Jena Research Papers in Business and Economics

Comparing the Minimum Completion Times of Two Longest-First Scheduling-Heuristics

Rico Walter

13/2010

Jenaer Schriften zur Wirtschaftswissenschaft

Working and Discussion Paper Series
School of Economics and Business Administration
Friedrich-Schiller-University Jena

ISSN 1864-3108

Publisher:

Wirtschaftswissenschaftliche Fakultät
Friedrich-Schiller-Universität Jena
Carl-Zeiß-Str. 3, D-07743 Jena

www.jbe.uni-jena.de

Editor:

Prof. Dr. Hans-Walter Lorenz
h.w.lorenz@wiwi.uni-jena.de

Prof. Dr. Armin Scholl
armin.scholl@wiwi.uni-jena.de

www.jbe.uni-jena.de

Comparing the Minimum Completion Times of Two Longest-First Scheduling-Heuristics

Rico Walter

Friedrich-Schiller-Universität Jena, Fakultät für Wirtschaftswissenschaften,

Lehrstuhl für ABWL/Management Science, Carl-Zeiß-Straße 3, D-07743 Jena, Germany

rico.walter@uni-jena.de

Abstract

For the problem of non-preemptively scheduling n independent jobs on m identical parallel machines so that the minimum (or earliest) machine completion time is maximized, we compare two well-known longest-first heuristics - the LPT- (longest processing time) and the RLPT-heuristic (restricted LPT).

We prove that the minimum completion time of the LPT-schedule is at least as long as the minimum completion time of the RLPT-schedule. Furthermore, we show that the minimum completion time of the RLPT-heuristic always remains within a factor of $1/m$ of the optimal minimum completion time. The paper finishes with a conjecture on the probabilistic behavior of the RLPT-heuristic compared to the LPT-heuristic in case of two machines.

Keywords: Scheduling, Heuristics, Minimum completion time, Worst-case analysis

1 Introduction

1.1 Problem description and notation

We consider $m \geq 2$ identical parallel machines and a set $\mathcal{J} = \{J_1, \ldots, J_n\}$ of n independent jobs, i.e., no precedence constraints exist between any two jobs. Each job has to be processed without interruption by exactly one machine; regardless which one. Job J_j has a non-negative processing time (or length) t_j which does not depend on the machine by which the job is processed. We assume the jobs to be labeled so that $t_1 \geq \ldots \geq t_n \geq 0$. Furthermore, without loss of generality we assume n to be a multiple of m, and we assume $t_{n-m+1} > 0$. This brings some technical benefits for the theoretical analyses presented in the Sections 2 and 3. For economy of notation, we usually omit jobs of length 0 in the examples, and we often identify the jobs by their index.

The goal of the scheduling scenario is to assign the jobs to the machines so that the minimum completion time of the machines is maximized (without introducing idle times). A feasible assignment is called schedule. In other words, the goal is to partition the set of jobs into (at most) m subsets so that the smallest subset sum is maximized. A subset sum is simply the sum of the job processing times in the subset and corresponds to a machine completion time. We let C_i denote the completion time of machine i. The minimum completion time is denoted by $C_{min} = \min_{i=1,\ldots,m}\{C_1, \ldots, C_m\}$, the maximum completion time (also known as makespan) is denoted by C_{max}, and we denote the difference between the maximum and the minimum completion time by C_Δ. Thereby, the superscript * refers to an optimal schedule while expressions with superscript ^{H} refer to a schedule generated by a heuristic H.

Concerning the analysis presented in the following sections it is useful to divide $\mathcal{J}$ into n/m **ranks**, with jobs $J_{rm+1}, \ldots, J_{rm+m}$ in rank $r+1$, $r = 0, 1, \ldots, n/m-1$.

1.2 Related objective functions

The problem of maximizing the minimum completion time belongs to the class of covering problems as the jobs should "cover" the longest possible time interval that is common to all machines.

It has applications in the sequencing of maintenance actions for modular gas turbine aircraft engines, see [6]. In some sense it is dual - but in general not equivalent - to the well-known problem of minimizing the makespan which belongs to the class of packing problems. Here, the jobs should be "packed" into the smallest possible time interval on all machines. Both objective functions indirectly aim at practice-oriented balanced schedules. While the C_{max}-criterion attempts to level the total workload by concentrating on the longest running machine(s), with the C_{min}-criterion the key focus is on the shortest running machine(s). Balanced solutions are often sought if the machines are operated by workers among which the total workload should be distributed almost equally or if the machines should be utilized almost equally. In this context, the problem of minimizing C_Δ comprises both C_{min}-maximization and C_{max}-minimization and it even seems to be a more direct measure of "near-equality" [2]. However, we concentrate our investigations on the C_{min}-maximization problem and extend our main result (see Theorem 2.1) to the problem of minimizing C_Δ. Both problems are not as well studied as the makespan minimization problem.

An illustrative and small example revealing the non-equivalence of the three objectives in case of more than two machines is the following job-system consisting of seven jobs with positive processing times given by the vector $T = (46, 39, 27, 26, 16, 13, 10)$. Assuming $m = 3$, the (uniquely) optimal partitions are

- $\{\{J_1, J_7\}, \{J_2, J_4\}, \{J_3, J_5, J_6\}\}$ concerning C_{min},
- $\{\{J_1, J_5\}, \{J_2, J_6, J_7\}, \{J_3, J_4\}\}$ concerning C_{max},
- $\{\{J_1, J_6\}, \{J_2, J_5\}, \{J_3, J_4, J_7\}\}$ concerning C_Δ.

1.3 The heuristics LPT and RLPT

We put emphasis on the comparison of two well-known longest-first heuristics, the LPT- (longest processing time) and the RLPT-heuristic (restricted longest processing time) which are briefly described next.

The LPT-heuristic sorts all jobs in non-increasing order according to the processing times. Then, each job is assigned sequentially to the next machine available. Ties are broken arbitrarily.

In comparison, the RLPT-heuristic assigns the jobs rank by rank in order of increasing ranks. Jobs within a rank are assigned in non-increasing order according to the processing times to distinct machines as the machines become available after executing all previous ranks. Thus, with the RLPT-heuristic the assignment of the jobs of a certain rank is related to the current machine completion times after the execution of all previous ranks. This is the main difference compared to the LPT-heuristic where each job is assigned to the machine with minimum completion time so far, i.e., after the assignment of all previous jobs. So, even within a rank jobs do not have to be assigned to distinct machines in the LPT-schedule, as in Figure 1.

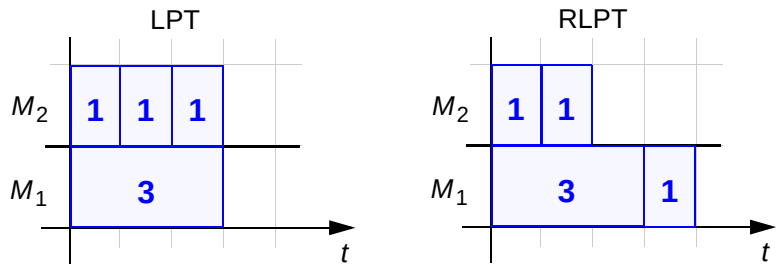

Figure 1: Exemplary LPT- and RLPT-schedule

1.4 Previous work and intention of the paper

As the underlying problem of maximizing C_{min} is known to be NP-hard, it seems unlikely that an efficient algorithm will be found for obtaining an optimal solution. Woeginger [8] derived the

first polynomial-time approximation scheme for the problem under consideration.

Of special interest is the result by Coffman and Sethi [3]. They proved that the makespan of the LPT-schedule is not worse than the makespan of the RLPT-schedule. The main result of this paper is the proof of the corresponding C_{min}-statement. In Section 2, we will show that the minimum completion time of the LPT-schedule is not worse than the minimum completion time of the RLPT-schedule.

Regarding the worst-case behavior of the LPT-heuristic applied to the C_{min}-maximization problem, Deuermeyer et al. [5] showed that the minimum completion time of the LPT-schedule is never less than 3/4 times the optimal minimum completion time. This bound is asymptotically tight when m tends to infinity. Csirik et al. [4] tightened the analysis for any fixed m and proved that the minimum completion time of the LPT-schedule is at least $(3m-1)/(4m-2)$ times the optimal minimum completion time. In addition, we take a look at the worst-case performance of the RLPT-heuristic in Section 3. We show that the minimum completion time of the RLPT-heuristic always remains within a factor of $1/m$ of the optimal minimum completion time. This bound is asymptotically tight but cannot be reached exactly.

2 Comparison of the minimum completion times

In this section we present a detailed comparison of the minimum completion times of LPT- and RLPT-schedules. Therefore, it is useful to introduce the term **profile** of a schedule which provides a measure finer than the minimum completion time and allows a comparison of (partial) schedules after each rank. So, let the multiset $\{h_1(g), \ldots, h_m(g)\}$ give the times at which the machines finish execution of tasks in rank g in schedule S^H. Then, the ordered m-tuple $h(g) = (h_1(g), \ldots, h_m(g))$ so that $h_i(g) \leq h_{i+1}(g)$ for all i is called *profile after rank g of the (partial) schedule* S^H. It is important to note that $h_i(g)$ and $h_i(g+1)$ may correspond to different machines. Moreover, note that $h_1(n/m) = C^H_{min}$ and $h_m(n/m) = C^H_{max}$. The main result of our research is the proof of the following theorem.

Theorem 2.1
The minimum completion time of the LPT-schedule is at least as long as the minimum completion time of the RLPT-schedule, i.e., $C^{LPT}_{min} \geq C^{RLPT}_{min}$.

Proof
We let $l(g)$ denote the LPT-profile and $r(g)$ denote the RLPT-profile after rank g. In order to prove the theorem we show that for all machines i with $l_i(g) < r_1(g) + t_{gm}$ it must be true that $l_i(g) \geq r_i(g)$. As we are particularly interested in the comparison of $l_1(g)$ and $r_1(g)$, with the previous statement we get in case $l_1(g) < r_1(g) + t_{gm}$ that $l_1(g) \geq r_1(g)$ must be true, and in the other case $l_1(g) \geq r_1(g) + t_{gm}$ we can directly conclude that $l_1(g) \geq r_1(g)$ as all processing times are non-negative. The proof works by induction in the number of ranks.

Base of Induction: $g = 1$

The LPT-heuristic assigns each job of rank 1 to a different machine. Hence, the LPT-schedule and the RLPT-schedule are identical after the assignment of rank 1.

Step of Induction: $g \rightarrow g+1$

Suppose that after rank g for all machines i with $l_i(g) < r_1(g)+t_{gm}$ it is true that $l_i(g) \geq r_i(g)$ and rank $g+1$ is being assigned next. Let $0 \leq k \leq m$ jobs of rank $g+1$, i.e., the jobs $gm+1, \ldots, gm+k$ (in case that $k \geq 1$), begin before $r_1(g) + t_{gm}$ in the LPT-schedule. As tasks even within a rank do not have to be assigned to distinct machines in the LPT-schedule, these k jobs will be assigned to the first $h \leq k$ elements of the $l(g)$-profile. The remaining $0 \leq m-k \leq m$ jobs of rank $g+1$ begin at or after $r_1(g)+t_{gm}$ in the LPT-schedule. Since $r_1(g+1) \leq r_1(g)+t_{gm}$, none of the machines that process at least one of the $m-k$ shortest jobs of rank $g+1$ can finish earlier than $r_1(g+1)+t_{(g+1)m}$ in the $l(g+1)$-profile. In other words, we assume that the following inequality-chain holds in the $l(g)$-profile:

$$r_1(g) + t_{gm} \leq l_1(g) \leq \ldots \leq l_m(g) \text{ (case } k = 0)$$

and

$$l_1(g) \leq \ldots \leq l_h(g) < r_1(g) + t_{gm} \leq l_{h+1}(g) \leq \ldots \leq l_m(g) \text{ (case } k \geq 1).$$

The inductive hypothesis ensures $l_i(g) \geq r_i(g)$ for all $i = 1, \ldots, h$.

Case 1: There exists a machine in the LPT-schedule that processes at least two of the $m - k$ shortest jobs of rank $g + 1$.

This directly yields $r_1(g+1) + t_{(g+1)m} \leq l_1(g+1)$, i.e., none of the m machines finishes before $r_1(g + 1) + t_{gm+m}$ in the $l(g + 1)$-profile.

Case 2: None of the machines in the LPT-schedule processes more than one of the $m - k$ shortest jobs of rank $g + 1$.

Here, we distinguish the following two main subcases $k = h$ and $k > h(\geq 1)$. Each of these two subcases will be subdivided further.

Subcase 1: $k = h$.

In case $h = 0$, each machine processes exactly one job of the current rank, i.e., element i of the $l(g)$-profile processes job $gm + i$ for $i = 1, \ldots, m$. As none of the jobs of the current rank starts before $r_1(g) + t_{gm}$ in the LPT-schedule, we can conclude further that no machine finishes earlier than $r_1(g + 1) + t_{gm+m}$ in the $l(g + 1)$-profile.

In case $h > 0$, element i of the $l(g)$-profile processes job $gm + i$ for $i = 1, \ldots, h$. From the inductive hypothesis we know that $l_i(g) + t_{gm+i} \geq r_i(g) + t_{gm+i}$ for $i = 1, \ldots, h$. As mentioned earlier, the $m-k$ machines that process exactly one of the $m-k$ shortest jobs of the current rank cannot finish earlier than $r_1(g + 1) + t_{gm+m}$.

Provided that the last $m - k$ elements of the $l(g)$-profile each process exactly one of the $m - k$ shortest jobs of the current rank, only the corresponding machines to the first h elements in the $l(g)$-profile can finish earlier than $r_1(g + 1) + t_{gm+m}$ in the $l(g + 1)$-profile. Due to the inductive hypothesis we can find for each of these h machines a distinct machine in the $r(g+1)$-profile which finishes not later. Thus, inequality $l_i(g+1) \geq r_i(g+1)$ holds for all i with $l_i(g + 1) < r_1(g + 1) + t_{(g+1)m}$.

In the other case, at least one of the first h elements in the $l(g)$-profile processes one of the $m - k$ shortest jobs. So, assume that $\bar{h} \leq \min\{h, m - h\}$ of the first h elements in the $l(g)$-profile process exactly one of the shortest $m - k$ jobs. This means that the last $\bar{h}$ elements in the $l(g)$-profile do not process any job of the current rank. Further, after having assigned $gm + h$ jobs assume that $l_{h_1}(g) + t_{gm+h_1}$ ($h_1 \in \{1, \ldots, h\}$) is the longest current completion time of all $\bar{h}$ machines of the LPT-schedule that process exactly one of the longest k jobs and one of the shortest $m-k$ jobs of the current rank. Then, we can conclude $l_{h_1}(g) + t_{gm+h_1} \leq l_{m-\bar{h}+1}(g)$. We also know that only the $h - \bar{h}$ elements out of the first h elements in the $l(g)$-profile which process exactly one job of the current rank and the last $\bar{h}$ elements in the $l(g)$-profile which do not process any job of the current rank can finish earlier than $r_1(g+1)+t_{gm+m}$ in the $l(g+1)$-profile. Again, we can find for each of these at most h machines a distinct machine in the $r(g + 1)$-profile which finishes not later. Thus, inequality $l_i(g+1) \geq r_i(g+1)$ holds for all i with $l_i(g+1) < r_1(g+1)+t_{(g+1)m}$.

Subcase 2: $k > h$.

In this case, at least one of the first $h \geq 1$ elements in the $l(g)$-profile processes more than one of the k longest jobs of the current rank. This means that at least the last $k-h$ elements of the $l(g)$-profile do not process any job of the current rank.

Assume that $gm + \bar{k}$ ($2 \leq \bar{k} \leq h + 1 \leq k$) is the first job of the current rank that is not assigned to element $l_{\bar{k}}(g)$. So, after the assignment of $gm+\bar{k}-1$ jobs we have the current completion times

$$\begin{aligned} &l_i(g) + t_{gm+i} \ (i = 1, \ldots, \bar{k} - 1) \\ &\text{and } \ l_i(g) \ (i = \bar{k}, \ldots, m) \end{aligned}$$

in the LPT-schedule and

$$r_i(g) + t_{gm+i} \ (i = 1, \ldots, \bar{k} - 1)$$
$$\text{and } \ r_i(g) \ (i = \bar{k}, \ldots, m)$$

in the RLPT-schedule. As the first h elements in the $l(g)$-profile fulfill the condition $l_i(g) < r_1(g) + t_{gm}$ $(i = 1, \ldots, h)$, the inductive hypothesis ensures

$$l_i(g) + t_{gm+i} \geq r_i(g) + t_{gm+i}$$

for $i = 1, \ldots, \bar{k} - 1$.
Job $gm + \bar{k}$ is assigned to one of the first $(\bar{k} - 1)$ elements in the $l(g)$-profile, i.e.,

$$\min_{1 \leq i \leq \bar{k}-1} \{l_i(g) + t_{gm+i}\} < l_{\bar{k}}(g).$$

In particular, we know

$$r_1(g+1) \leq \min_{1 \leq i \leq \bar{k}-1} \{r_i(g) + t_{gm+i}\} \leq \min_{1 \leq i \leq \bar{k}-1} \{l_i(g) + t_{gm+i}\}.$$

By this, we can directly conclude that none of the machines that process at least one of the jobs $gm+\bar{k}, \ldots, gm+m$ can finish earlier than $r_1(g+1)+t_{gm+m}$ in the $l(g+1)$-profile. Thus, if a machine processes at least two of the jobs $gm + \bar{k}, \ldots, gm + m$, then none of the m machines finishes before $r_1(g + 1) + t_{gm+m}$ in the $l(g + 1)$-profile. In the other case, i.e., the jobs $gm + \bar{k}, \ldots, gm + m$ are assigned to distinct machines, at most the elements out of the first $\bar{k} - 1$ elements in the $l(g)$-profile that process exactly one job of the current rank and the last $k - h$ elements in the $l(g)$-profile which do not process any job of the current rank can finish earlier than $r_1(g + 1) + t_{gm+m}$ in the $l(g + 1)$-profile. These are at most $(\bar{k} - 1 - (k - h)) + (k - h) = \bar{k} - 1 \leq h$ machines. For each of them in the $l(g + 1)$-profile we can find a distinct machine in the $r(g + 1)$-profile which finishes not later. This is correct since $l_i(g) + t_{gm+i} \geq r_i(g) + t_{gm+i}$ for all $i \in \{1, \ldots, \bar{k} - 1\}$ and $l_{m-k+h+1}(g) \geq l_{\bar{k}}(g) > \min_{i=1,\ldots,\bar{k}-1}\{l_i(g) + t_{gm+i}\}$. Thus, we get $l_i(g + 1) \geq r_i(g + 1)$ for all i with $l_i(g + 1) < r_1(g + 1) + t_{(g+1)m}$.

This completes the proof of Theorem 2.1. ■

The next corollary is a direct consequence of Theorem 2.1 and the result $C_{max}^{LPT} \leq C_{max}^{RLPT}$ by Coffman and Sethi (see [3]).

Corollary 2.2
The C_Δ-value of the LPT-schedule is at most as large as the C_Δ-value of the RLPT-schedule, i.e., $C_\Delta^{LPT} \leq C_\Delta^{RLPT}$.

To sum up, we can state that the LPT-heuristic generates schedules which are more balanced (in the sense of Subsection 1.2) than RLPT-schedules. An even stronger conclusion is that the RLPT-heuristic is dominated by the LPT-heuristic concerning any of the three objective functions C_{max}, C_{min} and C_Δ. This means that the application of the RLPT-heuristic to any job-system of the underlying problem cannot lead to better results than the LPT-heuristic yields. Nevertheless, the RLPT-heuristic might be a reasonable procedure whenever cardinality-balanced schedules, i.e., each machine processes n/m jobs, are required. For those scenarios, the LPT-heuristic is inapplicable. The interest in cardinality-balanced schedules arises from practical scheduling problems such as the allocation of component types to VLSI-chip manufacturing machines (cf. [7]). For this reason we will take a look at the worst-case as well as the probabilistic performance of the RLPT-heuristic in the subsequent sections.

3 Worst-case analysis of the RLPT-heuristic

This section deals with a determination of the worst-case ratios $C_{min}^{RLPT}/C_{min}^{*}$ and $C_{min}^{RLPT}/C_{min}^{LPT}$.

Theorem 3.1
The performance bounds

$$\frac{C_{min}^{RLPT}}{C_{min}^{*}} > \frac{1}{m} \quad \text{and} \quad \frac{C_{min}^{RLPT}}{C_{min}^{LPT}} > \frac{1}{m}$$

are asymptotically tight for any fixed number $m \geq 2$ of machines but cannot be reached exactly.

Note that the same bound applies when RLPT-scheduling is compared to LPT-scheduling instead of optimal scheduling.

We do not intend to prove Theorem 3.1 in all detail. We rather give a sketch of the proof and present a family of instances for any fixed number $m \geq 2$ that approaches the bound. The main idea of the proof is to compare the RLPT-heuristic with the SPT-heuristic (shortest processing time) which sorts all jobs in non-decreasing order according to the processing times and assigns each job sequentially to the next machine available. This leads to a simple but nice structure of SPT-schedules.

Lemma 3.2
Whenever the SPT-heuristic assigns a job to a machine, then this machine has maximum completion time so far afterwards.

This result is quite obvious, so the proof is omitted. Clearly, Lemma 3.2 can be used to determine all machine completion times in an SPT-schedule.

Corollary 3.3
The completion time of the i-th longest running machine in an SPT-schedule is given by $t_i + t_{m+i} + \ldots + t_{n-m+i}$.

With the previous corollary we can directly conclude Corollary 3.4.

Corollary 3.4
The minimum completion time of the RLPT-schedule is at least as long as the minimum completion time of the SPT-schedule, i.e., $C_{min}^{RLPT} \geq C_{min}^{SPT}$.

Proof
The proof is quite simple, too. As jobs of the same rank have to be assigned to distinct machines in the RLPT-schedule we can conclude that

$$C_{min}^{RLPT} \geq t_m + t_{2m} + \ldots + t_n = C_{min}^{SPT}$$

whereas the equality is due to Corollary 3.3. ■

Hence, we can conclude $C_{min}^{RLPT}/C_{min}^{*} \geq C_{min}^{SPT}/C_{min}^{*}$. As the SPT-heuristic is a special case of the List-Scheduling algorithm whose performance ratio is $1/m$ concerning C_{min}-maximization as shown in [8], we can deduce $C_{min}^{SPT}/C_{min}^{*} \geq 1/m$. To verify that this lower bound is tight for any fixed number m of machines, we present the following job-system consisting of $2m-1$ jobs with positive processing times:

$$t_1 = \ldots = t_{m-1} = m,$$
$$t_m = \ldots = t_{2m-1} = 1.$$

This job-system also reveals the tight lower bound of $1/m$ of the ratio $C_{min}^{SPT}/C_{min}^{LPT}$. So far, we can conclude the following lower bounds:

$$\frac{C_{min}^{RLPT}}{C_{min}^{*}} \geq \frac{1}{m} \quad \text{and} \quad \frac{C_{min}^{RLPT}}{C_{min}^{LPT}} \geq \frac{1}{m}.$$

Next, we present a family of job-systems for any fixed number $m \geq 2$ so that the minimum completion times of the RLPT-schedules approach the $1/m$-bound. Therefore, assume $d \in \mathbb{N}$ and consider the following job-system:

$$t_1 = \ldots = t_{m-1} = dm,$$
$$t_m = \ldots t_{dm+m-1} = 1.$$

Then, we get $C^*_{min} = dm = C^{LPT}_{min}$, whereas the minimum completion time of the RLPT-schedule is $C^{RLPT}_{min} = d + 1$. Hence,

$$\frac{C^{RLPT}_{min}}{C^*_{min}} = \frac{C^{RLPT}_{min}}{C^{LPT}_{min}} = \frac{d+1}{dm} = \frac{1}{m} + \frac{1}{dm} \underset{d\to\infty}{\longrightarrow} \frac{1}{m}.$$

It remains to show that the bounds cannot be reached exactly. As mentioned before, we do not intend to prove this in all detail. The main idea of this last part of the proof is to consider the set of job-systems for which the SPT-heuristic generates schedules with a minimum completion time of exactly $1/m$ of the optimal minimum completion time. In these cases, the processing times have to fulfill the following properties:

(*i*) The $n-m+1$ shortest processing times must sum up to at most t_{m-1}, i.e., $\sum_{j=m}^{n} t_j \leq t_{m-1}$.

(*ii*) The number of jobs n must be larger than m and

$$\begin{aligned} & t_1 \geq t_2 \geq \ldots \geq t_{m-1} > \\ > & t_m = \ldots = t_{2m-1} \geq \\ & \vdots \\ \geq & t_{n-m} = \ldots = t_{n-1} > \\ > & t_n = 0. \end{aligned}$$

Then, it is rather straightforward to verify that for any of those job-systems inequality $C^{RLPT}_{min} > C^{SPT}_{min}$ holds.

4 Experimental results

As mentioned at the end of Section 2, scheduling scenarios exist that require cardinality-balanced schedules. In contrast to the RLPT-heuristic, the LPT-heuristic is inapplicable in such scenarios. To gain a little insight into the probabilistic behavior of the RLPT-heuristic we conducted an experimental study. Thereby, we determined how often the RLPT-heuristic does not generate a worse schedule than the LPT-heuristic. In case of $m = 2$ machines we found interesting relations, although we do not have theoretical proofs so far.

In our experiments we assumed the processing times to be independent samples, uniformly distributed in the unit interval $[0, 1]$. We shall also remark that n denotes the number of jobs with positive processing times in this section, and n needs not to be a multiple of m. Our experimental results led us to the following conjecture.

Conjecture 4.1
Assume the processing times to be independent samples, uniformly distributed in the unit interval $[0, 1]$ *and assume* $m = 2$ *and* $n \geq 4$. *Then,*

(i) the probability $Pr\{C^{RLPT}_{min} = C^{LPT}_{min}\}$ *that the RLPT- and the LPT-schedule have the same minimum completion time is*

$$Pr\{C^{RLPT}_{min} = C^{LPT}_{min}\} = \begin{cases} \frac{3}{4} & \text{if } 2 \mid n, \\ \frac{7}{8} & \text{if } 2 \nmid n. \end{cases}$$

(ii) the probability $Pr\{C_{min}^{RLPT}(n-k) = C_{min}^{LPT}(n-k)\}$ *that the RLPT- and the LPT-schedule have the same (current) minimum completion time after the assignment of the* $(n-k)$ *longest jobs is*

$$Pr\{C_{min}^{RLPT}(n-k) = C_{min}^{LPT}(n-k)\} = 1 - \frac{1}{2^{k+2}}$$

in case n *and* k *are of same parity and* $(n-k) \geq 4$.

To briefly summarize the experimental results we can state that, depending on the parity of n but not on the concrete number of jobs, in 75% or 87.5% of cases equality $C_{min}^{RLPT} = C_{min}^{LPT}$ holds. In other words, the omission of the cardinality-balance constraint leads only in 25% or 12.5% of cases to a better LPT-schedule.

5 Conclusions

For the problem of maximizing the minimum completion time, we proved that the RLPT-heuristic is outperformed and dominated by the LPT-heuristic. Nevertheless, in scheduling scenarios with a cardinality-balance constraint on schedules the RLPT-heuristic seems to be an appropriate procedure. At least in case of two machines, this additional constraint rarely results in a worse minimum completion time compared to the LPT-heuristic which disregards this constraint.

References

[1] Bruno, J.; Coffman, E. G., Jr.; Sethi, R.: Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM 17*, No. 7, 382–387 (1974).

[2] Coffman, E. G., Jr.; Langston, M. A.: A performance guarentee for the greedy set-partitioning algorithm. *Acta Informatica 21*, 409–415 (1984).

[3] Coffman, E. G., Jr.; Sethi, R.: Algorithms minimizing mean flow time: schedule-length properties. *Acta Informatica 6*, 1–14 (1976).

[4] Csirik, J.; Kellerer, H.; Woeginger, G.: The exact LPT-bound for maximizing the minimum completion time. *Operations Research Letters 11*, Issue 5, 281–287 (1992).

[5] Deuermeyer, B. L.; Friesen, D. K.; Langston, M. A.: Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM Journal on Algebraic and Discrete Methods 3*, No. 2, 190–196 (1982).

[6] Friesen, D. K.; Deuermeyer, B. L.: Analysis of greedy solutions for a replacement part sequencing problem. *Mathematics of Operations Research 6*, No. 1, 74–87 (1981).

[7] Tsai, L.-H.: Asymptotic analysis of an algorithm for balanced parallel processor scheduling. *SIAM Journal on Computing 21*, No. 1, 59–64 (1992).

[8] Woeginger, G. J.: A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters 20*, Issue 4, 149–154 (1997).