th International Colloquium on Automata, Languages and Programming (ICALP)*, 2007.

[13] G. Finn and E. Horowitz. A linear time approximation algorithm for multiprocessor scheduling.. In *BIT, Vol. 19, No. 3, pages 312–320*, 1979.

[14] D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, P. Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP), pages 123–134*, 2002.

[15] M. Gairing, T. Lücking, M. Mavronicolas, and B. Monien. Computing Nash equilibria for scheduling on restricted parallel links.. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC), pages 613–622*, 2004.

[16] M. R. Garey, David S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness.W. H. Freeman& Co. 1979.

[17] R.L. Graham. Bounds for Certain Multiprocessing Anomalies. *Bell Systems Technical Journal*, 45:1563–1581, 1966.

[18] R.L. Graham. Bounds on Multiprocessing Timing Anomalies. *SIAM J. Appl. Math.*, 17:263–269, 1969.

[19] D.S. Hochbaum and D.B. Shmoys. Using dual approximation algorithms for scheduling problems: Practical and theoretical results. *Journal of the ACM*, 34(1):144–162, 1987.

[20] N. Immorlica, L. Li, V. Mirrokni, and A. Schulz. Coordination Mechanisms for Selfish Scheduling, *Theoretical Computer Science*, vol. 410(17):1589–1598, 2009.

[21] E. Koutsoupias and C. Papadimitriou. Worst-case Equilibria. *Computer Science Review*,3(2): 65-69, 1999.

[22] C. Papadimitriou. Algorithms, Games, and the Internet. In *Proceedings of 33rd ACM Symposium on Theory of Computing (STOC)*, pages 749–753, 2001.

[23] 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.

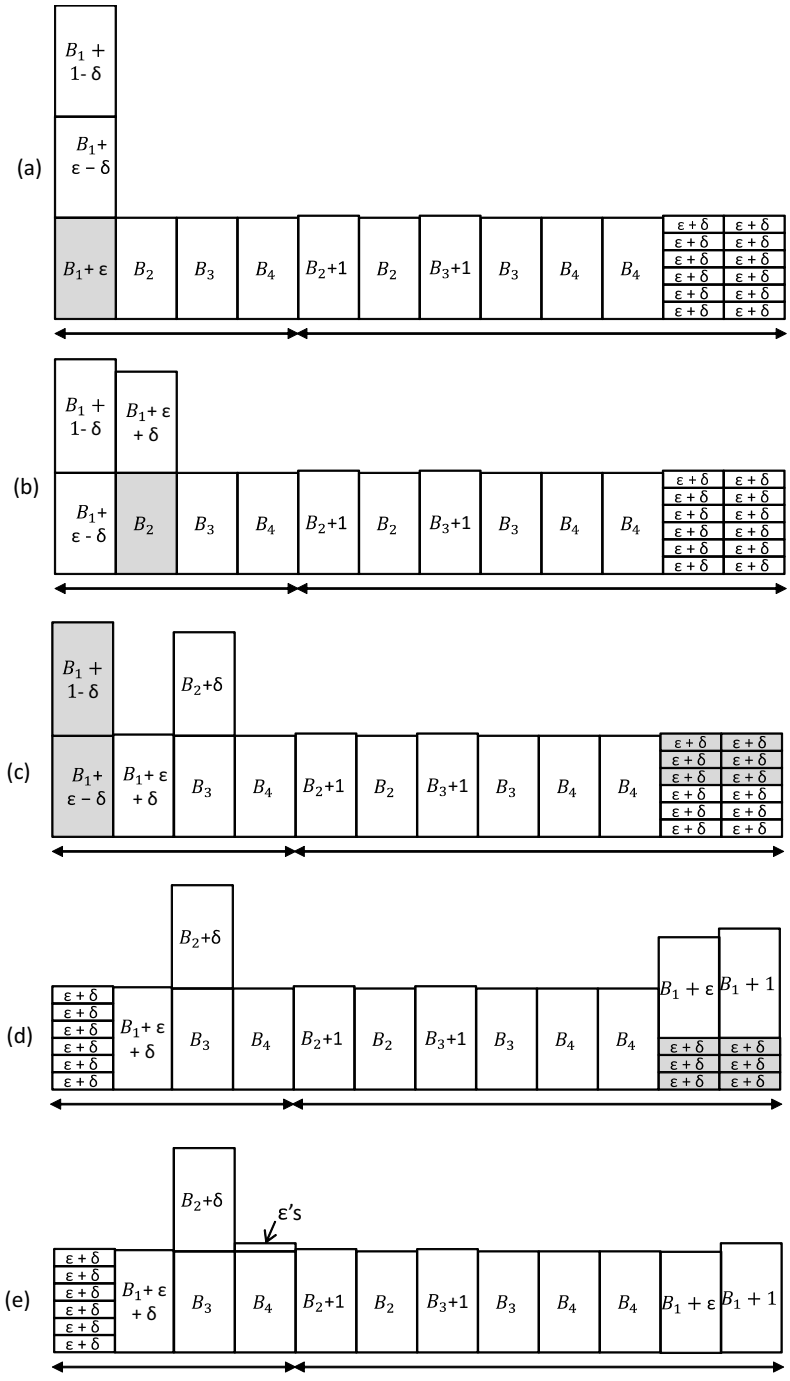

Figure 14: An instance achieving $SPOA = 2 + \frac{\delta}{OPT}$, where the SE is reached by a sequence of coalitional improvement steps. The jobs forming the coalitions are in grey.

- $2 - \frac{1}{m_0 - m'}$ עבור מעברי שיפור כאשר יש חשיבות לסדר הפעולות במקרה של הורדת מכונות.

בנוסף, ננתח את המשחק גם כאשר בכל שלב מותרת פעילות של קואליציה של שחקנים, כזו שכל חברי הקואליציה משפרים בה את ההשמה שלהם. נראה שקיימת השמה אופטימלית המהווה שיווי משקל חזק, ושהמחיר החזק של אנרכיה הוא 3.

תקציר

העבודה עוסקת במשחק המתקבל בבעיית שיבוץ של משימות על מכונות זהות. נתונות n משימות באורכים שונים, כל משימה מונעת על ידי שחקן אנוכי. יש לעבד את המשימות על m מכונות זהות כאשר כל שחקן בוחר את המכונה עליה תושם המשימה שלו. בהנתן שיבוץ s_0 של עבודות למכונות, מתבצע שינוי כלשהו במערכת שבעקבותיו יתכן והשחקנים יצטרכו או יעדיפו לבחור השמה שונה עבור המשימה שלהם.

נבחן שני שינויים אפשריים במערכת: נתון שיבוץ התחלתי על m_0 מכונות. m' מכונות חדשות נוספות למערכת או m' מכונות יורדות מהמערכת. כשמתווספות מכונות ניתן להעביר אליהן משימות שהיו משובצות על המכונות המקוריות. כשיורדות מכונות, המשימות שהיו משובצות עליהן חייבות לעבור למכונות שנותרו. בשני המיקרים יתכנו גם תזוזות פנימיות בקרב המשימות על המכונות המקוריות.

במידה וסוכן בוחר עבור המשימה שלו השמה השונה מההשמה המקורית s_0 אורך המשימה גדל בפרמטר הארכה נתון δ. כלומר, אורך משימה j הוא p_j במידה והוא נשאר על המכונה המקורית שבה הוצב בs_0 או $p_j + \delta$ במידה ועבר ממנה. התשלום של סוכן שבבעלותו משימה j המושמת על מכונה i, הוא סכום ארכי המשימות על מכונה i כאשר אורכים אלו כוללים את ההארכות שהתווספו בשל מעברים.

פונקציית הרווחה החברתית, על פיה נעריך את איכותה של השמה, היא העלות המקסימלית של איזשהו סוכן. פונקציה זו שקולה לפונקציית המטרה של הבאה למינימום את העומס של המכונה עם העומס המקסימלי.

בעבודה זו נתייחס למספר בעיות העולות במשחק זה. כל סוכן מעוניין להביא למינימום את התשלום עבור המשימה שלו על-ידי שיפור מיקומה. סוכן יכול להעביר משימה למכונה אחרת אם מעבר כזה מקטין את התשלום שלו. במצב של שיווי משקל נאש, אין אף משימה שניתן להפחית את התשלום שלה על ידי מעבר למכונה אחרת.

נציג תוצאות המתייחסות לחישוב שיווי משקל נאש טהור (השמה יציבה) ולבעיה של השגת השמה יציבה בעלת מחיר מינימאלי. נראה שכשקיימים קנסות על המעברים קיימת התכנסות לשיווי משקל נאש וקיים סידור של השחקנים שאם פועלים על פיו, כלומר, כל שחקן בתורו מבצע את הצעד הטוב ביותר עבורו, התכנסות זאת נעשית בזמן לינארי. נראה שקיימת השמה יציבה אופטימלית ונציג חסמים עליונים ותחתונים למחיר האנרכיה (PoA).

מעבר שיפור הוא מעבר שבו הסוכן מוריד את המחיר שהוא משלם ע"י שיפור מקומה. נראה שמחיר האנרכיה אינו מוגבל ותלוי בערך של הקנס על מעבר. לעומת זאת, כאשר אנחנו דנים בערך מחיר האנרכיה המתקבל מביצוע מעברי שיפור של הסוכנים, נראה שהערכים הם:

- $2 + \frac{m'-1}{m_0}$ עבור מעברי שיפור במקרה של הוספת מכונות.
- $m_0 - m'$ עבור מעברי שיפור במקרה של הורדת מכונות.

IDC HERZLIYA
Efi Arazi School of Computer Science

המרכז הבינתחומי בהרצליה

בית-ספר אפי ארזי למדעי המחשב

משחקי איזון עומס במערכות דינאמיות

סיכום עבודת גמר המוגשת כמילוי חלק מהדרישות לקראת
תואר מוסמך במסלול מחקרי במדעי המחשב

על-ידי סופיה בליקובצקי

העבודה בוצעה בהנחיית פרופ׳ תמי תמיר

ינואר 2013