# Bounds for the Convergence Time of Local Search in Scheduling Problems*

Tobias Brunsch, Michael Etscheid, and Heiko Röglin

Department of Computer Science, University of Bonn, Germany
{brunsch,etscheid,roeglin}@cs.uni-bonn.de

**Abstract.** We study the convergence time of local search for a standard machine scheduling problem in which jobs are assigned to identical or related machines. Local search corresponds to the best response dynamics that arises when jobs selfishly try to minimize their costs. We assume that each machine runs a coordination mechanism that determines the order of execution of jobs assigned to it. We obtain various new polynomial and pseudo-polynomial bounds for the well-studied coordination mechanisms Makespan and Shortest-Job-First, using worst-case and smoothed analysis. We also introduce a natural coordination mechanism FIFO, which takes into account the order in which jobs arrive at a machine, and study both its impact on the convergence time and its price of anarchy.

## 1 Introduction

We analyze the following scheduling problem: Given $m$ machines and $n$ jobs, find an assignment of the jobs to the machines minimizing the maximum costs of a job, which are defined according to a coordination mechanism. The jobs may have different job sizes and the machines may have different machine speeds. A typical definition of the costs of a job is the sum of the job sizes assigned to the same machine divided by the machine speed, which is a natural choice when the makespan is to be minimized. In other contexts it might be more realistic to assume an order in which the jobs on a machine are executed and that a job only pays for the execution time of itself and all previous jobs.

Even in the case of identical machine speeds, the problem is known to be strongly NP-hard [10] and local search is a popular tool to approximate good solutions. Here, a job unilaterally changes its assignment and moves to another machine if it can reduce its costs this way. Throughout this paper, we assume a best response policy, i.e., a moving job selects a machine that minimizes its costs. If there is no job left that can improve its costs, we have attained a local optimum, which is guaranteed to be reached after a finite number of steps. Although the quality of the worst local optimum has been thoroughly analyzed [4,6,7,9,17], there is not much work about the convergence time needed to find one via local search.

### 1.1 Terminology

Let us first describe the studied problem in detail. Consider an instance with $m$ machines and $n$ jobs. Each machine $i$ has a *speed* $s_i \in \mathbb{Q}_{>0}$ and each job $j$ has a

* This research was supported by ERC Starting Grant 306465 (BeyondWorstCase).

*job size* $p_j \in \mathbb{Q}_{>0}$. Let $s_{\min}, s_{\max}, p_{\min}$, and $p_{\max}$ be the minimal and maximal speeds and job sizes. Let $W = \sum_{j=1}^{n} p_j$ be the sum of the job sizes. For *identical machines*, $s_{\max} = s_{\min} = 1$, and for *unit-weight jobs*, $p_{\max} = p_{\min} = 1$.

For an assignment $\sigma\colon \{1, \dots, n\} \to \{1, \dots, m\}$ that maps the jobs to the machines, let $L_i = \sum_{j \in \sigma^{-1}(i)} p_j / s_i$ be the *load* of machine $i$. The maximum load is called *makespan*. The *costs* of a job $j$ are defined according to a *coordination mechanism*, which assigns costs to every job depending only on the set of jobs that have chosen the same machine, but not on the residual schedule.

1. In the *Makespan* model, all jobs assigned to the same machine are executed simultaneously such that the costs $c_j^\sigma = L_{\sigma(j)}$ of a job $j$ correspond to the load of its machine. This is the most common coordination mechanism and it corresponds to linear weighted congestion games on parallel links.

2. In the *FIFO* model, the jobs on each machine are executed one after another. Therefore, we need a permutation $\pi$ on the jobs that determines the order in which the jobs on a machine get processed. The costs of a job $j$ are then $c_j^{(\sigma,\pi)} = \sum_{j' \in J_\sigma^\pi(j)} \frac{p_{j'}}{s_{\sigma(j)}}$, where $J_\sigma^\pi(j)$ is the set of jobs $j'$ on the same machine with $\pi(j') \leq \pi(j)$. If a job $j$ jumps to another machine, it is inserted as the last job, i.e., $\pi(j) = n$.

3. In the *SJF* (shortest job first) model, the jobs are executed one after another, but the permutation of the jobs is at any time implicitly given by their job sizes where the smallest job on a machine is executed first. Ties for jobs of equal size are broken arbitrarily. This means that the costs of a job are defined as $c_j^\sigma = \sum_{j' \colon \sigma(j') = \sigma(j) \wedge \pi(j') \leq \pi(j)} \frac{p_{j'}}{s_{\sigma(j)}}$, where $\pi$ is a permutation of the jobs assigned to machine $\sigma(j)$ such that $\pi(j') < \pi(j)$ if $p_{j'} < p_j$ and $\pi(j') > \pi(j)$ if $p_{j'} > p_j$.

The FIFO model is not a coordination mechanism in the classical sense as the order in which the jobs are executed depends on previous iterations. Nevertheless, we believe that this model can easily be motivated by many real-world applications where the first-come, first-served principle is ubiquitous.

In the case of the Makespan and SJF models, we call $\sigma$ a *schedule*. In the FIFO model, we call the tuple $(\sigma, \pi)$ a *schedule*. Often, we omit the parameters $\sigma$ and $\pi$ if they are clear from the context, or we replace them by an iteration number $t$. Then we mean the schedule before the move of iteration $t$ gets executed.

We say that a job is *unsatisfied* if it could improve its costs by jumping to a different machine. When an unsatisfied job jumps, it always jumps to a machine minimizing its costs, i.e., we consider best response dynamics. If there is no unsatisfied job, we call the current schedule a *local optimum*. The *convergence time* for an instance is the maximum number of jumps it can take starting from an arbitrary schedule until a local optimum is reached. The *price of anarchy* is the ratio of the makespans of the worst local optimum and the global optimum.

If there are several unsatisfied jobs, we choose the next job to jump according to a pivot rule:

- *Best Improvement*: Select a job for which the largest improvement of its costs is possible.
- *Random*: Select a job uniformly at random from the set of unsatisfied jobs.
- *Min Weight*: Select a smallest unsatisfied job.
- *Max Weight*: Select a largest unsatisfied job.
- *Fixed Priority*: Select the unsatisfied job with the largest priority according to a given order on the jobs. This pivot rule includes Min Weight and Max Weight as special cases.

### 1.2 Smoothed analysis

Despite its bad running time, which can be exponential, and the large price of anarchy in theory, local search is a popular tool in practice as it typically delivers good local optima very quickly. At first glance, this seems like a contradiction, but the instances in the theoretical proofs are rather contrived and rarely observed in practice. To have a more realistic understanding of local search in theory, we use the framework of smoothed analysis introduced by Spielman and Teng [18] to explain the practical success of the simplex method. This model can be considered as a less pessimistic variant of worst-case analysis in which the adversarial input is subject to a small amount of random noise and it is by now a well-established alternative to worst-case analysis. This random noise can be motivated, for example, by measurement errors, numerical imprecision, and rounding errors, which often occur in practice. It can also model influences that cannot be quantified exactly but for which there is no reason to believe that they are adversarial.

We follow the more general model of smoothed analysis introduced by Beier and Vöcking [2]. In this model, the adversary is even allowed to specify the probability distribution of the random noise. The influence he can exert is described by a parameter $\phi \geq 1$ denoting the maximum density of the noise. The model is formally defined as follows.

**Definition 1.** *In a* $\phi$-smooth *instance* $\mathcal{I}$*, the adversary chooses the following input data:*

- *the number* $m$ *of machines;*
- *arbitrary machine speeds* $s_1, \ldots, s_m$ *in the case of non-identical machines;*
- *the number* $n$ *of jobs;*
- *for each* $p_j$*, a probability density* $f_j : [0, 1] \to [0, \phi]$ *according to which* $p_j$ *is chosen independently of the processing requirements of the other jobs.*

*The* smoothed convergence time *is the worst expected convergence time of any* $\phi$*-smooth instance and the* smoothed price of anarchy *is the worst expected price of anarchy of any* $\phi$*-smooth instance.*

Note that the only perturbed part of the instance are the processing requirements. Formally, a $\phi$-smooth instance is not a single instance but a distribution over instances. The parameter $\phi$ determines how powerful the adversary is. He can, for example, define

an interval of length $1/\phi$ for each job size from which it is drawn uniformly at random. Hence, for $\phi = 1$ the model corresponds to an average-case analysis and for $\phi \to \infty$ the adversary becomes as powerful as in a worst-case analysis.

## 2 Related Work and Results

Since its invention, smoothed analysis has been successfully applied in a variety of contexts. Two surveys [15,19] summarize some of these results.

The notion of coordination mechanisms has been introduced by Christodoulou et al. [5] in the context of congestion games. There has been extensive research about the price of anarchy for the different coordination mechanisms. In the Makespan model it is constant for identical machines [9,17] and $\Theta\left(\min\left\{\frac{\log m}{\log\log m}, \log\frac{s_{\max}}{s_{\min}}\right\}\right)$ for related machines [6]. The smoothed price of anarchy for related machines is $\Theta(\log\phi)$ regardless of whether the job sizes [4] or the machine speeds [7] are perturbed.

Immorlica et al. [13] showed a price of anarchy of $2 - 1/m$ for identical and $\Theta(\log m)$ for related machines for the SJF model, which is the same as for list schedules, i.e., schedules that are generated by a greedy assignment.

The FIFO model has been introduced implicitly by Brunsch et al. [4] through the equivalent concept of near list schedules which was used as a generalization of local optima w.r.t. the Makespan model and list schedules. They showed that the smoothed price of anarchy is $\Theta(\log\phi)$. We complement this by the corresponding worst-case results for identical and related machines to obtain the same tight bounds as in the SJF model.

There is less known about the convergence times in the different models. As we are up to our knowledge the first ones who consider the FIFO model, there are no previous results about convergence times. We show tight results for special cases like identical machines and several upper bounds depending on $W/p_{\min}$ for different pivot rules in the general case. Although we conjecture polynomial bounds for all cases, we give the first non-trivial proofs for this natural problem. Immorlica et al. [13] showed for the SJF model that if the jobs are asked on a rotational basis if they want to jump, the convergence time is in $O(n^2)$. This is in sharp contrast to our result that for the Min Weight pivot rule it can take an exponential number of iterations even in the case of two identical machines.

Brucker et al. [3] considered the Makespan model with the difference that only jobs from a machine with maximum load—a so-called *critical* machine—are allowed to jump, i.e., a local optimum is reached as soon as every job on a critical machine is satisfied. They gave an algorithm that finds a local optimum after $O(n^2)$ improving steps for identical machines. From this, one can easily derive an algorithm for identical machines in the Makespan model: Run Brucker's algorithm exhaustively until every job on a critical machine is satisfied. As on identical machines the minimum load of a machine is monotonically increasing, these jobs cannot become unsatisfied again by any sequence of improving steps. Hence, the jobs on the critical machine are fixed and therefore we can remove the critical machine together with its assigned jobs from the instance. Repeating this argument yields a running time of $O(n^2\dot{m})$ improving steps.

As the monotonicity argument does not hold anymore in the case of related machines, we are not aware of a way to use similar results by Schuurman and Vredeveld [17] and Hurkens and Vredeveld [12] for Brucker's model on related machines.

In the Makespan model, Even-Dar et al. [8] showed for identical machines that the Max Weight and the Random pivot rule converge in $n$ and $O(n^2)$ steps, respectively, while the Min Weight pivot rule can take an exponential number of steps. They also showed an upper bound of $(n/m+1)^{m-1}$ for any pivot rule for identical machines. We extend this result by showing that every pivot rule converges in $O(n \cdot W/p_{\min})$ steps, which can also be seen as a generalization of their result that every pivot rule converges in $O(W+n)$ steps in the case of integer weights. For related machines and unit-weight jobs, Even-Dar et al. [8] showed that there is a pivot rule that converges in $mn$ steps. We improve this by showing that the convergence time for any pivot rule with best response policy is exactly $n$. For the case of related machines and integral job sizes and machine speeds, they showed that any pivot rule converges in $O(W^2 \cdot s_{\max}^2/s_{\min})$ steps. We prove a similar bound for the Best Improvement pivot rule on arbitrary weights. An overview of our results on convergence times is given in Table 1, Table 2, and Table 3.

| | | |
|---|---|---|
| identical machines | $n-1$ | (Thm. 1) |
| unit-weight jobs | $n$ | (Thm. 4) |
| two machines | $\Theta(n)$ | (Thm. 5) |
| Best Improvement | $O(m^2 n \cdot W/p_{\min})$ | (Thm. 6) |
| Random | $O(m^2 n^2 \cdot W/p_{\min})$ | (Thm. 7) |
| Fixed Priority | $O(n^2 \cdot W/p_{\min})$ | (Thm. 8) |
| lower bounds | $\Omega(mn)$, $\Omega(m^2)$ for Min Weight | (Thm. 9) |

**Table 1.** FIFO convergence times

| | | |
|---|---|---|
| identical machines | $O(n \cdot W/p_{\min})$ | (Thm. 2) |
| unit-weight jobs | $n$ | (Thm. 4) |
| Best Improvement | $O(m^2 n \cdot W^2/p_{\min}^2)$ | (Thm. 10) |

**Table 2.** Makespan convergence times

| | | |
|---|---|---|
| Max Weight on two identical machines | $2^{\Omega(n)}$ | (Thm. 3) |
| Max Weight on two identical machines with random weights | $2^{\Omega(\sqrt{n})}$ | (Thm. 3) |
| Min Weight | $n$ | (Thm. 11) |
| Random | $O(n^2)$ | (Thm. 11) |

**Table 3.** SJF convergence times

### 2.1 Paper organization

The remainder of this paper is organized as follows. In Section 3 we show how to convert superpolynomial deterministic convergence times to smoothed polynomial convergence times. In Section 4 and Section 5 we deal with the special cases of identical machines and unit-weight jobs, respectively, before we turn to the more general case of related machines in Section 6. We conclude with the analysis of the price of anarchy in the FIFO model in Section 7 and some remarks in Section 8. Deferred proofs can be found in the appendix.

## 3 Smoothed Analysis

Some of our shown convergence times include the factor $W/p_{\min}$. While in the worst case this fraction can be exponentially large, in the smoothed setting they turn into expected polynomial convergence times.

**Lemma 1.** *If the convergence time is bounded by $f(m,n) \cdot W/p_{\min}$ for some polynomial $f$, then the smoothed convergence time is bounded by $O(f(m,n) \cdot n^3 \log(m) \cdot \phi)$.*

Unfortunately, our result about the convergence time of the Best Improvement pivot rule in the Makespan model depends quadratically on $p_{\min}$. This does not allow us to derive an expected polynomial convergence time, but instead we can show that with high probability the convergence time is polynomially bounded.

**Lemma 2.** *The convergence time of the Best Improvement pivot rule in the Makespan model is in $m^2 n^7 \phi^2$ with probability at least $1 - 1/n$.*

## 4 Identical Machines

In the FIFO model, the costs of a job decrease monotonically while the minimum load of a machine increases monotonically when considering identical machines. As a moving job always jumps to a machine with minimum load, every job can jump at most once. This leads to the following result.

**Theorem 1.** *In the FIFO model, for any pivot rule the worst-case running time is exactly $n - 1$.*

For the Makespan model, Even-Dar et al. [8] proved that the Min Weight pivot rule can take as many as $\Omega((n/m^2)^{m-1})$ steps. They also showed that this is near to the worst case as every pivot rule terminates after $O((\frac{n}{K}+1)^K)$ steps, where $K$ is the number of different job weights. We derive the bound $O(n \cdot \frac{W}{p_{\min}})$ for arbitrary pivot rules, which is a significant improvement if $\frac{W}{p_{\min}}$ is small. This bound is almost optimal as it is easy to see that the worst case instance used in [8] has $\frac{p_{\max}}{p_{\min}} = (n/(m-1))^{m-2}$. It is also a generalization of the result that every pivot rule converges in $O(W+n)$ steps in the case of integer weights.

**Theorem 2.** *In the Makespan model, every pivot rule terminates after $O(n \cdot \frac{W}{p_{\min}})$ steps.*

*Proof.* As Even-Dar et al. [8] pointed out (without proof), after a job $j$ moved to machine $i$, it can only be unsatisfied again after a strictly greater job moved to machine $i$ in the meantime: A job always jumps to a machine with minimum load and the minimum load increases monotonically. Consider the last job $j'$ entering machine $i$ in iteration $t'$ before job $j$ jumps away from machine $i$ in iteration $t$. Then machine $i$ must be a machine with minimum load before iteration $t'$. Now if $p_j > p_{j'}$, then $L_i^{t+1}$ would by stricly smaller than $L_i^{t'}$, which is a minimum load in a former iteration. If $p_j = p_{j'}$, then job $j$ cannot be unsatisfied because job $j'$ is not unsatisfied.

Based on their idea of push-out potentials, we define the potential $\phi := \sum_{i=1}^{m} u_i^t \leq W$, where $u_i^t$ is the maximum total weight of jobs on machine $i$ that could consecutively move away from $i$, starting in the schedule before iteration $t$. When a job $j$ jumps from machine $i$ to machine $i'$, then $u_{i'}^t$ was 0 beforehand. As mentioned above, no job from any other machine than machine $i'$ can become unsatisfied by the move of job $j$ and thus the potential $\phi$ decreases by at least $u_i^t - u_i^{t+1} - u_{i'}^{t+1}$.

If $u_{i'}^{t+1}$ was larger than $p_j$, then there would be a sequence of moves from jobs away from machine $i'$ such that the load of machine $i'$ after these moves would be less than $L_{i'}^t$. But machine $i'$ was a machine with minimum load before iteration $t$, a contradiction. Note also that $u_i^t - u_i^{t+1} \geq p_j$: Let $J'$ be the jobs on machine $i$ with total weight $u_i^{t+1}$ which could consecutively jump away from machine $i$ after iteration $t$. Then $J' \cup \{j\}$ could consecutively jump away before iteration $t$ and thus $u_i^t \geq \sum_{j' \in J' \cup \{j\}} p_{j'} = u_i^{t+1} + p_{j'}$. We can conclude that $u_i^t - u_i^{t+1} - u_{i'}^{t+1} \geq 0$ and thus that $\phi$ is actually a potential.

We call a jump of job $j$ to machine $i$ in iteration $t$ *stable* if after that jump, another job moves to $i$ before a job leaves $i$. As discussed above, through the stable jump the total potential of all machines except machine $i$ decreases by at least $p_j \geq p_{\min}$ and $u_i^{t'} = 0$ at time $t'$ when the next job enters or leaves machine $i$. Hence, every stable jump induces a potential drop of at least $p_{\min}$. We maintain a set of indices: In the initial schedule, every job has an index attached to it. When a job $j$ moves away from machine $i$, then the indices attached to $j$ get transferred to the job $j'$ that moved last to machine $i$. If no such job exists, the indices get deleted. Afterwards, a new index gets attached to job $j$ on its new machine if it was a stable jump.

When a job $j$ moves to machine $i$, then no job on machine $i$ was unsatisfied beforehand as $j$ jumps to a machine with minimum load. Thus, when an index gets reattached from job $j'$ to job $j$, then $j$ made $j'$ unsatisfied and thus $p_j$ is strictly greater than $p_{j'}$ because only larger jobs can make smaller jobs unsatisfied. Therefore, every index can be reattached at most $n$ times. Furthermore, every time a job $j$ jumps away from a machine $i$, it has at least one index attached to it: Assume to the contrary that it is the first jump without attached indices. If it is the first jump by job $j$ or its last jump was stable, then there is by definition an attached index. Otherwise, there is a job $j'$ that left machine $i$ such that job $j$ is the last job entering machine $i$ beforehand and thus job $j'$ transferred its indices to job $j$. Hence, the number of indices is at least one $n$th of the total number of jumps. There can only be $W/p_{\min}$ many stable jumps as otherwise $\phi$ would be negative. This yields the desired bound. □

Finally let us consider the SJF model. The Max Weight pivot rule in the SJF model can take an exponential number of steps even on two identical machines. Also an average-case analysis yields a superpolynomial convergence time. We do not consider the Random pivot rule and the Min Weight pivot rule in this section because for these rules we prove in Section 6.3 polynomial upper bounds even for the more general setting of related machines. We leave it as an open question whether the convergence time of the Best Improvement pivot rule is polynomial for identical machines.

**Theorem 3.** *In the SJF model, the convergence time of the Max Weight pivot rule is $2^{\Omega(n)}$ even for two identical machines. The smoothed convergence time of the Max Weight pivot rule is $2^{\Omega(\sqrt{n})}$ even for two identical machines and $\phi = 1$.*

## 5 Unit-Weight Jobs

In the case of unit-weight jobs, Even-Dar et al. [8] claimed that for Makespan, there exists a pivot rule which converges in $mn$ steps and that there is a pivot rule with convergence time $\Omega(mn)$ if jobs do not necessarily move to the machine yielding the biggest improvement but only have to improve their costs by jumping. We show that all pivot rules have linear convergence time if jobs have to jump to the best machine.

**Theorem 4.** *In both the FIFO and the Makespan model for unit-weight jobs, the convergence time for any pivot rule is $n$ for any number $m \geq 2$ of machines.*

## 6 Related Machines

For the most general case of related machines we use potential functions in order to show pseudo-polynomial convergence times for different pivot rules in both the FIFO and the Makespan model.

The potential $\phi_{\text{FIFO}}$ used in the FIFO model is the Rosenthal potential introduced in [16], which is the sum of the execution times of the jobs. It is easy to see that $\phi_{\text{FIFO}}$ decreases by at least $\Delta$ when the jumping job improves its execution time by $\Delta$. It decreases even more if the jumping job was not on top of its original machine.

For the Makespan model, we use the potential

$$\phi_{\text{Makespan}} := \sum_{i=1}^{m} \frac{1}{s_i} \cdot \left( \left( \sum_{j \in \sigma^{-1}(i)} p_j \right)^2 + \sum_{j \in \sigma^{-1}(i)} p_j^2 \right),$$

defined by Even-Dar et al. [8].

The fastest machine has always load at most $W/s_{\max}$. If there is a machine with load greater than $2W/s_{\max}$, then a job from this machine can improve its costs by at least $W/s_{\max}$ by jumping to the fastest machine. This gives rise to the following lemma.

**Lemma 3.** *The following two statements hold:*

1. *If there is a machine with load greater than $2W/s_{\max}$, the best improvement can be achieved by a jump from some job from a machine with load greater than $W/s_{\max}$ to a machine with load at most $W/s_{\max}$.*
2. *If there is no machine with load greater than $2W/s_{\max}$, then $\phi_{FIFO} = O(n \cdot \frac{W}{s_{\max}})$ and $\phi_{Makespan} = O(\frac{W^2}{s_{\max}})$.*

**Corollary 1.** *For the Best Improvement pivot rule after $n$ iterations and for the Random pivot rule after expected $O(n \log n)$ iterations there is no machine left with load greater than $2W/s_{\max}$.*

### 6.1 FIFO

Before we come to the general cases, let us first mention a linear-time result for the special case of $m = 2$ machines.

**Theorem 5.** *In the FIFO model, the convergence time for any pivot rule on two related machines is at least $n$ and at most $2n - 2$. There are pivot rules for which $2n - 2$ is tight.*

The main idea of the following proofs is that if a job jumps that is not on top of its machine, the costs of all jobs above the moving job and thus the potential $\phi_{\text{FIFO}}$ decrease by at least $p_{\min}/s_{\max}$. We are able to show that this must happen after a polynomial number of steps for the Best Improvement and for Fixed Priority pivot rules.

**Theorem 6.** *In the FIFO model, the convergence time of the Best Improvement pivot rule is in $O(m^2 n \cdot W/p_{\min})$.*

*Proof.* According to Lemma 3 and Corollary 1, after $O(n)$ iterations the potential $\phi_{\text{FIFO}}$ is in $O(n \cdot W/s_{\max})$. Hence, it suffices to show that in every sequence of $m^2$ consecutive iterations, $\phi_{\text{FIFO}}$ drops by at least $p_{\min}/s_{\max}$. Therefore, let us consider a sequence $S$ of maximum length in which $\phi_{\text{FIFO}}$ drops by strictly less than $p_{\min}/s_{\max}$. It is obvious that only jobs that are on top of some machine can jump as the running times of all the jobs above the moving job decrease by at least $p_{\min}/s_{\max}$.

For a given point in time, we call a job *active* if it jumps until the end of the sequence $S$. At any time, there can only be at most one active job on any machine. To see this, assume to the contrary that there are two active jobs $j_1$ and $j_2$ at the same time $t_1$ on a machine $i$. Let job $j_1$ w.l.o.g. be directly above job $j_2$, and let $t_2 > t_1$ be the first iteration in which job $j_2$ leaves machine $i$ again. Define $\alpha := c_{j_1}^{t_1} - c_{j_1}^{t_2}$ as the difference of $j_1$'s running times at time $t_1$ and $t_2$. As job $j_2$ was a top-most job in iteration $t_2$ and no job below $j_2$ could jump before $j_2$ jumped, job $j_1$ would have a running time of $L_i^{t_1} - p_{j_2}/s_i$ if it jumped to machine $i$ in the next step, yielding a total improvement of $j_1$'s running time of at least $p_{\min}/s_{\max}$. If $j_1$ does not jump back to machine $i$ in the next step, then either we have reached an equilibrium (then $p_{j_2}/s_i \leq \alpha$) or there is a job (possibly also $j_1$) that can improve by strictly more than $p_{j_2}/s_i - \alpha$.

Hence, the potential drops by at least $p_{j_2}/s_i \geq p_{\min}/s_{\max}$ during all the jumps of job $j_1$ between $t_1$ and $t_2$ and the iteration following $t_2 + 1$.

Thus, we have shown that also at the beginning of the sequence $S$ there are at most $m$ active jobs as on each machine there is at most one active job. It also implies that no job $j$ can jump back to a machine $i$ it has already been onto as all jobs lying underneath $j$ stay on machine $i$ until the end of the sequence $S$. Hence, every job jumps at most $m-1$ times and the length of $S$ is bounded from above by $m(m-1)$. □

**Theorem 7.** *In the FIFO model, the expected convergence time of the Random pivot rule is in* $O(m^2n^2 \cdot W/p_{\min})$.

For Fixed Priority pivot rules, we cannot assume anymore that after a linear number of iterations there is no machine with load more than $2W/s_{\max}$ left and thus that $\phi_{\text{FIFO}}$ is small. On the other hand, we know that the sum of the running times of all jobs that have already jumped is bounded by $O(n \cdot W/s_{\max})$ and we are able to show that during $O(n)$ consecutive iterations, either a job jumps for the first time or the potential $\phi_{\text{FIFO}}$ drops by at least $p_{\min}/s_{\max}$. In order to bound the potential by $O(n \cdot W/s_{\max})$, we use the modified potential function

$$\phi'_{\text{FIFO}} := \sum_{j=1}^{n} \min\left\{c_j, \frac{W}{s_{\max}}\right\}.$$

**Theorem 8.** *In the FIFO model, the convergence time of any Fixed Priority pivot rule is in* $O(n^2 \cdot W/p_{\min})$.

*Proof.* As $0 \leq \phi'_{\text{FIFO}} \leq n \cdot W/s_{\max}$, we only have to show that during every sequence of $n+1$ steps, either $\phi'_{\text{FIFO}}$ drops by at least $p_{\min}/s_{\max}$ or a job must jump for the very first time. In such a sequence, it must be the case that a job $j_2$ jumps directly after a job $j_1$, where the priority of $j_2$ is greater than the priority of $j_1$. This means that $j_2$ jumps to the old machine $i$ of job $j_1$ as it could not jump before the move of $j_1$. If it was not $j_1$'s first jump, let $t_2$ be the point in time between the two jumps by $j_1$ and $j_2$, and let $t_1$ be the point in time before $j_1$ jumps the last time before $t_2 - 1$. As $j_2$ does not want to jump to machine $i$ at time $t_1$, but does this later at time $t_2$, it must be the case that $L_i^{t_1} > L_i^{t_2}$. Hence, between $t_1+1$ and $t_2-1$ a job $j'$ assigned to machine $i$ at time $t_1$ must leave its machine. But during this time, job $j_1$ lies above job $j'$ yielding a running time improvement of $p_{j'}/s_i \geq p_{\min}/s_{\max}$ for job $j_1$ through the jump by $j'$. As $j_1$ has jumped before, its running time before the jump by $j'$ was already at most $W/s_{\max}$, meaning that also $\phi'_{\text{FIFO}}$ drops by at least $p_{\min}/s_{\max}$. □

The machine speeds do not occur in our bounds for the convergence times. Nevertheless, different machine speeds result in a higher convergence time than in the case of identical machines, as the following result shows. We believe that our proofs for the upper bounds on the convergence times are too pessimistic and thus we conjecture polynomial convergence times for all pivot rules. This is in contrast to the superpolynomial lower bounds in the Makespan and SJF model but a crucial difference is that the costs of a job can never increase in the FIFO model.

**Theorem 9.** *In the FIFO model, local search can take $\Omega(mn)$ steps. The convergence time for the Min Weight pivot rule is in $\Omega(m^2)$.*

*Proof.* For the lower bound $\Omega(mn)$, let $\ell \geq 1$ and $k \geq 1$ be two integers. There are $m = 2k+1$ machines and $n = k\ell + k + 1$ jobs split up in $2k+1$ job classes $J_1, \ldots, J_{2k+1}$. The machine speeds are $s_i = 2^{i-1}$ for $1 \leq i \leq 2k$ and $s_{2k+1} = 2^{2k+1}$. The job classes $J_1, \ldots, J_k$ each contain $\ell$ jobs with sizes $2^0, \ldots, 2^{\ell-1}$ and the job classes $J_{k+1}, \ldots, J_{2k+1}$ each contain a single job with size $2^{\ell+j}$ for job class $J_j$.

Initially, each job class $J_j$ is assigned to machine $j$ and the jobs on a machine are processed in monotonically increasing order of the job sizes. We consider the following $k$ rounds $1, \ldots, k$. Before round $i$ begins, the jobs from job class $J_j$, $j \leq k$, are on machine $j + i - 1$ such that they are processed in increasing order of the sizes, the jobs from job classes $J_{k+1}, \ldots, J_{k+i-1}$ are on machine $2k+1$ and the other jobs have not moved before. Then we let the single job from class $J_{k+i}$ move from machine $k+i$ to machine $2k+1$. Thereupon, the jobs from class $J_k$ move in ascending order of the sizes from machine $k+i-1$ to machine $k+i$, the jobs from class $J_{k-1}$ move in ascending order of the sizes from machine $k+i-2$ to machine $k+i-1$ and so on. One can easily see that every job strictly decreases its costs while moving. All jobs from the job classes $J_1, \ldots, J_k$ move in every round. Hence, there are $\Omega(k^2\ell) = \Omega(mn)$ iterations.

For the lower bound $\Omega(m^2)$ for the Min Weight pivot rule, let again $k$ be an integer and let $\varepsilon > 0$ be appropriately small. There are $m = n = 2k+1$ machines and jobs. The machine speeds are $s_i = 1 + i \cdot \varepsilon$ for $1 \leq i \leq 2k$ and $s_{2k+1} = 4k$. The job sizes are $p_j = 1 - j \cdot \varepsilon$ for $1 \leq j \leq k$, $p_j = 2 + 2j \cdot \varepsilon$ for $k+1 \leq j \leq 2k$, and $p_{2k+1} = 4k$. Initially, every job $j$ is assigned to machine $j$ and the loads on the first $k$ machines are less than 1, $L_{k+1} = \ldots = L_{2k} = 2$ and $L_{2k+1} = 1$. One can easily see that every job $k+1, \ldots, 2k$ can move to machine $2k+1$ as $L_{2k+1}$ remains to be less than 2 and that every such jump induces jumps from the jobs $1, \ldots, k$. Hence, there are $\Omega(k^2) = \Omega(m^2)$ iterations. □

### 6.2 Makespan

In this section, we consider the Best Improvement pivot rule in the Makespan model. We use the fact that the potential $\phi_{\text{Makespan}}$ decreases by at least $2p_{\min} \cdot p_{\min}/s_{\max}$ if a sequence of jobs decrease their running time by a total of $p_{\min}/s_{\max}$ through jumping. This is due to a lemma by Even-Dar et al. [8] that if a jumping job $j$ improves its execution time by $\Delta$, then $\phi_{\text{Makespan}}$ drops exactly by $2p_j\Delta$.

Suppose that a job $j$ wants to jump away from machine $i$ to machine $i'$ and there is a smaller job $j'$ on machine $i$. At the current time, the costs of $j$ and $j'$ are the same as they are on the same machine. But the additional costs job $j'$ would generate on any machine are strictly smaller than the additional costs job $j$ would generate. Hence, job $j'$ would have smaller costs on machine $i'$ than job $j$. This leads to the following observation.

**Observation 1** *When a job jumps away from a machine $i$ according to the Best Improvement pivot rule, it was a smallest job on machine $i$.*

Let us now provide the main ideas of our proof. Imagine there are two jobs $j_1, j_2$ on the same machine $i$ and job $j_1$ jumps away in iteration $t_1$ making a small improvement directly before job $j_2$ leaves machine $i$ in iteration $t_2 = t_1 + 1$. Then job $j_1$ could improve its running time by $p_{j_2}/s_i$ by jumping back to machine $i$ in iteration $t_2 + 1$. If, however, $t_2 > t_1 + 1$, it could happen that another job $j_3$ from job $j_1$'s new machine leaves this machine leaving job $j_1$ unable to jump back. But then job $j_3$ is smaller than $j_1$ according to Observation 1 and thus could jump to machine $i$ in iteration $t_2 + 1$ unless it already made a big improvement or another job from job $j_3$'s new machine jumped away in the meantime etc. Lemma 4 proves that the potential drops significantly during such a sequence.

**Lemma 4.** *If two jobs jump away from a machine $i$ at iterations $t < t'$ and no job enters machine $i$ between $t$ and $t'$, then the potential $\phi_{Makespan}$ drops by at least $p_{\min}^2/s_{\max}$ during the iterations $t, \ldots, t' + 1$ when using the Best Improvement pivot rule.*

Imagine now there are two jobs $j_1, j_2$ entering the same machine $i$ in two consecutive iterations $t_1$ and $t_2 = t_1 + 1$, where job $j_1$ moves first. Then job $j_2$ would improve its running time by at least $p_{j_1}/s_i$ if it jumped in iteration $t_1$ as it also has the incentive to move to machine $i$ after job $j_1$'s jump. But if $t_2 > t_1 + 1$, it could be that in iteration $t_1$ job $j_2$'s running time is smaller than in iteration $t_2$ and in the meantime another job $j_3$ enters job $j_2$'s machine. If job $j_3$ is much larger than job $j_2$, then job $j_2$ would improve much by jumping to job $j_3$'s old machine. Otherwise, job $j_3$ could have moved to machine $i$ in iteration $t_1$ unless another job entered job $j_3$'s old machine in the meantime etc. Lemma 5 shows that also in this case the potential drops significantly.

**Lemma 5.** *If two jobs enter a machine $i$ at iterations $t' < t$ and no job leaves machine $i$ between $t'$ and $t$, then the potential $\phi_{Makespan}$ drops by at least $p_{\min}^2/(2 \cdot s_{\max})$ between $t'$ and $t + 1$ when using the Best Improvement pivot rule.*

Hence, we are able to show that if there is a machine to which two jobs migrate without a job leaving or from which two jobs leave without a job entering, the potential $\phi_{\text{Makespan}}$ drops significantly. The proof then concludes with the observation that this must happen every $O(m^2 n)$ iterations.

**Theorem 10.** *In the Makespan model, the convergence time of the Best Improvement pivot rule is in $O(m^2 n \cdot W^2/p_{\min}^2)$.*

*Proof.* According to Lemma 3, after $O(n)$ iterations the potential $\phi_{\text{Makespan}}$ is in $O(W^2/s_{\max})$. Hence, it suffices to show that in every sequence of $m^2 n$ consecutive iterations, $\phi_{\text{Makespan}}$ drops by at least $p_{\min}^2/(2s_{\max})$.

Let $S$ be a sequence of maximum length such that $\phi_{\text{Makespan}}$ drops by less than $p_{\min}^2/(2s_{\max})$, lasting from iteration $t_0$ to iteration $t_\ell$. We maintain a set of indices, which is empty at time $t_0$. When a job $j$ jumps from a machine $i_1$ to a machine $i_2$ at iteration $t \in \{t_0, \ldots, t_\ell\}$ and if there has not been a job that jumped to machine $i_1$ during the iterations $t_0, \ldots, t$, generate a new index which gets attached to machine $i_2$.

Otherwise, reattach the index previously attached to machine $i_1$ to machine $i_2$. Lemma 4 shows that this is well-defined as there cannot be another job leaving machine $i_1$ before another index gets attached to this machine.

At the end of the sequence, there can only be at most $m$ indices. If an index gets reattached from machine $i_1$ to machine $i_2$ at iteration $t$, then $L_{i_1}^t > L_{i_2}^{t+1}$, i.e., the running time of the machine an index is attached to is strictly monotonically decreasing.

Consider an index that jumps with job $j$ at iteration $t$ and with job $j'$ at iteration $t'$ to the same machine $i$. Let $j = j_1, j_2, \ldots, j_\ell$ be the jobs that entered machine $i$ and let $j'_1, \ldots, j'_\ell$ be the jobs that left machine $i$ during the iterations $t, \ldots, t'-1$ in this order. Lemma 4 and Lemma 5 show that the order in which this happened must be $j_1, j'_1, j_2, j'_2, \ldots, j_\ell, j'_\ell$ and that the sequences have the same length, i.e., the sequences are well-defined. As always only a smallest job on a machine is able to achieve the best improvement and for every $k$, job $j_k$ is on machine $i$ when job $j'_k$ leaves this machine, it must be the case that $L_i^t \leq L_i^{t'}$. But in the iterations $t+1$ and $t'+1$, the same index is attached to machine $i$, meaning that $L_i^t + p_j/s_i = L_i^{t+1} > L_i^{t'+1} = L_i^{t'} + p_{j'}/s_i$, i.e., $p_j > p_{j'}$. This means that an index cannot be attached twice to the same machine by a jump of the same job and thus an index gets reattached at most $n \cdot m$ times. This concludes the proof. □

### 6.3 SJF

**Theorem 11.** *In the SJF model, the convergence time of the Min Weight pivot rule is exactly $n$, even on two machines. The expected convergence time of the Random pivot rule is less than $n^2$.*

## 7 Price of Anarchy for FIFO

Brunsch et al. [4] already showed that the smoothed price of anarchy for near list schedules in the Makespan model, which correspond to local optima in the FIFO model, is $\Theta(\log \phi)$. We give matching bounds for the deterministic case.

**Theorem 12.** *In the FIFO model, the price of anarchy for local search is $\Theta(\log m)$ on related machines and $2 - 1/m$ on identical machines.*

## 8 Concluding Remarks

We have shown several bounds for the convergence times of local search regarding three different coordination mechanisms on rational inputs. The choice of the right pivot rule decides in the Shortest Job First model between linear and exponential convergence times. The FIFO model is new but we believe that it is a realistic choice for many different real-life applications. We were able to show that every pivot rule converges in this model in linear time on identical machines and a large class of reasonable pivot rules converges in smoothed polynomial time on related machines. An interesting observation is that the machine speeds do not occur in any bound. We leave it as a conjecture that

every pivot rule converges in polynomial time in the FIFO model. Another interesting open problem is whether the Best Improvement pivot rule in the Makespan model converges in smoothed or even deterministic polynomial time on related machines. We were only able to show that this happens with high probability when the input is perturbed.

*Acknowledgments.* We thank Clemens Rösner for helpful discussions about the lower bounds for the SJF model and the proof of Theorem 2.

## References

1. Aspnes, J., Azar, Y., Fiat, A., Plotkin, S., Waarts, O.: On-line routing of virtual circuits with applications to load balancing and machine scheduling. J. ACM 44(3), 486–504 (1997), http://dx.doi.org/10.1145/258128.258201
2. Beier, R., Vöcking, B.: Random knapsack in expected polynomial time. Journal of Computer and System Sciences 69(3), 306–329 (2004)
3. Brucker, P., Hurink, J., Werner, F.: Models and algorithms for planning and scheduling problems improving local search heuristics for some scheduling problems. part ii. Discrete Applied Mathematics 72(1), 47 – 69 (1997), http://www.sciencedirect.com/science/article/pii/S0166218X96000364
4. Brunsch, T., Röglin, H., Rutten, C., Vredeveld, T.: Smoothed performance guarantees for local search. Math. Program. 146(1-2, Ser. A), 185–218 (2014), http://dx.doi.org/10.1007/s10107-013-0683-7
5. Christodoulou, G., Koutsoupias, E., Nanavati, A.: Coordination mechanisms. In: Proc. of ICALP 2004, vol. 3142, pp. 345–357 (2004), http://dx.doi.org/10.1007/978-3-540-27836-8_31
6. Czumaj, A., Vöcking, B.: Tight bounds for worst-case equilibria. Transactions on Algorithms ACM 3(1) (2007)
7. Etscheid, M.: Performance guarantees for scheduling algorithms under perturbed machine speeds. In: Proc. of ISAAC 2013. pp. 207–217 (2013), http://dx.doi.org/10.1007/978-3-642-45030-3_20
8. Even-Dar, E., Kesselman, A., Mansour, Y.: Convergence time to nash equilibria. In: Proc. of ICALP 2003, pp. 502–513 (2003), http://dx.doi.org/10.1007/3-540-45061-0_41
9. Finn, G., Horowitz, E.: A linear time approximation algorithm for multiprocessor scheduling. BIT 19, 312–320 (1979)
10. Garey, M.R., Johnson, D.S.: Complexity results for multiprocessor scheduling under resource constraints. SIAM Journal on Computing 4, 397–411 (1975)
11. Graham, R.L.: Bounds for certain multiprocessing anomalies. Bell System Technical Journal, The 45(9), 1563–1581 (Nov 1966)
12. Hurkens, C.A., Vredeveld, T.: Local search for multiprocessor scheduling: how many moves does it take to a local optimum? Operations Research Letters 31(2), 137–141 (2003)
13. Immorlica, N., Li, L., Mirrokni, V.S., Schulz, A.S.: Coordination mechanisms for selfish scheduling. Theor. Comput. Sci. 410(17), 1589–1598 (2009), http://dx.doi.org/10.1016/j.tcs.2008.12.032
14. Lueker, G.S.: Exponentially small bounds on the expected optimum of the partition and subset sum problems. Rand. Struc. and Alg. 12(1), 51–62 (1998), http://dx.doi.org/10.1002/(SICI)1098-2418(199801)12:1<51::AID-RSA3>3.0.CO;2-S
15. Manthey, B., Röglin, H.: Smoothed analysis: Analysis of algorithms beyond worst case. it - Information Technology 53(6), 280–286 (2011)
16. Rosenthal, R.W.: A class of games possessing pure-strategy Nash equilibria. Internat. J. Game Theory 2, 65–67 (1973)
17. Schuurman, P., Vredeveld, T.: Performance guarantees of local search for multiprocessor scheduling. Informs Journal on Computing 19(1), 52–63 (2007)
18. Spielman, D., Teng, S.H.: Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. J. ACM 51(3), 385–463 (2004)

19. Spielman, D., Teng, S.H.: Smoothed analysis: An attempt to explain the behavior of algorithms in practice. Communications of the ACM 52(10), 76–84 (2009)

# A Deferred Proofs

## A.1 Proofs from Section 3

*Proof (Lemma 1).* Let $T$ be a random variable for the convergence time. Then $T$ is trivially bounded by $m^n$ as there are only $m^n$ different schedules and no schedule can appear twice in a sequence of monotonically improving steps. As the job sizes are each drawn from the interval $[0, 1]$ and thus the total weight $W$ is at most $n$, we can bound the expected value of $T$ in the following way.

$$\begin{aligned}\mathbf{E}[T] &= \int_0^\infty \mathbf{Pr}[T \geq \alpha]\,\mathrm{d}\alpha = \int_0^{m^n} \mathbf{Pr}[T \geq \alpha]\,\mathrm{d}\alpha \leq \int_0^{m^n} \mathbf{Pr}\left[f(m,n) \cdot \frac{n}{p_{\min}} \geq \alpha\right]\mathrm{d}\alpha \\ &= f(m,n) \cdot n \cdot \int_0^{\frac{m^n}{f(m,n)\cdot n}} \mathbf{Pr}[p_{\min} \leq 1/\alpha]\,\mathrm{d}\alpha \leq f(m,n) \cdot n \cdot \int_0^{m^n} \mathbf{Pr}[p_{\min} \leq 1/\alpha]\,\mathrm{d}\alpha.\end{aligned}$$

The random variable $p_{\min}$ is at least $1/\alpha$ exactly if all job sizes are at least $1/\alpha$. For every job size $p_j$ this happens independently of the other jobs sizes with probability at least $1 - \phi/\alpha$ as the density function of $p_j$ is bounded by $\phi$. Hence, with a Union Bound it follows

$$\mathbf{Pr}[p_{\min} \leq 1/\alpha] = \mathbf{Pr}[\exists j \colon p_j \leq 1/\alpha] \leq \sum_{j=1}^{n} \mathbf{Pr}[p_j \leq 1/\alpha] \leq \frac{n\phi}{\alpha},$$

and thus,

$$\int_0^{m^n} \mathbf{Pr}[p_{\min} \leq 1/\alpha]\,\mathrm{d}\alpha \leq 1 + \int_1^{m^n} \frac{n\phi}{\alpha}\mathrm{d}\alpha = 1 + n\phi \cdot \ln(m^n) = 1 + n^2\phi \cdot \ln m. \qquad \square$$

*Proof (Lemma 2).* Analogously to the proof of Lemma 1, we can bound the probability of $1/p_{\min}^2 \geq \alpha$ for some $\alpha$ using a Union Bound.

$$\mathbf{Pr}[1/p_{\min}^2 \geq \alpha] \leq \mathbf{Pr}[p_{\min} \leq 1/\sqrt{\alpha}] \leq \frac{n\phi}{\sqrt{\alpha}}.$$

This bound is at most $1/n$ for $\alpha \geq n^4\phi^2$. Theorem 10 shows that the Best Improvement pivot rule converges in the Makespan model in $O(m^2 n \cdot W^2/p_{\min}^2)$ steps. As $W^2 \leq n^2$, the probability that the Best Improvement pivot rule does not converge in $m^2n^7\phi^2$ steps is bounded from above by

$$\mathbf{Pr}[m^2n^3/p_{\min}^2 \geq m^2n^7\phi^2] = \mathbf{Pr}[1/p_{\min}^2 \geq n^4\phi^2] \leq \frac{1}{n}. \qquad \square$$

## A.2 Proofs from Section 4

*Proof (Theorem 1).* First consider an instance with $m = n$ machines and every job has size 1. In the initial schedule all jobs are assigned to the first machine. Then in every

step one job moves from the first machine to an empty machine. This happens $n-1$ times.

For the upper bound, first observe that the running time of the least loaded machine is monotonically increasing (this was also observed in [8]). As a job always strictly decreases its running time by moving and moves to a least loaded machine, each job can only move once. The first job on each machine does not move. Hence, the number of iterations is bounded from above by $n-1$. □

The proof idea for Theorem 3 is the following. We consider $2k+1$ jobs with job sizes $p_0 \ll p_1 < \ldots < p_{2k} < (1+1/k) \cdot p_1$, which are to be assigned to $m = 2$ identical machines. Initially, job 1 and all even-numbered jobs $j \geq 4$ are assigned to machine 1, whereas job 2 and all odd-numbered jobs $j \geq 3$ are assigned to machine 2. As the job sizes are all almost equal, only the number of jobs smaller than job $j$ on each machine and not their exact weights determine on which machine job $j$ wants to be. One can then show that in every second iteration either job $2k-1$ or job $2k$ jumps. Then by induction, the number of iterations grows exponentially in $k$.

One can use the same idea for the second part of the theorem to prove an expected superpolynomial convergence time when all job sizes are drawn uniformly at random from the interval $[0,1]$. Let $k = \Theta(\sqrt{n})$ and let $p_1 < \ldots < p_{2k}$ be the $2k$ largest job sizes. Then with constant probability $p_{2k} < (1+1/k) \cdot p_1$ and the other $n-2k$ smaller jobs can be distributed among the two machines in such a way that they generate nearly the same load on both machines and thus do not affect the jumps of the $2k$ largest jobs.

Now let us go into detail. Let $k \geq 2$. We construct a scheduling instance $I_k$ with $m = 2$ identical machines and $n = 2k+1$ jobs and an initial schedule for this instance from which the Max Weight pivot rule needs $2^{k-1}$ steps to a local optimum. The jobs $1, \ldots, 2k$ have processing requirements $p_1 < p_2 < \ldots < p_{2k-1} < p_{2k} < (1+1/k) \cdot p_1$. Additionally there is an extremely small job 0 with processing requirement $p_0 \leq (k+1) \cdot p_1 - k \cdot p_{2k} \leq p_1$. Note that $(k+1) \cdot p_1 - k \cdot p_{2k} = k \cdot ((1+1/k) \cdot p_1 - p_{2k})$ is strictly positive. The additional job 0 is not needed for the proof of the first part of Theorem 3. However, we will see later that its existence makes it possible to extend the theorem (with a slightly weaker bound for the convergence time) to $\phi$-perturbed scheduling instances.

**Lemma 6.** *Consider an arbitrary schedule $\sigma$ on instance $I_k$. For a job $j$ and a machine $i \in \{1,2\}$, let $J_i(j)$ denote the set of jobs on machine $i$ that have a smaller size than job $j$ ignoring job 0, i.e.,*

$$J_i(j) = \left\{ j' \in \{1, \ldots, j-1\} : \textit{job } j' \textit{ is assigned to machine } i \right\}.$$

- *If $|J_1(j)| < |J_2(j)|$, then job $j$ prefers to jump onto (stay on) machine 1.*
- *If $|J_1(j)| > |J_2(j)|$, then job $j$ prefers to jump onto (stay on) machine 2.*

Lemma 6 states that in the case $|J_1(j)| \neq |J_2(j)|$, the values $p_0, p_1, \ldots, p_{2k}$ are not of interest when we have to decide which of the machines 1 and 2 is favored by job $j$ in the current schedule. We only have to count on each machine the jobs whose indices are smaller than $j$, ignoring job 0. Only in the case $|J_1(j)| = |J_2(j)|$, the values $p_0, p_1, \ldots, p_{2k}$ matter.

*Proof (Lemma 6).* Due to symmetry, we only have to consider the first claim. For this, observe that $|J_1(j)| < k$ since $|J_1(j)| < |J_2(j)|$, $J_1(j) \cap J_2(j) = \emptyset$, and $J_1(j) \cup J_2(j) \subseteq \{1, \dots, 2k\}$. We obtain

$$\sum_{j' \in J_2(j)} p_{j'} - \sum_{j' \in J_1(j)} p_{j'} \geq |J_2(j)| \cdot p_1 - |J_1(j)| \cdot p_{2k} \geq p_1 + |J_1(j)| \cdot (p_1 - p_{2k})$$
$$> p_1 + k \cdot (p_1 - p_{2k}) = (k+1) \cdot p_1 - k \cdot p_{2k} \geq p_0 \,.$$

Hence, the total processing requirement of all jobs on machine 1 that are smaller than job $j$ is at most

$$p_0 + \sum_{j' \in J_1(j)} p_{j'} < \sum_{j' \in J_2(j)} p_{j'} \,.$$

The latter sum is a lower bound for the total processing requirement of all jobs on machine 2 that are smaller than job $j$. □

**Definition 2.** *Let $\sigma$ be an arbitrary schedule on instance $I_k$. By $\chi(\sigma) \in \mathbb{Z}$ we denote the difference between the number of jobs from $\{1, \dots, 2k\}$ on machine 1 and the number of jobs from $\{1, \dots, 2k\}$ on machine 2. We call schedule $\sigma$* balanced*, if $\chi(\sigma) = 0$. Otherwise, $\sigma$ is called* imbalanced*. In an imbalanced schedule, we call the machine with more jobs from $\{1, \dots, 2k\}$ the* critical machine*.*

Now consider the following schedule $\sigma_k$: Job 0 is assigned arbitrarily and job $j \in \{1, \dots, 2k\}$ is assigned to

$$\begin{cases} \text{machine 1} & \text{if } j = 1 \text{ or } (j \geq 3 \text{ and } j \text{ is even}) \,, \\ \text{machine 2} & \text{if } j = 2 \text{ or } (j \geq 3 \text{ and } j \text{ is odd}) \,. \end{cases}$$

In the remainder of this section, let $S$ denote the sequence of schedules that we obtain when we consider the Max Weight pivot rule, starting with schedule $\sigma_k$.

**Lemma 7.** *For any schedule $\sigma$ from the sequence $S$, $\chi(\sigma) \in \{-2, 0, 2\}$.*

*Proof.* We prove the claim by induction. For the initial schedule, the claim is true since $\chi(\sigma_k) = 0$. Now consider an arbitrary schedule $\sigma$ and its predecessor $\sigma'$ in $S$. For $\sigma'$, the induction hypothesis states that $\chi(\sigma') \in \{-2, 0, 2\}$. If $\chi(\sigma') = 0$, then $|\chi(\sigma)| = 2$ since the value $\chi$ changes by 2 in each iteration. Let us consider the case $|\chi(\sigma')| = 2$. We show that the largest job $j$ on the critical machine $i$ of $\sigma'$ will jump in the next iteration yielding $\chi(\sigma) = 0$. Due to $|\chi(\sigma')| = 2$, we obtain $|J_i(j)| = k$ and $|J_{i'}(j)| \leq k - 1$ in the schedule $\sigma'$, where $i' = 3 - i$ denotes the other machine. Hence, in accordance with Lemma 6, job $j$ desires to jump. Due to the Max Weight pivot rule, it prevents all jobs $j' < j$ from jumping. On the other hand, all other jobs $j' > j$ must be on machine $i'$ due to the choice of $j$. As $|J_i(j')| = |J_i(j)| + 1 = k + 1 > |J_{i'}(j')|$ for the schedule $\sigma'$, these jobs do not desire to jump. This again follows from Lemma 6. Consequently, job $j$ will jump from machine $i$ to machine $i'$ in the next iteration. This concludes the proof. □

**Lemma 8.** *Let $\sigma$ be an arbitrary imbalanced schedule from the sequence $S$. Then at least one of the two largest jobs $2k-1$ or $2k$ is assigned to the critical machine.*

*Proof.* Assume, to the contrary, that there is an imbalanced schedule $\sigma$ in $S$ that assigns both jobs $2k-1$ and $2k$ to the non-critical machine. We consider the first such schedule in the sequence and let $i$ and $i'$ denote the critical and the non-critical machine, respectively. Since the first schedule $\sigma_k$ in the sequence $S$ is balanced and, thus, is not equal to $\sigma$, schedule $\sigma$ must have a predecessor $\sigma'$ in the sequence $S$. Due to Lemma 7 and the observation, that the value $\chi$ changes by 2 in each iteration, schedule $\sigma'$ must be balanced. Furthermore, since machine $i$ becomes the critical machine after the following iteration, a job must jump from machine $i'$ to machine $i$. Consequently, as both, job $2k-1$ and job $2k$, are not assigned to machine $i$ in schedule $\sigma$, they must be assigned to machine $i'$ in schedule $\sigma'$. Now let us consider the largest job $j$ that is assigned to machine $i$ in schedule $\sigma'$. We will show that this job is the next to jump, contradicting the fact that the next jump is from machine $i'$ to machine $i$.

As all jobs $j'$ that are larger than $j$ (including job $2k-1$ and $2k$) are assigned to machine $i'$, none of these can improve by jumping to machine $i$ as $|J_i(j')| = k > |J_{i'}(j')|$ because schedule $\sigma'$ is balanced and $j$ is the largest job on machine $i$. On the other hand, we know that $|J_{i'}(j)| \leq |J_{i'}(2k-1)| = k-2 < |J_i(j)|$ since both jobs $2k-1$ and $2k$ are assigned to machine $i'$. This implies that job $j$ will be the next job to move. □

**Corollary 2.** *In the sequence $S$, the number of jumps involving one of the jobs $2k-1$ or $2k$ is at least half the number of all jumps.*

*Proof.* Consider an arbitrary imbalanced schedule $\sigma$ from the sequence $S$. According to Lemma 8, at least one job from $\{2k-1, 2k\}$ is assigned to the critical machine $i$. Let $j \geq 2k-1$ be the largest job on the critical machine. Then, $|J_i(j)| = k > |J_{2-i}(j)|$, i.e., job $j$ desires to change its machine. Hence, the next jump will be performed by job $2k-1$ or job $2k$.

Since every second schedule from the sequence $S$ is imbalanced and the last schedule of $S$ cannot be imbalanced due to the previous argument, at least half of the jumps involves job $2k-1$ or job $2k$. □

The first part of Theorem 3 follows directly from the following lemma.

**Lemma 9.** *The number of iterations in the sequence $S$ is at least $2^{k-1}$.*

*Proof.* We prove the lemma by induction on $k$. For the base case $k=2$, job 1 and job 4 are assigned to machine 1 and job 2 and job 3 are assigned to machine 2 in schedule $\sigma_2$. Hence, job 3 will jump to machine 1 in the next iteration because $p_1 < p_2$. Afterwards, job 4 will jump to machine 2. Hence, the number of jumps is at least $2 = 2^{2-1}$.

For the inductive step let us consider an arbitrary value $k \geq 3$. First of all observe that

$$p_1 < p_2 < \ldots < p_{2k} < (1 + 1/k) \cdot p_1 < (1 + 1/(k-1)) \cdot p_1$$

and

$$\begin{aligned} p_0 &\leq (k+1) \cdot p_1 - k \cdot p_{2k} \\ &< ((k-1)+1) \cdot p_1 - (k-1) \cdot p_{2k} \\ &< ((k-1)+1) \cdot p_1 - (k-1) \cdot p_{2(k-1)} \,. \end{aligned}$$

Particularly, the jobs $0, 1, \ldots, 2(k-1)$, together with the machines 1 and 2, form a scheduling instance $I_{k-1}$ with the properties required to apply the inductive hypothesis. Moreover, when we remove the jobs $2k-1$ and $2k$ from schedule $\sigma_k$, then we obtain $\sigma_{k-1}$ for which we can apply the inductive hypothesis. We classify the iterations that we obtain starting from schedule $\sigma_k$ as follows: if job $2k-1$ or job $2k$ changes the machine, then we call this iteration a Type 2 iteration. Otherwise it is a Type 1 iteration. Observe that the subsequence of all Type 1 iterations is exactly the sequence of iterations that we get when we start with schedule $\sigma_{k-1}$. This is due to the fact that the behavior of the jobs $0, 1, \ldots, 2(k-1)$ is not affected by the larger jobs $2k-1$ and $2k$. Hence, the number of Type 1 iterations is at least $2^{k-2}$ in accordance with the inductive hypothesis. Finally, Corollary 2 states that there are at least as many Type 2 iterations as Type 1 iterations, yielding the lower bound of $2^{k-1}$ for the total number of iterations. □

*Proof (Theorem 3).* Only the second part of the theorem is left to show. Let $n$ denote the number of jobs and let $r_1, \ldots, r_n$ denote the random processing requirements. For $\phi = 1$, every processing requirement $r_i$ is chosen uniformly at random from $[0, 1]$. Let $x = \frac{1}{3\sqrt{n}} \leq \frac{1}{3}$ and $k = \lfloor \frac{x}{4} \cdot n \rfloor = \lfloor \frac{\sqrt{n}}{12} \rfloor$. We show that the convergence time is $2^{\Omega(\sqrt{n})}$, unless at least one of two failure events $\mathcal{F}_1$ or $\mathcal{F}_2$ occurs.

We define $\mathcal{F}_1$ to be the event that fewer than $2k$ values $r_i$ lie in the interval $[1-x, 1]$ or that fewer than $n/3$ values $r_i$ lie in the interval $[0, 1-x]$. In expectation, $nx = \frac{\sqrt{n}}{3}$ values $r_i$ lie in the interval $[1-x, 1]$. Due to the Chernoff bound, the probability that fewer than $2k \leq \frac{nx}{2}$ values $r_i$ lie in the interval $[1-x, 1]$, is bounded from above by $\exp(-xn/8) = \exp(-\sqrt{n}/24)$, which tends to 0 for $n \to \infty$. Additionally, the probability that at least $n \cdot 2x \leq 2n/3$ values $r_i$ fall into the interval $[1-x, 1]$ is at most $1/2$ due to Markov's inequality. Hence, with constant probability the failure event $\mathcal{F}_1$ does not occur. From now on assume that this is the case. Then at least $2k$ values lie in $[1-x, 1]$ and at least $n/3$ values lie in $[0, 1-x]$.

By $p_1 < \ldots < p_{2k}$ we denote the $2k$ largest values $r_i$. Observe that, under the assumption $\neg\mathcal{F}_1$, $p_1 \geq 1 - x$ and hence

$$\begin{aligned} \bar{p_0} &:= (k+1) \cdot p_1 - k \cdot p_{2k} \\ &= p_1 - k \cdot (p_{2k} - p_1) \\ &\geq 1 - x - k \cdot x \\ &= 1 - (k+1) \cdot x \end{aligned}$$

and

$$(k+1) \cdot x = kx + 2x - x \leq \frac{x^2 n}{4} + 2x - x \leq \frac{1}{36} + \frac{2}{3} - x \leq 1 - x \,.$$

Consequently, $\bar{p_0} \geq x > 0$ and

$$p_{2k} = \frac{1}{k} \cdot ((k+1) \cdot p_1 - \bar{p_0}) < \left(1 + \frac{1}{k}\right) \cdot p_1 \,.$$

Hence, the jobs with processing requirements $p_1, \ldots, p_{2k}$ together with the two machines form an instance $I_k$ as used in the proof of the first part of Theorem 3 (except for the missing job 0).

This alone does not suffice to prove the theorem because we cannot simply eliminate the other jobs from the instance. Instead we will distribute them onto the two machines in such a way that their total contributions to the loads of the two machines are almost the same. Let $R_1 \subseteq \{i \mid r_i \in [1-x, p_1)\}$ denote the remaining values from $[1-x, 1]$, let $R_2 \subseteq \{i \mid r_i \in [0, 1-x)\}$, and let $R = R_1 \cup R_2$. We look for a subset $R' \subseteq R$ such that the gap $\Delta(R') := \left|\sum_{i \in R'} r_i - \sum_{i \in R \setminus R'} r_i\right|$ is small. For this, we first choose a subset $R'_1 \subseteq R_1$ such that $\alpha(R'_1) := \sum_{i \in R_1 \setminus R'_1} r_i - \sum_{i \in R'_1} r_i \in [-1, 1]$. Such a subset must always exist and it can be constructed by greedily assigning the jobs from $R_1$ one after another to the machine with lower load.

Now we use the principle of deferred decisions and assume that the index set $R_2$ is already fixed. Note that the processing requirements of the jobs in $R_2$ have not been chosen yet. Then each value $r_i$ with $i \in R_2$ is uniformly distributed on $[0, 1-x]$. Since we assume that the failure event $\mathcal{F}_1$ does not occur, we have $|R_2| \geq n/3$. Our goal is now to find a subset $R'_2 \subseteq R_2$ such that the gap $\beta(R'_2) := \sum_{i \in R'_2} r_i - \sum_{i \in R_2 \setminus R'_2} r_i$ is close to $\alpha(R'_1)$ because $\Delta(R'_1 \cup R'_2) = \beta(R'_2) - \alpha(R'_1)$. Lueker [14] studied the *partition gap* of a set of random numbers. Adapted to our notation, he proved that with probability exponentially close to 1, for every $\alpha \in [-1, 1]$ there exists a subset $R'_2 \subseteq R_2$ such that $\beta(R'_2)$ is exponentially close to $\alpha$. In particular, the probability that there does not exist a subset $R'_2$ for which $|\beta(R'_2) - \alpha(R'_1)| \leq \bar{p_0}$ goes to 0 for $n \to \infty$. Failure event $\mathcal{F}_2$ is defined to be the event that no such subset $R'_2$ exists.

If neither $\mathcal{F}_1$ nor $\mathcal{F}_2$ occurs, we consider the scheduling instance with jobs $1, \ldots, 2k$ with sizes $p_1, \ldots, p_{2k}$ and jobs $2k+1, \ldots, n$ with sizes corresponding to the remaining, smaller values $r_i$. The jobs $2k+1, \ldots, n$ are assigned to the machines 1 and 2 as induced by the aforementioned partition, ensuring that the jobs from the larger class are assigned to machine 1. The jobs $1, \ldots, 2k$ are now assigned to the machines 1 and 2 according to schedule $\sigma_k$. As long as one of the jobs $1, \ldots, 2k$ can jump, none of the smaller jobs $2k+1, \ldots, n$ will move due to the Max Weight pivot rule. Hence, as long as the jobs $1, \ldots, 2k$ move, they behave exactly the same as in schedule $\sigma_k$ where machine 1 is assigned jobs from $\{1, \ldots, 2k\}$ and the additional job $p_0$ and machine 2 is only assigned jobs from $\{1, \ldots, 2k\}$. By Lemma 9, this takes at least

$$2^{k-1} = 2^{\lfloor nx/4 \rfloor - 1} = 2^{\lfloor \sqrt{n}/12 \rfloor - 1}$$

iterations. This proves the theorem because with constant probability neither $\mathcal{F}_1$ nor $\mathcal{F}_2$ occurs. □

### A.3 Proofs from Section 5

*Proof (Theorem 4).* For the lower bound on two machines, assign all jobs to a machine of speed $1/(2n)$ and set the speed of the other machine to 1. Then all jobs jump to the faster machine.

For the upper bound, we show that every job jumps at most once. Assume to the contrary that there is a job $j$ that jumps twice and let $t$ be the iteration in which job $j$ is the first time able to move again after its first jump. Then in the previous iteration, a job $j' \neq j$ jumped from a machine $i_1$ to a machine $i_2$ and job $j$ is now able to jump to machine $i_1$ as the load of no other machine decreased during the last iteration. Job $j$ cannot be on machine $i_2$ because it has the same size as job $j'$ and thus we would end up in the same potential as in iteration $t-1$ if job $j$ jumped from machine $i_2$ to machine $i_1$. But then job $j$ could also jump to machine $i_2$ in the previous iteration because

$$c_j^{t-1} = c_j^t > L_{i_1}^t + \frac{1}{s_{i_1}} = L_{i_1}^{t-1} > L_{i_2}^{t-1} + \frac{1}{s_{i_2}}.$$

This contradicts the choice of $t$. □

### A.4 Proofs from Section 6

**Definition 3.** *Let $T := \frac{W}{s_{\max}}$. For any given schedule, we categorize the machines in the following way:*

- *A machine $i$ is a* Type 2 *machine if $L_i > 2T$.*
- *A machine $i$ is a* Type 1 *machine if $2T \geq L_i > T$.*
- *A machine $i$ is a* Type 0 *machine if $T \geq L_i$.*

*Proof (Lemma 3).* Consider a job $j$ on a machine $i$ that can improve by jumping onto machine $i'$.

If $i$ is a Type 0 machine, then the costs of $j$ are at most $T$ and thus $j$ can only improve by at most $T$. The costs of the top-most job on any Type 2 machine are greater than $2T$. Therefore, such a job can improve by strictly more than $T$ by jumping onto the fastest machine, where its new costs would be at most $W/s_{\max} = T$.

Second, $i'$ must be a type 0 machine as the new costs of $j$ are at most its costs if it would jump to the fastest machine, which is at most $W/s_{\max} = T$. This proves the first part of the lemma.

If there are no Type 2 machines, then $2T$ is an upper bound for the costs of any job and thus also for the makespan of the current schedule. Then $\phi_{\text{FIFO}} \leq n \cdot 2T = O(n \cdot \frac{W}{s_{\max}})$. The Makespan potential for a schedule $\sigma$ can be bounded in the following way:

$$\begin{aligned}\phi_{\text{Makespan}} &= \sum_{i=1}^{m} \frac{1}{s_i} \cdot \left( \left( \sum_{j \in \sigma^{-1}(i)} p_j \right)^2 + \sum_{j \in \sigma^{-1}(i)} p_j^2 \right) \leq \sum_{i=1}^{m} \frac{2}{s_i} \cdot \left( \sum_{j \in \sigma^{-1}(i)} p_j \right)^2 \\ &= \sum_{i=1}^{m} \left( 2L_i \cdot \sum_{j \in \sigma^{-1}(i)} p_j \right) \leq 4T \cdot \sum_{i=1}^{m} \sum_{j \in \sigma^{-1}(i)} p_j = O(T \cdot W) = O\left( \frac{W^2}{s_{\max}} \right).\end{aligned}$$

□

### A.5 Proofs from Section 6.1

*Proof (Theorem 5).* The lower bound of $n$ for every pivot rule follows directly from Theorem 4.

For the lower bound of $2n-2$ for some pivot rules, consider two machines with speeds $s_1 = 2$ and $s_2 = 1$ and $n$ jobs with sizes $p_1 = 2, p_2 = \ldots = p_{n-1} = 1/(2n)$, and $p_n = 1$. All jobs are assigned to the slower machine 2 and their permutation $\pi$ is the identity, i.e., job 1 is the first job to be processed and job $n$ is on top.

Let the jobs $n, \ldots, 1$ jump from machine 2 to machine 1 in this order. This is valid as $(\sum_j p_j)/s_1 < 2 = p_1/s_2$. Then let the jobs $2, \ldots, n-1$ jump back to machine 2. This is valid as $(n-2)/(2n) < 1/2 = p_n/s_1$ and $n$ is the first job on machine 1.

For the upper bound, consider the setting of two machines with $s_1 \geq s_2$. We claim that after a job jumped from the faster machine 1 to the slower machine 2, no job can jump in the other direction from machine 2 to machine 1. Then there are at most $n$ jumps from machine 2 to machine 1 and at most $n-2$ jumps from machine 1 to machine 2 as the first job on the faster machine 1 does never jump to machine 2 and the last job leaving machine 2 does also not jump back.

In order to show the claim, let a job $j$ jump from machine 1 to machine 2 in iteration $t$ and let $j'$ be another job on machine 2. Then,

$$c_{j'}^{t+1} = c_{j'}^{t} \leq L_2^t < L_1^t - \frac{p_j}{s_2} = L_1^{t+1} + \frac{p_j}{s_1} - \frac{p_j}{s_2} \leq L_1^{t+1} \leq L_1^{t+1} + \frac{p_{j'}}{s_1},$$

i.e., job $j'$ cannot improve its costs by jumping. The strict inequality comes from the fact that job $j$ improved its costs by jumping in iteration $t$. □

*Proof (Theorem 7).* Due to Corollary 1, after an expected number of $O(n \log n)$ iterations, the potential $\phi_{\text{FIFO}}$ is bounded by $O(nW/s_{\max})$. As shown in the proof of Theorem 6, after at most $m(m-1)$ iterations there is a job $j$ such that $\phi_{\text{FIFO}}$ decreases by a total of at least $p_{\min}/s_{\max}$ in this sequence if job $j$ jumps next. Hence, after an expected number of $O(n)$ such sequences the potential $\phi_{\text{FIFO}}$ drops by this amount. □

### A.6 Proofs from Section 6.2

**Definition 4.** *Let $(j_1, \ldots, j_\ell), (i_0, \ldots, i_\ell)$, and $t_1 < \ldots < t_\ell$ be sequences such that for any $k = 1, \ldots, \ell$, job $j_k$ jumps from machine $i_{k-1}$ to machine $i_k$ in iteration $t_k$. These sequences are called*

1. forward thread *if from iteration $t_k + 1$ to iteration $t_{k+1} - 1$, no job leaves machine $i_k$ for every $k = 1, \ldots, \ell - 1$.*
2. backward thread *if from iteration $t_k + 1$ to iteration $t_{k+1} - 1$, no job enters machine $i_k$ for every $k = 1, \ldots, \ell - 1$.*

Intuitively, a forward thread always follows the first job leaving a machine and a backward thread backtracks the last job entering a machine in the past.

**Lemma 10.** *Consider a forward thread $((j_1,\dots,j_\ell),(i_0,\dots,i_\ell),t_1<\dots<t_\ell)$. Then $p_{j_1}\geq\dots\geq p_{j_\ell}$ and the potential $\phi_{Makespan}$ decreases in total by at least $2(L_{i_0}^{t_1}-L_{i_\ell}^{t_\ell+1})\cdot p_{\min}$ in the iterations $t_1,\dots,t_\ell$.*

*Proof.* The job sizes are monotonically decreasing because at iteration $t_k$ for $2\leq k\leq\ell$, job $j_{k-1}$ is also on machine $i_{k-1}$ when job $j_k$ leaves this machine and only the smallest job on a machine is able to leave according to Observation 1.

The improvement of the jumping job in iteration $t_k$ is given by $L_{i_{k-1}}^{t_k}-L_{i_k}^{t_k+1}$ leading to a potential drop of at least $2(L_{i_{k-1}}^{t_k}-L_{i_k}^{t_k+1})\cdot p_{\min}$. As no job leaves machine $i_k$ between the iterations $t_k+1$ and $t_{k+1}-1$, it holds $L_{i_k}^{t_k+1}\leq L_{i_k}^{t_{k+1}}$. By summing over all possible choices for $k$, we attain the desired bound. □

*Proof (Lemma 4).* Let job $j$ jump away from machine $i$ at iteration $t$ and let job $j'$ jump away from machine $i$ at iteration $t'$ such that no job enters machine $i$ in the meantime. W.l.o.g., also no job left machine $i$ in the meantime. Let $(j=j_1,\dots,j_\ell),(i=i_0,\dots,i_\ell)$, and $t=t_1<\dots<t_\ell$ be the maximum forward thread starting in iteration $t$ with $t_\ell<t'$. If $L_{i_\ell}^{t_\ell+1}\leq L_i^t-p_{\min}/(2s_{\max})$, the potential drop follows from Lemma 10. Otherwise, the potential drop gained by letting job $j_\ell$ jump from machine $i_\ell$ to machine $i$ at iteration $t'+1$ is at least

$$\begin{aligned}&L_{i_\ell}^{t'+1}-\left(L_i^{t'+1}+\frac{p_{j_\ell}}{s_i}\right)\geq L_{i_\ell}^{t_\ell+1}-\left(L_i^{t'+1}+\frac{p_{j_\ell}}{s_i}\right)>L_i^t-L_i^{t'+1}-\frac{p_{j_\ell}}{s_i}-\frac{p_{\min}}{2s_{\max}}\\ =&\frac{p_j+p_{j'}-p_{j_\ell}}{s_i}-\frac{p_{\min}}{2s_{\max}},\end{aligned}$$

where the first inequality stems from the maximality of the chosen thread, i.e., no job left machine $i_\ell$ in the meantime. Again by Lemma 10, it holds $p_j\geq p_{j_\ell}$ and thus the improvement for job $j_\ell$ would then be at least $p_{\min}/(2s_{\max})$. The Best Improvement pivot rule chooses a job in iteration $t'+1$ that gains at least that much leading to a potential drop of at least $p_{\min}^2/s_{\max}$. □

**Lemma 11.** *Consider a backward thread $((j_1,\dots,j_\ell),(i_0,\dots,i_\ell),t_1<\dots<t_\ell)$. Then the potential $\phi_{Makespan}$ decreases in total by at least $(p_{j_1}-p_{j_\ell})\cdot p_{\min}/s_{\max}$ in the iterations $t_1,\dots,t_\ell+1$ when using the Best Improvement pivot rule.*

*Proof.* Let $1\leq k\leq\ell$ and consider the iterations $t_k$ and $t_k+1$. If job $j_{k+1}$ (which could also be equal to job $j_k$) jumped from machine $i_k$ to machine $i_{k-1}$ in iteration $t_k+1$, then the total improvement for the two jumping jobs would be

$$\begin{aligned}&L_{i_{k-1}}^{t_k}-\left(L_{i_k}^{t_k}+\frac{p_{j_k}}{s_{i_k}}\right)+L_{i_k}^{t_k+1}-\left(L_{i_{k-1}}^{t_k+1}+\frac{p_{j_{k+1}}}{s_{i_{k-1}}}\right)\\ =&L_{i_{k-1}}^{t_k}-\left(L_{i_{k-1}}^{t_k+1}+\frac{p_{j_{k+1}}}{s_{i_{k-1}}}\right)=\frac{p_{j_k}-p_{j_{k+1}}}{s_{i_{k-1}}},\end{aligned}$$

where we used that $L_{i_k}^{t_k}+\frac{p_{j_k}}{s_{i_k}}=L_{i_k}^{t_k+1}$ and $L_{i_{k-1}}^{t_k+1}=L_{i_{k-1}}^{t_k}-\frac{p_{j_k}}{s_{i_{k-1}}}$. This leads to a potential drop of $\phi_{\text{Makespan}}$ by at least $2(p_{j_k}-p_{j_{k+1}})\cdot p_{\min}/s_{\max}$. We do not know whether this

term is positive and hence whether this move by job $j_{k+1}$ is legal, but we know that the job that jumps in iteration $t_k + 1$ improves by at least as much as job $j_{k+1}$ would improve by jumping to machine $i_{k-1}$. By summing over all possible choices for $k$, we count every iteration at most twice leading to the lower bound of $(p_{j_1} - p_{j_\ell}) \cdot p_{\min}/s_{\max}$ for the total decrease of $\phi_{\text{Makespan}}$. □

*Proof (Lemma 5).* Let job $j'$ enter machine $i$ at iteration $t'$ and let job $j$ enter machine $i$ at iteration $t$ such that no job leaves machine $i$ in the meantime. W.l.o.g., also no job enters machine $i$ in the meantime. Let $(j_1, \ldots, j_\ell = j), (i_0, \ldots, i_\ell = i)$, and $t_1 < \ldots < t_\ell = t$ be the maximum backward thread ending in iteration $t$ with $t_1 > t'$. As job $j_1$ jumps from machine $i_0$ to machine $i_1$ in iteration $t_1$ and no other job enters machine $i_1$ from then on until iteration $t_2$, it holds $L_{i_0}^{t_1} > L_{i_1}^{t_1} + \frac{p_{j_1}}{s_{i_1}} \geq L_{i_1}^{t_2}$. Reiterating this argument yields $L_{i_0}^{t_1} > L_i^t + \frac{p_j}{s_i}$. As the chosen sequence is maximal, it also holds $L_{i_0}^{t'} \geq L_{i_0}^{t_1} > L_i^t + \frac{p_j}{s_i} = L_i^{t'} + \frac{p_{j'}+p_j}{s_i}$ and job $j_1$ is on machine $i_0$ during iteration $t'$. If $p_{j_1} > p_j + p_{\min}/2$, the potential drop follows from Lemma 11. Otherwise,

$$L_{i_0}^{t'} \geq L_i^{t'} + \frac{p_{j'} + p_{j_1} - p_{\min}/2}{s_i} \geq L_i^{t'} + \frac{p_{j_1}}{s_i} + \frac{p_{\min}}{2s_{\max}},$$

i.e., job $j_1$ would improve by at least $p_{\min}/(2s_{\max})$ by jumping from machine $i_0$ to machine $i$ in iteration $t'$. Hence, the job $j'$ moving in iteration $t'$ must decrease its running time by at least as much yielding a potential drop of at least $p_{\min}^2/s_{\max}$. □

### A.7 Proofs from Section 6.3

*Proof (Theorem 11).* W.l.o.g., $p_1 \leq \ldots \leq p_n$. When the Min Weight pivot rule selects a job $j$ to jump, all jobs $1, \ldots, j-1$ do not want to jump. As larger jobs do not affect the decision whether a job wants to jump, the jobs $1, \ldots, j-1$ do not want to jump to job $j$'s old machine afterwards and thus, the jobs $1, \ldots, j$ will never jump again. Hence, every job jumps at most once and the convergence time is at most $n$. On the other hand, every job jumps if all jobs are initially assigned to an arbitrarily slow machine.

For the analysis of the Random pivot rule, let $j$ be the smallest unsatisfied job. It takes an expected number of $O(n)$ iterations until job $j$ gets selected to jump. Afterwards, it will not jump again. By linearity of expectation, the convergence time for the Random pivot rule is then in $O(n^2)$. □

### A.8 Proofs from Section 7

*Proof (Theorem 12).* It is easy to see that every list schedule is a local optimum w.r.t. the FIFO model. Hence, the lower bounds $\Omega(\log m)$ by Aspnes et al. [1] for related machines and $2 - 1/m$ by Graham [11] for identical machines translate to lower bounds for local search in the FIFO model. Graham's proof for the upper bound of $2 - 1/m$ can be applied to the FIFO model without any changes.

For the upper bound for related machines, we use the notation of *near list schedules* defined in the Makespan model by Brunsch et al. [4]:

**Definition 5.** *We call a schedule $\sigma$ on machines $1, \ldots, m$ with speeds $s_1, \ldots, s_m$ a* near list schedule*, if we can index the jobs in such a way that*

$$L_{i'} + \frac{p_j}{s_{i'}} \geq L_i - \sum_{\ell \in \sigma^{-1}(i):\, \ell < j} \frac{p_\ell}{s_i} \tag{1}$$

*for all machines $i' \neq i$ and all jobs $j \in J_i(\sigma)$.*

This definition is equivalent to the definition of a locally optimal schedule w.r.t the FIFO model if one inverses the order of the jobs. Hence, in order to bound the price of anarchy of local search in the FIFO model, we can use their results for near list schedules in the Makespan model.

W.l.o.g., $s_1 \geq \ldots \geq s_m$ and let the makespan of an optimal schedule be exactly 1. Let $\sigma$ be a near list schedule with a makespan of strictly less than $c + 2$ for some integer $c$, and let $i^*$ be the fastest machine with $L^\sigma_{i^*} < 2$. Brunsch et al. [4] showed that the total processing requirement on the machines $i^*, \ldots, m$ in any optimal schedule is at least $(c-1) \cdot s_1$ (cf. Lemma 9 and Lemma 13 with $k = t = 2$). On the other hand, they showed that $s_1 \geq 2^{\lfloor (c-1)/6 \rfloor} \cdot s_{i^*}$ (cf. Lemma 15 with $i_1 = 1$ and $i_2 = i^*$). Since in any optimal schedule, the running times of the machines $i^*, \ldots, m$ are at most 1 and the machine speeds are monotonically decreasing, it holds

$$m \geq m - i^* + 1 \geq (c-1) \cdot 2^{\lfloor (c-1)/6 \rfloor},$$

i.e., $c = O(\log m)$ and thus the price of anarchy for FIFO is in $O(\log m)$. □