Load Rebalancing Games in Dynamic Systems with Migration Costs*

Sofia Belikovetsky † Tami Tamir ‡

Abstract

We consider the following dynamic load balancing game: Given an initial assignment of jobs to identical parallel machines, the system is modified; specifically, some machines are added or removed. Each job's cost is the load on the machine it is assigned to; thus, when machines are added, jobs have an incentive to migrate to the new unloaded machines. When machines are removed, the jobs assigned to them must be reassigned. Consequently, other jobs might also benefit from migrations. In our *job-extension penalty* model, for a given *extension parameter* $\delta \geq 0$, if the machine on which a job is assigned to in the modified schedule 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. Our results show that the existence of job-migration penalties might lead to poor stable schedules; however, if the modification is a result of a sequence of improvement steps or, better, if the sequence of improvement steps can be supervised in some way (by forcing the jobs to play in a specific order) then any stable modified schedule approximates well an optimal one.

Our work adds two realistic considerations to the study of job scheduling games: 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.

Keywords: algorithms, load balancing games, dynamic systems, Nash equilibrium.

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.

*A preliminary version of this paper appears in the proceedings of the 6th International Symposium on Algorithmic Game Theory (SAGT'13).

†School of Computer Science, The Interdisciplinary Center (IDC), Herzliya, Israel. Email: sofia.belikovetsky@gmail.com

‡School of Computer Science, The Interdisciplinary Center (IDC), Herzliya, Israel. Email: tami@idc.ac.il

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 has been widely studied in recent years from a game theoretic perspective, see [14, 2, 5, 8], and a survey in [17].

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 since 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.

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.
4. Improvement steps are done unilaterally or by coalitions.

Applications: Traditional analysis of job scheduling 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 [15].

Our work adds two realistic considerations 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 in 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 begins (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.

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.

Note that in all the above applications, the delay caused due to a migration is independent of the migrating job. A similar, low-tech, application is freight transport, in which the whole cargo ship is delayed when items are added. The registration and handling of a new item takes fixed time, independent of the item's size and weight.

1.1 Model and Preliminaries

A job rescheduling setting is defined by the tuple $G = \langle M_0, M', N, p_j, \delta \rangle$, where M_0 is a set of 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 to machines, where $s(j)$ is the machine to which job j is assigned. The assignment is referred to as a schedule. We use s_0, s to denote the initial and the modified schedules, respectively. 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 $c_j(s) = L_{s(j)}$.

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 $i \neq s(j)$, $L_i + p_j + \delta_{i,j} \geq L_{s(j)}$.

Some of our results refer to outcomes of a sequence of improvement steps. Better-Response Dynamic (RD) is a local-search method where in each step some player is chosen and is allowed to change his assignment, given the assignment of the others. In better-RD, any improvement step can be performed. In best-RD, players select their best possible response. When best-RD or better-RD are performed, one job might migrate several times. The extension penalty is independent of the number of steps and only the final assignment matters. In particular, if j leaves its machine in s_0 and returns to it later, then j is not extended. This is justified by the applications motivating our work - in which the penalty is not due to physical migration cost but due to the adjustment of the job's processing to a new machine.

It is well known that decentralized decision-making may lead to sub-optimal solutions from the point of view of society as a whole. We quantify the inefficiency incurred due to self-interested behavior according to the *price of anarchy* (PoA) [14, 15] and *price of stability* (PoS) [1] measures.

The PoA is the worst-case inefficiency of a Nash equilibrium, while the PoS measures the best-case inefficiency of a Nash equilibrium. The social objective function we consider is the egalitarian one, 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*). For a schedule s, $makespan(s) = max_j c_j(s) = max_i L_i(s) = L_{max}(s)$. Formally,

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. That is $PoA(G) = \max_{s \in \Phi(G)} L_{max}(s)/OPT(G)$. 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 equilibrium and the social optimum of G. That is, $PoS(G) = \min_{s \in \Phi(G)} L_{max}(s)/OPT(G)$, 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. The *strong price of anarchy* and the *strong price of stability* are defined similarly, where $\Phi(G)$ refers to the set of strong Nash equilibria.

1.2 Related Work

The minimum makespan problem corresponds to the centralized version of our game in which all jobs obey the decisions of one utility. This is a well-studied NP-hard problem. The simple greedy List-scheduling (LS) algorithm [11] provides a $(2 - \frac{1}{m})$-approximation to the minimum makespan problem. A bit better approximation ratio of $(\frac{4}{3} - \frac{1}{3m})$ is guaranteed by the Longest Processing Time (LPT) algorithm [12]. A PTAS for the minimum makespan problem on identical machines is given in [13].

In the associated load balancing 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. Fotakis et al. showed that LPT-schedules are NE schedules [10]. In [6], Even-dar et al. analyzed the convergence time of BRD on unrelated machines. Note that our model can be seen as a restricted case of scheduling on unrelated machines. For every job j and machine M_i, the processing time of j on M_i is p_j if $i = s_0(j)$ and $p_j + \delta$ otherwise. Our analysis provides tighter results than those known for unrelated machines [5].

The concept of the price of anarchy (PoA) was introduced by Koutsoupias and Papadimitriou in [14]. They proved that the price of anarchy of job scheduling games is $2 - \frac{1}{m}$. In [9], 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.

Other related work deal with cost functions that depend on the internal order of jobs, e.g., in [4, 3], or cost function that is based on both the congestion on the machine and its activation cost [7]. Coordinated deviations were studied by Andelman et al. in [2]. A survey of results on selfish load balancing appears in [17].

Some of our results bound the inefficiency of a NE produced by a sequence of improvement steps (best or better-RD). Analysis of such *sequential* NE was initiated in [16] for several other games.

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 added or removed machines possesses 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 a NE that is achieved by performing improvement steps, we show that the PoA is

1. $\frac{m'-1}{m_0} + 2$ when machines are added and s_0 is a NE.
2. $m_0 + m'$ when machines are added and s_0 is not a NE.
3. $m_0 - m'$ when machines are removed (and s_0 is either a NE or not a NE).
4. $2 - \frac{1}{m_0 - m'}$ when machines are removed, s_0 is a NE, and jobs are activated in a specific order, denoted *two-phase better-RD*.

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 both the 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 $2 + \frac{\delta}{OPT}$ and that this bound is tight. Moreover, it is achieved even when the SE is reached by a sequence of coalitional improvement steps. Finally, we show that for any value of $\delta > 0$, it is NP-hard to determine whether a given modified schedule is a SE.

This paper 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. Finally, in Section 5 we conclude and suggest several directions for future work.

We note that in a dynamic setting in which machines are added or removed and migrations are free of cost (i.e., when $\delta = 0$), then the results known for classic load balancing games apply. In particular, the PoA assuming $\delta = 0$ is $2 - \frac{2}{m+1}$ for a game with m machines in the modified systems. 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 system's 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.

2.1 Equilibrium Existence and Computation

Every instance of the load rebalancing game with added machines and uniform extension penalty admits at least one pure Nash equilibrium. This follows from the fact that the sorted machines' load-vector corresponding to the schedule is decreasing lexicographically with any beneficial move. Thus, any better-RD process converges 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, $p_1 \geq p_2 \geq \ldots \geq p_n$. *Max-length best-RD* 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 phase of *max-length best-RD*, the system reaches a NE. While this result is valid also for the classic load balancing game [17], its proof for the load rebalancing game is more involved, since migrating jobs have stronger incentive to return to their initial machine (and get rid of the penalty).

Theorem 2.1 *Let s_0 be any initial schedule of n jobs on m_0 machines. Assume that m' machines are added. Starting from s_0, max-length best-RD reaches a pure Nash equilibrium after each job is activated once.*

Proof: 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, $\min\{L_a^0 - p_j, L_b^0 + p_j + \delta\} \geq \min\{L_b^0 + \delta, L_b^0 + p_j + \delta\} \geq \min\{L_a^0, L_b^0\}$. ∎

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 2.3 *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 2.4 *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,

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

This is true also in case $i = s_0(k)$, ($L^{t'}_b \leq L^{t'}_i + p_k \leq L^{t'}_i + p_j + \delta$). Thus, $L^{t'}_b - p_j - \delta \leq L^{t'}_i$ and condition C_1 does not hold. For M_a we show that condition C_2 does not hold, that is $L^{t'}_a \geq L^{t'}_b - p_j$. 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

$$L^{t'-1}_a > L^{t'-1}_b + p_k + \delta \tag{1}$$

After the move $L^{t'}_a = L^{t'-1}_a - p_k$ and $L^{t'}_b = L^{t'-1}_b + p_k + \delta$. Job j would benefit from migrating to M_a if $L^{t'}_a \leq L^{t'}_b - p_j$ which is equivalent to $L^{t'-1}_a - p_k + p_j < L^{t'-1}_b + p_k + \delta$. Since job j was activated before job k, we have $p_k \leq p_j$. Thus, $L^{t'-1}_a < L^{t'-1}_a - p_k + p_j$, and $L^{t'-1}_a < L^{t'-1}_b + p_k + \delta$, 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, $L^{t'}_a + p_k + \delta \geq L^{t'}_b$. This implies that $L^{t'}_a \geq L^{t'}_b - p_k - \delta$. Therefore, condition C_2 does not hold and job j will not benefit from migrating back to M_a. ∎

We conclude that max-length best-RD 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 $PoS = 1$ and $PoA = 2 - \frac{2}{m+1}$. 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 $\frac{m'-1}{m_0} + 2$ 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 $m_0 + m'$.

It is easy to see that a beneficial move does not increase the makespan. Therefore, by performing best-RD starting from any optimal assignment, we reach a NE whose makespan is equal to the optimum. This implies that the PoS equals 1. We turn to analyze the PoA. 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 2.5 *When the NE is not necessarily 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,

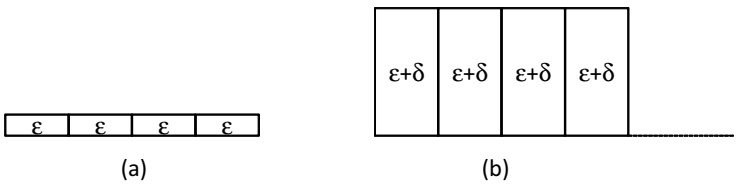

Figure 1: An instance achieving unbounded PoA. (a) the initial assignment, (b) the worst NE.

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 $\frac{\varepsilon+\delta}{\varepsilon} = r$. ∎

The more realistic scenario is when the NE is reached by performing beneficial moves, starting from a NE schedule s_0. We provide an upper bound that is tight when m' mod $m_0 = 1$, and almost tight for any other case.

Theorem 2.6 *The PoA when the NE is reached by better-RD is at most* $\frac{m'-1}{m_0} + 2$.

Proof: 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. 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

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

■

Let $m' = km_0 + \alpha$ for integers k and $\alpha < m_0$. By Theorem 2.6, we have that the PoA is at most $\frac{m'-1}{m_0} + 2 = \frac{m_0 k + \alpha - 1}{m_0} + 2 = k + \frac{\alpha - 1}{m_0} + 2$. We show that for $\alpha = 1$ and any k, the bound is tight. Almost tight analysis for other values of k, α is given in Appendix A.

Theorem 2.7 *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 $PoA > 2 + \frac{m'-1}{m_0} - \rho$, and the NE is reached by better-RD.*

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 present the construction of the lower bound in Fig. 2, where $m_0 = 3$ and $k = 1$ (implying $m' = 4$). The initial assignment is given in Fig. 2(a).

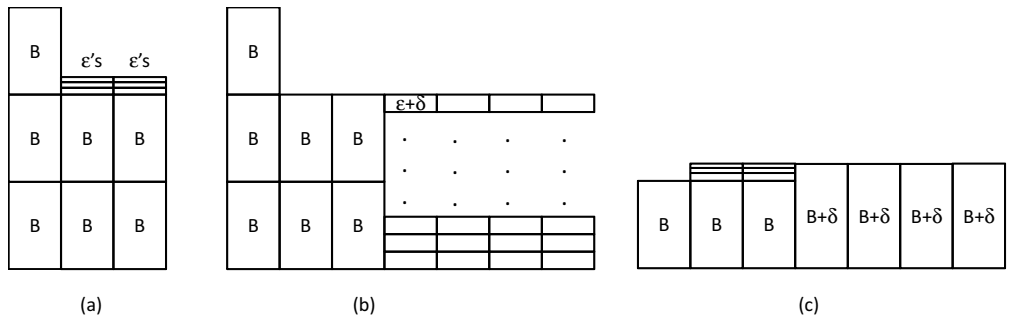

Figure 2: An instance achieving the maximal possible PoA. (a) the initial assignment, (b) the worst NE, and (c) the best NE.

Assume that m' machines are added and improving steps are performed. A possible NE (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 $(k+1)B(\delta + \varepsilon) =$

$(k+1)B$. 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 $2+k-\rho = 2+\frac{m'-1}{m_0}-\rho$. ∎

Finally, we analyze the PoA for arbitrary initial assignment. The analysis is tight for any number of machines m_0, m' and for any $\delta > 0$.

Theorem 2.8 *If the initial assignment is not necessary a NE, and the modified schedule is reached by better-RD, 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 performed, 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, $PoA \leq \frac{L_{max}}{OPT} \leq \frac{L^0_{max}}{OPT} \leq \frac{\sum_j p_j}{\sum_j p_j/(m_0+m')} \leq m_0+m'$. ∎

We show that this bound is tight for any number of machines m_0, m' and for any $\rho > 0$:

Theorem 2.9 *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 performed, 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 present the construction of the lower bound in Figure 3. The initial assignment is given in Figure 3(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 3(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 3(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 $B+\delta < B+1$.

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 $m_0+m'-\rho$. ∎

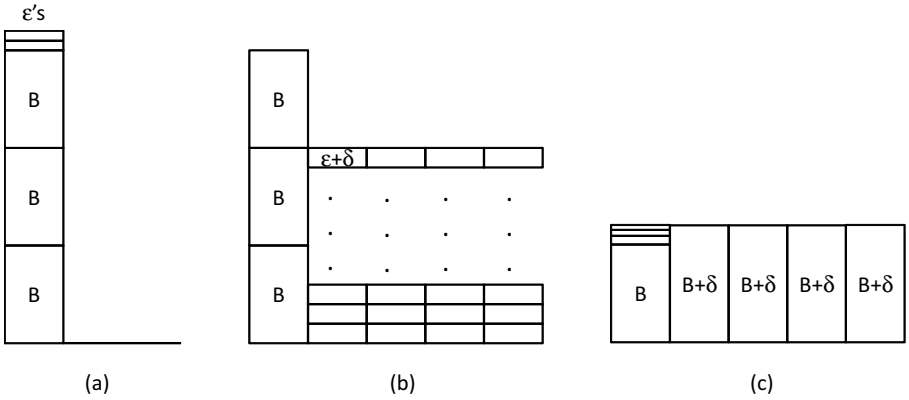

Figure 3: An instance achieving the maximal possible PoA. (a) the initial assignment (Not a NE), (b) the worst NE, and (c) the best NE.

3 Machines' Removal

In this section we study the scenario in which the systems' modification involves the removal of machines. Every job assigned to a removed machine must be reassigned. As a result, additional jobs might also be interested in migrating. 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 numbers of machines, that is $m_1 = m_0 - m'$. Throughout this section we assume that the initial schedule, s_0, is a NE. The last result in this section, Theorem 3.8, considers the case in which 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. When better-RD is applied, 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, move to a machine in M_1, and be extended. Clearly, jobs must not migrate into machines in M'.

Theorem 3.1 *Better-RD 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 $(L_1, \ldots, L_{m_1})$. ∎

Consider the following specific application of better-RD:

Two-phase max-length best-RD: 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. In both phases, each job performs its best move.

We first demonstrate that unlike the 'adding machines' scenario, when machines are removed, a single phase of max-length best-RD might not end up in a NE: Consider the initial schedule on $m_0 = 4$ machines given in Figure 4(a). Assume that the two right machines are removed and that $\delta = 1$.

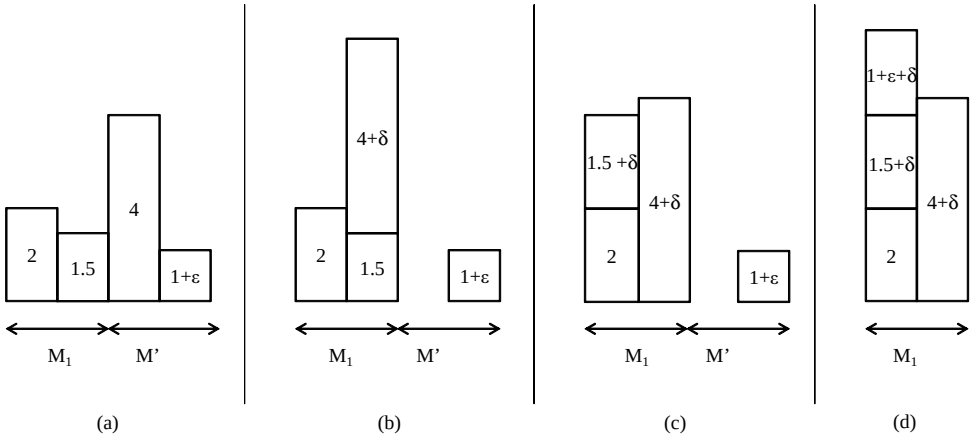

Figure 4: An example of single phase *max-length best-response* that does not converge to a NE.

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 4(b)). The job of length 2 is not interested in moving. Next, the job of length 1.5 moves (Figure 4(c)), and finally, the job of length 1 . Figure 4(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.

However, linear time convergence to a NE is guaranteed by the above *two-phase max-length best-RD*.

Theorem 3.2 Two-phase max-length best-RD *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 2.1.

∎

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 the discussion in Section 2.2, the PoS of the selfish load rebalancing game with removed machines and any job extension penalty is 1. As shown in Theorem 2.5, 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 better-RD, we can bound the PoA.

Theorem 3.4 *The PoA assuming that s_0 is a NE and the modified NE is reached by* better-RD *is at most m_1.*

Proof: Let n' be the number of jobs assigned to M' in s_0. It is easy to observe that along the application of better-RD, $L_{max} \leq \sum_j p_j + n'\delta$. Let $P = \sum_j p_j + n'\delta$. Thus, P is the maximal load that can be reached. Since n' job extensions are inevitable, we have $OPT \geq \frac{P}{m_1}$. Therefore, $PoA \leq \frac{P}{P/m_1} = m_1$. ∎

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

Theorem 3.5 *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 the NE is reached by* better-RD *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 present the construction of the lower bound in Figure 5. In this instance $m_0 = 3$ and $m' = 3$. The initial assignment is given in Figure 5(a).

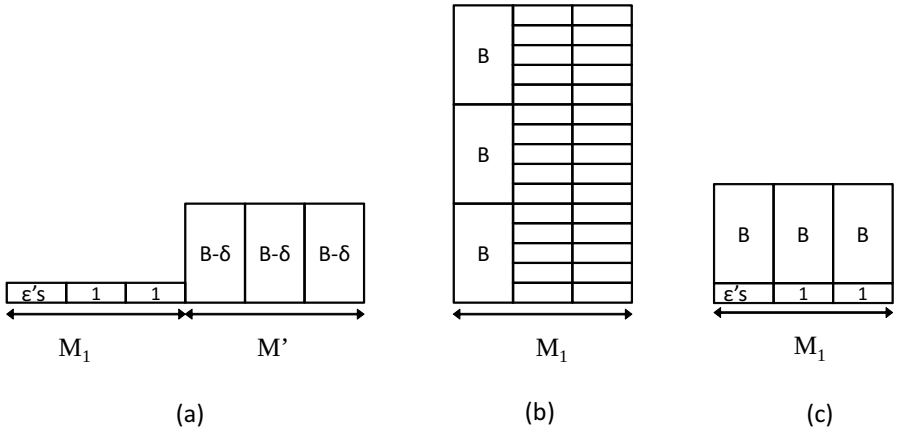

Figure 5: An instance achieving the maximal possible PoA by performing better-RD. (a) the initial assignment, (b) the worst NE, and (c) the best NE.

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 5(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 5(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-RD is m_1, a smaller bound can be shown if the NE is reached by *two-phase* better-RD. In the first phase, all the jobs that are assigned to machines in M' are activated, each performing its best move to a machine in M_1. In the second phase, all the jobs (now assigned to M_1) are activated, possibly several times, in an arbitrary order. As this is a specific application of better-RD, convergence to a NE is guaranteed.

Theorem 3.6 *The PoA assuming that s_0 is a NE and the modified NE is reached by* two-phase better-RD *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 performed 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 *two-phase best-RD*.

Theorem 3.7 *For any $m_1 > 1, m' > 2$, $\rho > 0$, there exists an initial NE schedule for which the PoA of a schedule achieved by* two-phase best-RD *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 present the construction of the lower bound in Figure 6. In this instance $m_0 = 6$ and $m' = 2$. The initial assignment is given in Figure 6(a).

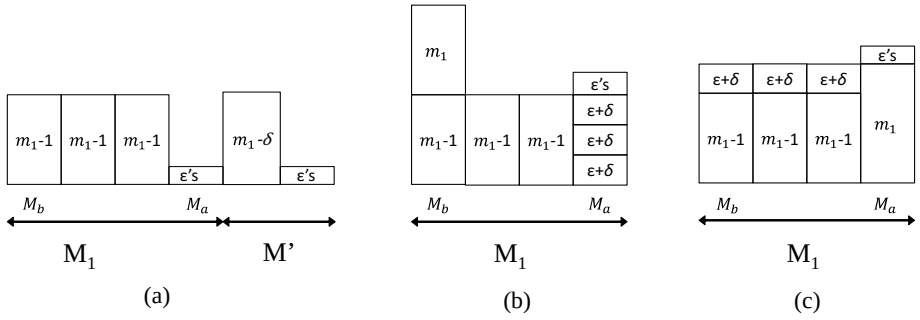

Figure 6: An instance achieving the maximal possible *PoA* by performing two-phase best-RD. (a) the initial assignment, (b) the worst NE, and (c) the best NE.

Assume that m' machines are removed and *two-phase best-RD* 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 6(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 6(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 a NE. The upper bound proof is similar to the proof of Theorem 2.8. The lower bound follows from Theorem 3.5.

Theorem 3.8 *If the initial assignment is not necessary a NE, and the modified schedule is reached by better-RD, then the PoA 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 perform 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.

It is not difficult to see (using the arguments used for the classic load balancing game [2]) that for any instance of the dynamic game, a SE exists. In particular, an assignment in which the

vector of loads is lexicographically minimal is a SE. On the other hand, as we show, finding a SE schedule, or deciding whether a NE schedule s is a SE is NP-hard. Moreover, given a set of jobs, it is NP-hard to determine whether this set has a beneficial coordinated deviation.

Theorem 4.1 *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.*

4.1 Equilibrium Inefficiency

We present tight bounds for the *strong price of anarchy*. By the discussion in Section 2.2, the *strong price of stability* is 1.

Theorem 4.2 *The SPoA of load rebalancing games with uniform extension 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, $\delta \leq OPT$. It is easy to observe that In any SE, at least one job has cost at most OPT, as otherwise, all jobs form together a coalition that prefers the optimal schedule. 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 $L_{min}(s) \leq OPT$ as observed above, we have $L_{max}(s) \leq L_{min}(s) + p_j + \delta \leq OPT + OPT + OPT = 3OPT$.

■

We show that the above analysis is tight even when the initial schedule is a SE in the cases of adding and removing machines. For simplicity, the instances below are described for specific values of m_0 and m'. It can be generalized by scaling and/or adding dummy jobs.

Theorem 4.3 *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 7(a). Note that each machine accommodates one long job of length B and one tiny job of length ε (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 7(b). We have $L_{max}(s) = 2B + \delta = 3B - \varepsilon$.

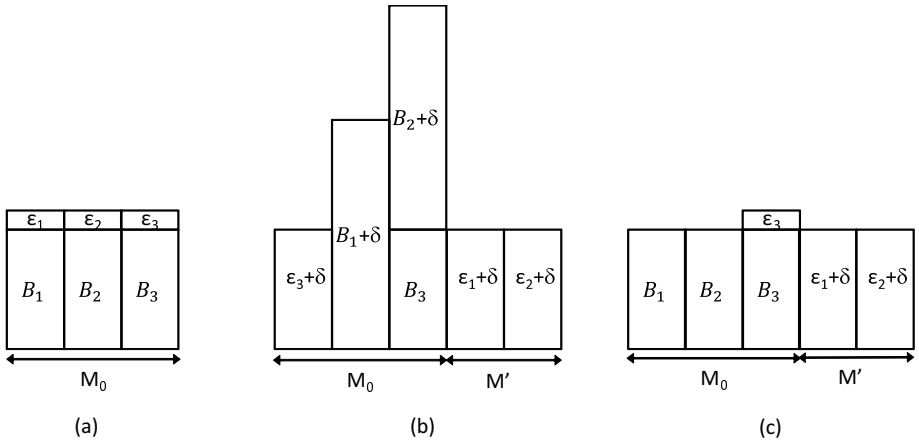

Figure 7: An instance achieving $SPOA = 3$. (a) the initial assignment, (b) a possible SE, and (c) the best SE.

We show that s is a SE by showing 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 will be 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 7(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 4.4 *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 8(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 8(b). We have $L_{max}(s) = 3B - \varepsilon$.

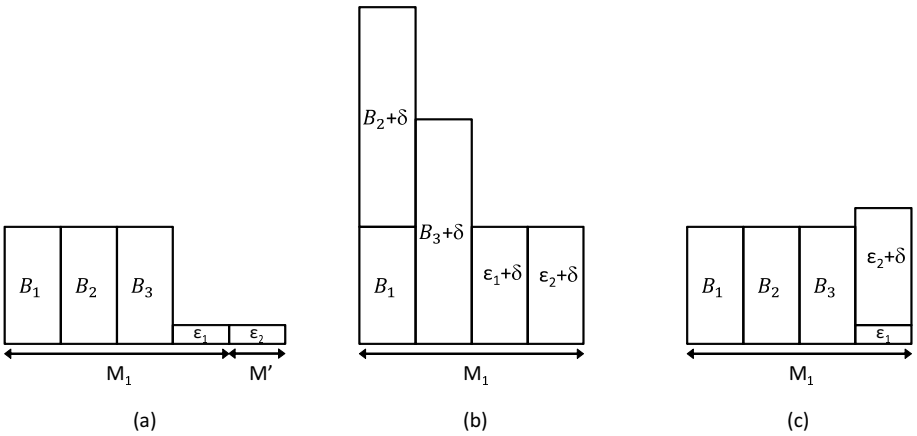

Figure 8: An instance achieving $SPOA = 3$. (a) the initial assignment, (b) a possible SE, and (c) the best SE.

The proof that s is a SE is similar to the one in the proof of Theorem 4.3. An optimal schedule for the modified instance is given in Figure 8(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 4.2. The examples in the proofs of Theorem 4.3 and 4.4 show that this bound is tight. Moreover, in Appendix B we show that this analysis is tight even if the initial schedule is a SE, and the final SE is reached by a sequence of coalitional improvement steps.

Theorem 4.5 *The SPoA of load rebalancing games with uniform extension penalty is at most* $2 + \frac{\delta}{OPT}$.

5 Summary and Future Work

We considered a dynamic variant of the classic load balancing game, in which machines are added or removed and job migrations are associated with job's extension

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, we explored the existence and calculation of Nash equilibrium and strong equilibrium and provided tight bounds for NE and SE inefficiency - in general and under various dynamics. Our results show that the existence of migration penalty might lead to poor stable schedules, however, if the modification is a result of a sequence of improvement steps or, better, if the sequence of improvement steps can be supervised in some way (by forcing the jobs play in a specific order, or select their best response) then the modified schedule approximates well an optimal one, with approximation ratio similar to the classic load balancing game. Thus, while migration costs discourage changes and increase the stability of any given configuration, it is still guaranteed that any stable configuration that is reached by natural dynamics has a reasonable social cost.

Possible directions for future work include the study of heterogenous systems, in particular unrelated machines, or non-uniform extension penalty. 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'. Another interesting variant is proportional extension, i.e., a migration of job j extends its processing time from p_j to $p_j(1+\delta)$.

Our analysis of coordinated deviation show that the SPoA heavily depends on the value of δ. Another direction for future work is to analyze instances in which δ is bounded by the instance parameters, e.g., when $\delta \leq p_{min}$. Analyzing equilibrium inefficiency with respect to the objective of minimizing the total players' cost is another possible direction.

Finally, in our model a migration of job 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 a 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. With individual penalties we can also consider accumulated migration costs – applied to jobs performing multiple migrations during better-RD. Such a model corresponds to systems in which the physical migration is associated with cost, unlike our model that suit systems in which the cost corresponds to some preprocessing or machine's set-up time.

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 *FOCS*, 2004.

[2] N. Andelman, M. Feldman, and Y. Mansour. Strong Price of Anarchy. In *SODA*, 2007.

[3] Y. Azar, K. Jain, and V.S. Mirrokni. (Almost) optimal coordination mechanisms for unrelated machine scheduling. In *SODA*, 2008.

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

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

[6] E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence time to Nash equilibria. In *ICALP*, 2003.

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

[8] A. Fiat, H. Kaplan, M. Levi, and S. Olonetsky. Strong Price of Anarchy for Machine Load Balancing. In *ICALP*, 2007.

[9] G. Finn and E. Horowitz. A linear time approximation algorithm for multiprocessor scheduling.. In *BIT*, vol. 19(3), pages 312–320, 1979.

[10] D. Fotakis, S. Kontogiannis, E. Koutsoupias, M. Mavronicolas, P. Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. In *ICALP*, 2002.

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

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

[13] 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.

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

[15] C. Papadimitriou. Algorithms, Games, and the Internet. In *STOC*, 2001.

[16] R. Paes Leme, V. Syrgkanis, É. Tardos, The curse of simultaneity. In *ITCS* 2012.

[17] B. Vöcking. *In N. Nisan, T. Roughgarden, E. Tardos and V. Vazirani, eds., Algorithmic Game Theory. Chapter 20: Selfish Load Balancing.* Cambridge University Press, 2007.

A Machines' Addition - Lower Bound for the PoA

Let $m' = km_0 + \alpha$ for integers k and $\alpha < m_0$. By Theorem 2.6, 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 2.7, 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 A.1 *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 present the construction of the lower bound in Figure 9. In this instance $m_0 = 4$ and $m' = 3$. The initial assignment is given in Figure 9(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.

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

[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 9(a)).

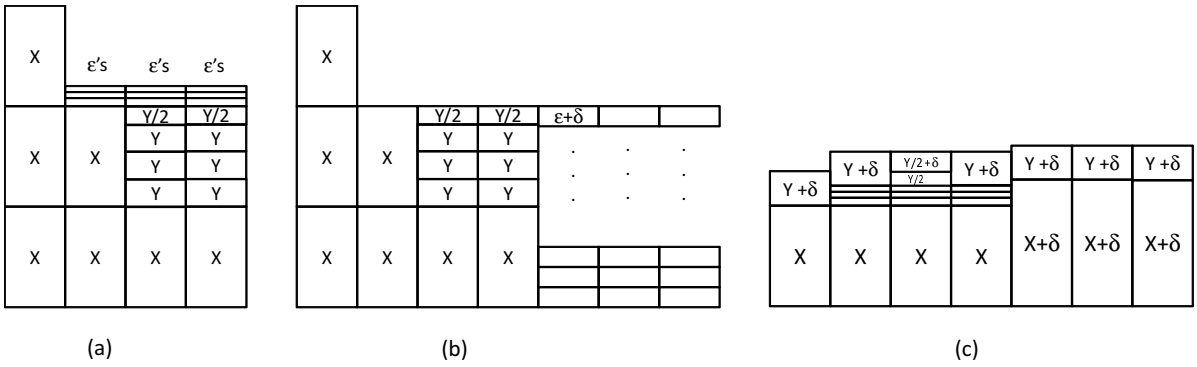

Figure 9: An instance achieving a high PoA for $m' < m_0$. (a) the initial assignment, (b) the worst NE, and (c) the best NE.

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[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 A.2 *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 present the construction of the lower bound in Figure 10. In this instance $m_0 = 3$ and $m' = 5$, thus, $k = 1$ and $\alpha = 2$. The initial assignment is given in Figure 10(a). It is easy to

[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.

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.

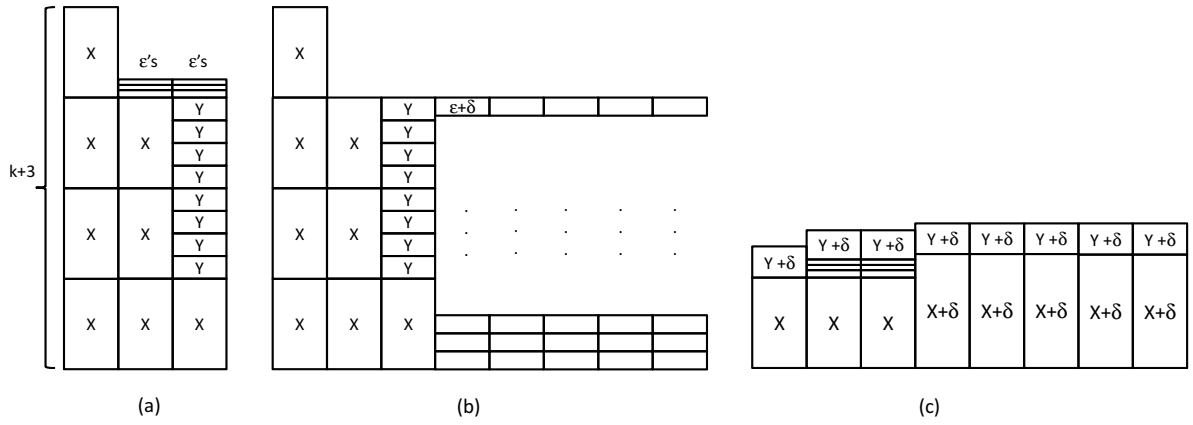

Figure 10: An instance achieving a high PoA for $k = 1$, $\alpha 21$. (a) the initial assignment, (b) the worst NE, and (c) the best NE.

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 $k + 2 + \frac{(\alpha-1)(k+3)-m_0+1}{m_0(k+2)+1} - \rho$. ∎

The lower bounds in Theorems $A.1$ and $A.2$ do not match the upper bound in Theorem 2.6. We believe that the upper bound can be reduced when $\alpha \neq 1$.

B Coordinated Deviations - Lower Bound for the SPoA

We show that the bound $SPoA \geq 2 + \frac{\delta}{OPT}$ is tight even if the initial schedule is a SE, and the final SE is reached by a sequence of coalitional improvement steps.

[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.

Theorem B.1 *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 11(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 $3B - 2\delta + 1$.

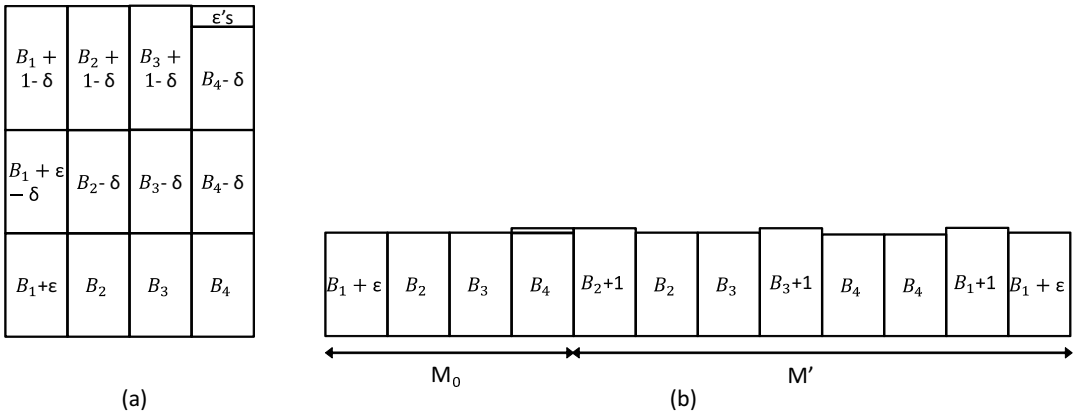

Figure 11: An instance achieving $SPOA = 2 + \frac{\delta}{OPT}$. (a) the initial assignment, (b) the best SE.

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 11(b) presents an optimal schedule after the modification. $L_{max}(OPT) = B + \varepsilon$. Figure 13 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 13(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 13(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 13(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 13(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 13(e).

Claim B.2 *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 11(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 B.3 *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 12(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 12(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 B.4 *The schedule s' is an SE.*

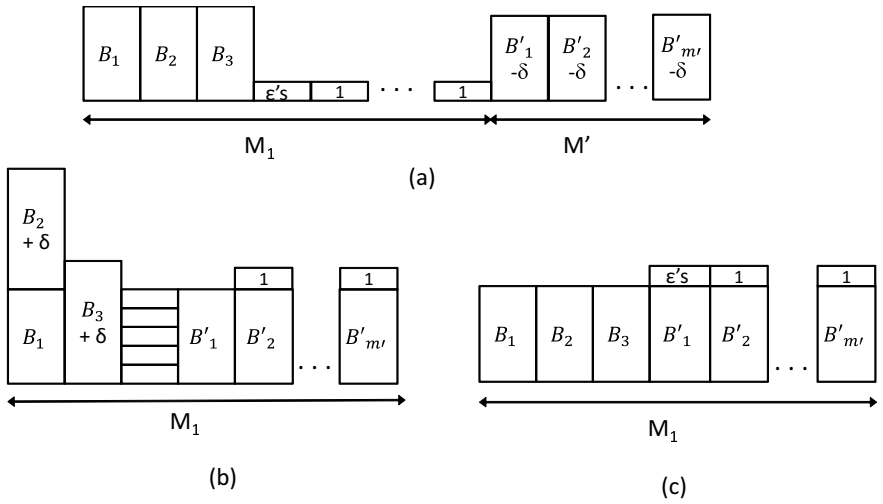

Figure 12: An instance achieving $SPOA = 2B + \delta$. (a) the initial assignment, (b) a possible SE, and (c) the best SE.

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. 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 12(c). $L_{max}(OPT) = B + 1$. Therefore,

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

■

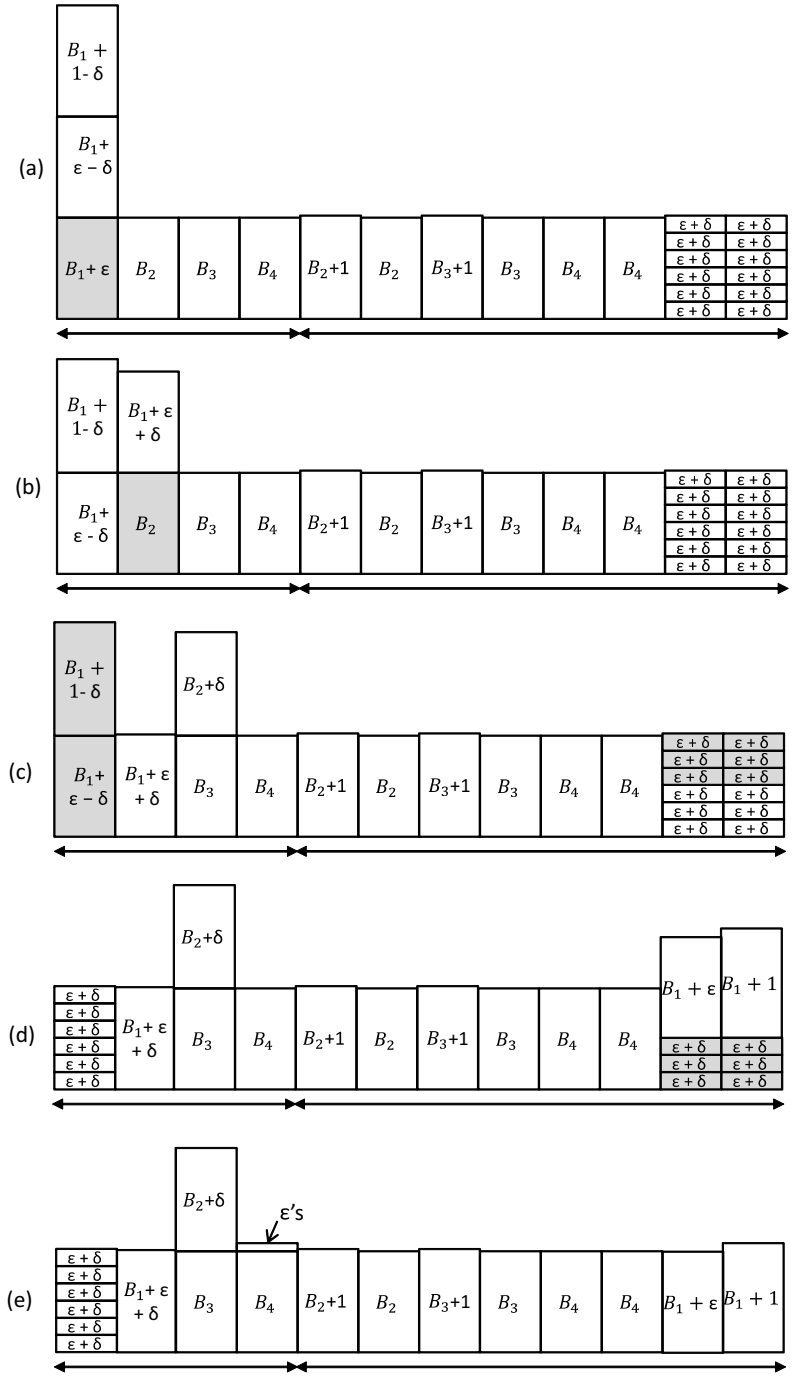

Figure 13: An instance achieving $SPOA = 2 + \frac{\delta}{OPT}$, where the SE is reached by a sequence of coalitional improvement steps. The jobs forming the coalitions are in grey.