CEJOR (2013) 21:125–139
DOI 10.1007/s10100-011-0217-4

ORIGINAL PAPER

Comparing the minimum completion times of two longest-first scheduling-heuristics

Rico Walter

Published online: 7 July 2011

Abstract For the basic 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 the performance relationship between two well-known longest-first heuristics—the LPT- (longest processing time) and the RLPT-heuristic (restricted LPT). We provide insights into the solution structure of these two sequencing heuristics and 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 comprehensive experimental study of the probabilistic behavior of RLPT-schedules compared to LPT-schedules in the two-machine case.

Keywords Scheduling · Identical parallel machines · Heuristics · Minimum completion time · Worst-case analysis

1 Introduction

1.1 Problem description and notation

This paper deals with a fundamental problem in scheduling theory that is formally described as follows. 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;

R. Walter (✉)
Fakultät für Wirtschaftswissenschaften, Lehrstuhl für ABWL/Management Science,
Friedrich-Schiller-Universität Jena, Carl-Zeiß-Straße 3, 07743 Jena, Germany
e-mail: rico.walter@uni-jena.de

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 $t_1 \geq \cdots \geq t_n \geq 0$. 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 Sects. 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 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 Friesen and Deuermeyer 1981) or more generally in the assignment of spare parts to machines for repeated repair. Regional allocation of investments is another application of this problem as mentioned by Haouari and Jemmali (2008).

Maximizing the minimum completion time is in some sense 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" (see Coffman and Langston 1984). 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 by far not as well studied as the makespan minimization problem.

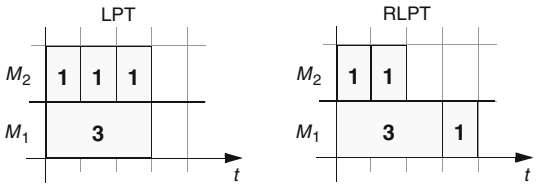

Fig. 1 Exemplary LPT- and RLPT-schedule

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 sequencing 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 Fig. 1.

1.4 Previous work and intention of the paper

As mentioned before, literature related to the $C_{\min}$-maximization problem is rather sparse compared to the multitude of contributions to the classical makespan minimization problem. Most of the literature on $C_{\min}$-maximization is devoted to the analysis of heuristics and approximation algorithms, since the underlying problem is known to be NP-hard. So, Deuermeyer et al. (1982) 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. (1992) 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. Furthermore, Woeginger (1997) derived the first polynomial-time approximation scheme for the problem under consideration. On-line versions of the problem have been treated by Azar and Epstein (1997) as well as He and Tan (2002). In a more recent publication, Haouari and Jemmali (2008) proposed an efficient exact branch-and-bound algorithm based on tight upper and lower bounds and a symmetry-breaking branching scheme.

The paper on hand provides new insights into the solution structure of the two sequencing heuristics LPT and RLPT and a fundamental result concerning the performance relationship between these heuristics is established. We prove that the minimum completion time of the LPT-schedule is not worse than the minimum completion time of the RLPT-schedule. This result contributes substantially to an earlier one by Coffman and Sethi (1976) on the performance of LPT and RLPT for the makespan minimization problem. Furthermore, we contribute to the worst-case analysis of heuristics by proving that the minimum completion time of the RLPT-heuristic always remains within a factor of $1/m$ of the optimal minimum completion time.

For this purpose, the remainder of the paper is organized as follows. Section 2 contains the detailed proof of the main result on the relationship between the minimum completion time of LPT- and RLPT-schedules. In Sect. 3 we take a look at the worst-case performance of the RLPT-heuristic. Then, Sect. 4 presents our experimental results on different questions with regard to the average case performance of the two sequencing heuristics under consideration. Concluding remarks within Sect. 5 finish the paper.

2 Comparison of the minimum completion times

This section presents a detailed comparison of the minimum completion times of LPT- and RLPT-schedules and provides new structural insights into the corresponding schedules, as well. 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 jobs 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.,* $C^{LPT}_{\min} \geq C^{RLPT}_{\min}$.

In order to prove this theorem we prove an even stronger statement on the schedule structure of the two heuristics. Therefore, consider the next theorem.

Theorem 2.2 *Let* $l(g)$ *and* $r(g)$ *denote the LPT- and the RLPT-profile after rank* g*, respectively. Then, for all machines* i *with* $l_i(g) < r_1(g) + t_{gm}$ *it must be true that* $l_i(g) \geq r_i(g)$.

Note that we are particularly interested in the comparison of $l_1(g)$ and $r_1(g)$ for $g = n/m$ but as well as after all previous ranks. However, it is not enough to consider only the first element in each of the two profiles because the first element of a profile may correspond to different machines from rank to rank. For this reason, a deeper knowledge about the machine completion times after each rank is inevitable which is provided by Theorem 2.2. This theorem ensures that in case $l_1(g) < r_1(g) + t_{gm}$ it must be true that $l_1(g) \geq r_1(g)$, and in the other case $l_1(g) \geq r_1(g) + t_{gm}$ we can directly conclude that $l_1(g) \geq r_1(g)$ because all processing times are non-negative.

The proof of Theorem 2.2 will be rather technical. So, to facilitate understanding we illustrate all of the (sub-)cases appearing in the proof by examples afterwards.

Proof of Theorem 2.2 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, i.e., $l(1) = r(1)$.

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

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 jobs 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. So each of the machines that process at least one of the $m - k$ shortest jobs of rank $g + 1$ cannot finish earlier than $r_1(g) + t_{gm} + t_{(g+1)m}$ in the $l(g + 1)$-profile.

In contrast to the LPT-heuristic, jobs within a rank have to be assigned to distinct machines in the RLPT-schedule. Hence, in the $r(g + 1)$-profile there exists a machine with current completion time $r_1(g) + t_{gm+1}$. Either this machine has minimum completion time after rank $g + 1$ or there exists another machine which finishes even earlier. So, we have $r_1(g + 1) \leq r_1(g) + t_{gm+1} \leq r_1(g) + t_{gm}$. Moreover, we can deduce $r_1(g) + t_{gm} + t_{(g+1)m} \geq r_1(g + 1) + t_{(g+1)m}$.

The following inequality-chains elucidate the situation in the $l(g)$-profile that is presupposed:

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

and

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

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 $g + 1$.

From the previous part we know that each machine that processes one of the $m-k$ shortest jobs of rank $g+1$ cannot finish earlier than $r_1(g+1)+t_{(g+1)m}$ in the $l(g+1)$-profile. Hence, if such a machine processes more than one of the $m-k$ shortest jobs, then none of the m machines finishes before $r_1(g+1)+t_{(g+1)m}$ in the $l(g+1)$-profile as with the LPT-heuristic jobs are always assigned sequentially to the next machine available. This yields $r_1(g+1)+t_{(g+1)m} \leq l_1(g+1)$.

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 $k=h=0$, we have $r_1(g)+t_{gm} \leq l_1(g) \leq \cdots \leq l_m(g)$ which means that no job of rank $g+1$ starts before $r_1(g)+t_{gm}$ in the LPT-schedule and no machine finishes earlier than $r_1(g+1)+t_{(g+1)m}$ in the $l(g+1)$-profile. Moreover, each machine processes exactly one job of the current rank, i.e., element i of the $l(g)$-profile processes job $gm+i$ for $i=1,\ldots,m$ because in Case 2 none of the machines processes more than one of the $m-k=m$ shortest jobs of rank $g+1$. Recall that each rank consists of exactly m jobs.
In case $k=h>0$, we have $l_1(g) \leq \cdots \leq l_h(g) < r_1(g)+t_{gm} \leq l_{h+1}(g) \leq \cdots \leq l_m(g)$ and we know that element i of the $l(g)$-profile processes job $gm+i$ for $i=1,\ldots,h$ because the $h(=k)$ longest jobs of rank $g+1$ are assigned to the first h elements of the $l(g)$-profile. 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 $r_1(g+1)+t_{(g+1)m}$.
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 $r_1(g+1)+t_{(g+1)m}$ in the $l(g+1)$-profile. Due to the inductive hypothesis, for each of these h machines there exists 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 $h(=k)$ jobs and one of the shortest $m-h$ 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 $r_1(g+1)+t_{(g+1)m}$ in the $l(g+1)$-profile. Again, for each of these at most h machines there exists 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

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

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 $gm+\bar{k}, \ldots, (g+1)m$ can finish earlier than $r_1(g+1)+t_{(g+1)m}$ in the $l(g+1)$-profile. Thus, if a machine processes at least two of the jobs $gm + \bar{k}, \ldots, (g+1)m$, then none of the m machines finishes before $r_1(g+1)+t_{(g+1)m}$ in the $l(g+1)$-profile. In the other case, i.e., the jobs $gm+\bar{k}, \ldots, (g+1)m$ are assigned to distinct machines -which is only possible if inequality $k \leq 2(\bar{k}-1) \leq 2h$ holds-, 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 $r_1(g+1)+t_{(g+1)m}$ 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 there exists 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 2.2. □

All in all, the proof of Theorem 2.2 consists of six cases, subcases, and the further subdivisions. Each of them is contained in one of the three subsequent examples which are meant for a better understanding.

Example 2.3 Consider $m = 4$ machines and $n = 20$ jobs, i.e., the jobs are divided into five ranks, with processing times given by $T_1 = (100, 90, 50, 42; 38, 35, 30, 30; 24, 20, 20, 16; 16, 12, 6, 6; 6, 6, 4, 3)$. Clearly, the profiles $l(1)$ and $r(1)$ are identical. So, we have $l(1) = r(1) = (42, 50, 90, 100)$ and $r_1(1) + t_4 = 42 + 42 = 84$. Hence, $h = 2$ and -due to the processing times of jobs in rank 2- $k = 3$ as well as $\bar{k} = 3$ which belongs to Subcase 2 of Case 2. Furthermore, the jobs 7 and 8 are assigned to distinct machines in the LPT-schedule. This leads to $l(2) = (90, 100, 110, 115)$ -note that $l_1(2)$ and $l_1(1)$ correspond to different machines-, $r(2) = (80, 85, 120, 130)$, and $r_1(2) + t_8 = 110$. So, $h = k = 2$, i.e., Subcase 1 of Case 2, and $\bar{h} = 1$ which means that the first element of $l(2)$ processes exactly one of the $m - k = 2$ shortest jobs of rank 3. We receive $l(3) = (115, 120, 130, 130)$ -note that $l_1(3), l_1(2)$, and $l_1(1)$ correspond to three different machines-, $r(3) = (104, 105, 140, 146)$, and $r_1(3) + t_{12} = 120$. Now, we get $h = k = 1$ and the last $m - k = 3$ elements of the $l(3)$-profile each process exactly one of the three shortest jobs of rank 4. This yields $l(4) = (131, 132, 136, 136)$, $r(4) = (117, 120, 146, 152)$ -now, $r_1(4)$ and $r_1(3)$ correspond to different machines-, and $r_1(4)+t_{16} = 123$. Hence, $h = k = 0$, i.e., no job of rank 5 starts before 123 in the LPT-schedule, and we get $l(5) = (137, 138, 139, 140)$ and $r(5) = (123, 126, 150, 155)$.

Example 2.4 Now, consider $m = 4$ machines and $n = 8$ jobs and let the processing times be given by $T_2 = (20, 18, 14, 6; 6, 4, 3, 2)$. Then, we get $l(1) = r(1) = (6, 14, 18, 20)$, $r_1(1) + t_4 = 12$, and $h = 1, k = 2, \bar{k} = 2$. Moreover, element 1 of the $l(1)$-profile processes two of the jobs 6,7,8, namely job 6 and job 8. Hence, none of the $m = 4$ machines finishes before $r_1(2) + t_8 = 14$ in the $l(2)$-profile. The respective profiles after the assignment of rank 2 are $l(2) = (17, 18, 18, 20)$ and $r(2) = (12, 18, 21, 22)$.

Example 2.5 Eventually, consider $m = 4$ machines and $n = 12$ jobs and let the processing times be given by $T_3 = (55, 46, 25, 20; 20, 18, 15, 7; 7, 1, 1, 1)$. As this example is meant to illustrate Case 1 of the proof of Theorem 2.2, we directly start with the profiles after the assignment of rank 2 which are as follows: $l(2) = (46, 50, 55, 55)$ and $r(2) = (40, 43, 61, 62)$. Since $r_1(2) + t_8 = 47$ we get $h = k = 1$. However, element 2 of the $l(2)$-profile processes all $m - k = 3$ shortest jobs of rank 3 (Case 1). Hence, none of the $m = 4$ machines finishes before $r_1(3)+t_{12} = 45$ in the $l(3)$-profile. The respective profiles are $l(3) = (53, 53, 55, 55)$ and $r(2) = (44, 47, 62, 63)$.

The following corollary contains an additional structural information on (partial) LPT- and RLPT-schedules that can be deduced from the proof of Theorem 2.2.

Corollary 2.6 *The number of machines i that fulfill $l_i(g) < r_1(g) + t_{gm}$ is monotonically decreasing in the number of assigned ranks.*

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 (1976).

Corollary 2.7 *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 Sect. 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 is an applicable procedure whenever cardinality-balanced schedules, i.e., each machine processes n/m jobs, are required. For those scenarios, the LPT-heuristic may generate infeasible solutions. The interest in cardinality-balanced schedules arises from practical scheduling problems such as the allocation of component types to VLSI-chip manufacturing machines (see Tsai 1992). For this reason we will take a look at the worst-case as well as the probabilistic performance of the RLPT-heuristic -albeit still for the unconstrained problem- in the subsequent sections.

3 Worst-case analysis of the RLPT-heuristic

In this section we are concerned with the determination of the worst-case ratios $C_{\min}^{RLPT}/C_{\min}^{*}$ and $C_{\min}^{RLPT}/C_{\min}^{LPT}$.

Theorem 3.1 *The performance bounds*

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

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 every 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. Furthermore, it is worth mentioning that there always exists an SPT-schedule in which jobs of the same rank are assigned to distinct machines.

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

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 of Corollary 3.4 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

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

whereas the equality is due to Corollary 3.3. This completes the proof of Corollary 3.4. □

Hence, we can conclude $C_{\min}^{RLPT}/C_{\min}^{*} \geq C_{\min}^{SPT}/C_{\min}^{*}$. As Woeginger (1997) showed that the List-Scheduling algorithm has a performance ratio of $1/m$ concerning $C_{\min}$-maximization, we can deduce $C_{\min}^{SPT}/C_{\min}^{*} \geq 1/m$. It is readily verified that this bound is tight. So far, we can conclude $C_{\min}^{RLPT}/C_{\min}^{LPT} \geq C_{\min}^{RLPT}/C_{\min}^{*} \geq 1/m$.

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:

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

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

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

It remains to show that the bounds cannot be reached exactly. As mentioned before, we do not intend to prove this in all its particulars. 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

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

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

4 Experimental study and results

As mentioned at the end of Sect. 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 more insight into the coherences between RLPT- and LPT-schedules we conducted an experimental study in case of $m = 2$ machines. Thereby, we determined how often

(i) the RLPT-heuristic does not generate a worse schedule than the LPT-heuristic,
(ii) the RLPT-heuristic does not generate a worse (partial) schedule than the LPT-heuristic after the assignment of $n - k \geq 4$ jobs.

In case that the LPT-schedule is better than the RLPT-schedule we also determined to which extent the minimum completion times differ on average as well as on maximum.

In our experiments we assumed the processing times to be independent samples, uniformly distributed in the unit interval [0, 1]. For each (n, k)-combination studied we generated 10^7 independent instances. We tested different values for n in the range from 4 to 501 as well as seven different values for k depending on the parity of n, i.e., $k \in \{0, 2, 4, 6, 8, 10, 12\}$ in case n is even and $k \in \{1, 3, 5, 7, 9, 11, 13\}$ in case n is odd. 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.

In Table 1, results rounded to four decimal places of the comparison of LPT- and RLPT-schedules are included. Thereby, the line labeled "EQU" refers to the relative number of instances for which the minimum completion time of the RLPT-schedule equals the minimum completion time of the LPT-schedule. In case that the minimum completion times differ, we computed the average ratio of $C_{\min}^{RLPT}$ to $C_{\min}^{LPT}$ (labeled AVG) and we saved the minimal ratio (labeled MIN), as well.

The results of our simulations reveal 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. However, the average ratio of the minimum completion time of the RLPT-schedule to the minimum completion time of the LPT-schedule quickly approaches 1 as the number of jobs increases. In other words, the average relative difference -measured as $(1 - AVG)$- between the two minimum

Table 1 Comparison of LPT- and RLPT-schedules

n	4	5	6	7	10	11	12	13	50	51
EQU	0.7499	0.8749	0.7500	0.8750	0.7500	0.8751	0.7500	0.8751	0.7499	0.8749
AVG	0.8982	0.9040	0.9462	0.9504	0.9825	0.9824	0.9882	0.9879	0.9993	0.9993
MIN	0.6668	0.6692	0.6050	0.6147	0.6360	0.6535	0.6587	0.6948	0.9917	0.9920
n	52	53	100	101	102	103	250	251	500	501
EQU	0.7499	0.8750	0.7499	0.8749	0.7498	0.8751	0.7501	0.8751	0.7499	0.8751
AVG	0.9994	0.9993	0.9998	0.9998	0.9998	0.9998	1.0000	1.0000	1.0000	1.0000
MIN	0.9919	0.9925	0.9973	0.9978	0.9977	0.9979	0.9996	0.9996	0.9999	0.9999

completion times is negligible. The same is true if we take a look at the maximal relative difference (or minimal ratio) instead. Nevertheless, for small numbers of jobs we experimentally found instances that yield a ratio of approximately 0.6. Recall from Sect. 3 that the ratio of $C_{\min}^{RLPT}$ to $C_{\min}^{LPT}$ is bounded below by 0.5 in case of two machines.

In a second part of our experimental studies we asked for the probability that the partial RLPT-schedule, i.e., after the assignment of a subset of the n jobs, is not worse than the partial LPT-schedule so far. This question comes along with the following one: The assignment of which job $\bar{j}$ leads to a worse partial RLPT-schedule than the respective partial LPT-schedule for the first time, i.e., $C_{\min}^{RLPT}(\bar{j}) < C_{\min}^{LPT}(\bar{j})$ and $C_{\min}^{RLPT}(j) = C_{\min}^{LPT}(j)$ for all previous jobs $j = 1, \ldots, \bar{j} - 1$. Clearly, $\bar{j}$ does not exist in every instance but, if it does so, it is rather straightforward to show that $\bar{j}$ is even and $\bar{j} \geq 4$ in case of $m = 2$ identical machines. The idea is to consider the minimum completion times of the two partial schedules after the assignment of an even number of jobs. Assume $C_{\min}^{RLPT}(2j) = C_{\min}^{LPT}(2j)$ for some $j \in \{1, \ldots, n/2 - 1\}$. Note, that this is equivalent to $C_{\Delta}^{RLPT}(2j) = C_{\Delta}^{LPT}(2j)$. Then, two cases have to be distinguished depending on the relation between $C_{\Delta}^{RLPT}(2j)$ and the processing time t_{2j+1} of the next job. The first case, i.e., $C_{\Delta}^{RLPT}(2j) \leq t_{2j+1}$, obviously cannot lead to different minimum completion times of the two schedules after the assignment of the jobs $2j + 1$ and $2j + 2$ as the two jobs are assigned to distinct machines even in the LPT-schedule. In the other case, i.e., $C_{\Delta}^{RLPT}(2j) > t_{2j+1}$, job $2j + 2$ is assigned to the same machine as job $2j + 1$ is assigned to in the LPT-schedule. This yields $C_{\min}^{RLPT}(2j+2) \leq C_{\min}^{LPT}(2j+2)$ where equality holds only in case $t_{2j+2} = 0$. Using the fact that equality $C_{\min}^{RLPT}(2j) = C_{\min}^{LPT}(2j)$ holds for the starting parameter $j = 1$, it follows immediately that after the assignment of an odd number of jobs the minimum completion times of the partial RLPT- and LPT-schedule cannot differ for the first time. Hence, $\bar{j}$ has to be even and $t_{\bar{j}} > 0$. Note that $C_{\Delta}^{RLPT}(2j) > t_{2j+1}$ for some j constitutes a necessary condition for $C_{\min}^{RLPT}(n) < C_{\min}^{LPT}(n)$ in the two-machine case.

Consider the exemplary job-system of Fig. 1, then we have $C_{\Delta}^{LPT}(2) = C_{\Delta}^{RLPT}(2) = 2 > t_3 = 1$ and $\bar{j} = 4$. We shall also remark that rare cases exist where RLPT, after $C_{\min}^{RLPT}(j) < C_{\min}^{LPT}(j)$ for some intermediate $j < n$, returns to $C_{\min}^{RLPT}(n) = C_{\max}^{LPT}(n)$ at the end. For instance, this is the case if we add an additional job with length $t_5 = 1$ to the job-system of Fig. 1. This yields $C_{\min}^{RLPT}(5) = C_{\min}^{LPT}(5) = 3$ whereas $C_{\min}^{RLPT}(4) < C_{\min}^{LPT}(4)$.

Table 2 Comparison of partial LPT- and RLPT-schedules

n	16						
k	0	2	4	6	8	10	12
EQU	0.75000	0.93745	0.98432	0.99608	0.99903	0.99975	0.99994
REF	0.75000	0.93750	0.98438	0.99609	0.99902	0.99976	0.99994
n	17						
k	1	3	5	7	9	11	13
EQU	0.87506	0.96876	0.99213	0.99803	0.99950	0.99988	0.99997
REF	0.87500	0.96875	0.99219	0.99805	0.99951	0.99988	0.99997
n	50						
k	0	2	4	6	8	10	12
EQU	0.75005	0.93750	0.98436	0.99611	0.99902	0.99975	0.99995
n	51						
k	1	3	5	7	9	11	13
EQU	0.87502	0.96875	0.99216	0.99805	0.99953	0.99988	0.99997
n	100						
k	0	2	4	6	8	10	12
EQU	0.75022	0.93752	0.98440	0.99605	0.99903	0.99976	0.99994
n	101						
k	1	3	5	7	9	11	13
EQU	0.87491	0.96872	0.99220	0.99806	0.99951	0.99987	0.99997

Table 2 contains our experimental results concerning the comparison of partial LPT- and RLPT-schedules. Thereby, the line labeled "EQU" refers to the relative number of instances for which the minimum completion time of the partial RLPT-schedule equals the minimum completion time of the partial LPT-schedule after the assignment of $(n-k)$ jobs. Due to the previous result on the parity of $\bar{j}$ we only considered (n,k)-constellations such that the difference $n-k$ is at least four and even. This time, the results are rounded to five decimal places with regard to Conjecture 4.1. For the same reason, we added the line labeled "REF" twice which represents the rounded values of $1-1/2^{k+2}$ and serves as a comparison of the experimental results.

Again, the results indicate that the probabilistic performance of partial RLPT-schedules in comparison to partial LPT-schedules depends strongly on the parity of n for a given k. Moreover, the probability that $C_{\min}^{RLPT}(n-k)$ equals $C_{\min}^{LPT}(n-k)$ is quickly approaching 1.0 with increasing k.

To sum up, our experimental results incorporate interesting relations between LPT- and RLPT-schedules in case of two machines which 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* $Pr\{C_{\min}^{RLPT} = C_{\min}^{LPT}\}$ *that the RLPT- and the LPT-schedule have the same minimum completion time is*

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

(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* $(n-k) \geq 4$.

Note that the two cases $k = 0$ (and n even) as well as $k = 1$ (and n odd) of the second part are consistent with the first part. Although we do not have a theoretical proof of Conjecture 4.1 so far, we are able to prove the second part in the special event that $n - k = 4$, which further supports this conjecture drawn from our experimental results. In this special case we are asked to determine the probability that the longest processing time is shorter than the sum of the second and the third longest processing time which ensures $C_{\min}^{RLPT}(4) = C_{\min}^{LPT}(4)$. Using some information on the joint density function of three order statistics, the proof consists merely in a rather straightforward calculation of the following integral:

$$\int_{x_3=0}^{1}\int_{x_2=x_3}^{1}\int_{x_1=x_2}^{1} f_{3,2,1:n}(x_3,x_2,x_1)\mathbf{1}_{(x_1<x_2+x_3)}dx_1dx_2dx_3 = \int_{0}^{1}\int_{x_3}^{1}\int_{x_2}^{\min\{1,x_2+x_3\}} \tfrac{n!}{(n-3)!}x_3^{n-3}dx_1dx_2dx_3$$

where

$$f_{3,2,1:n}(x_3, x_2, x_1) = \frac{n!}{(n-3)!}x_3^{n-3}, \quad (0 \leq x_3 < x_2 < x_1 \leq 1),$$

is the joint density function (see Arnold et al. (1992), pp. 25–26) of the third largest, the second largest and the largest order statistic with respect to our stochastic model. It is not difficult to verify that the value of the triple integral is $1 - 1/2^{n-2}$.

5 Conclusions

The paper on hand contributes to the basic scheduling problem of maximizing the minimum completion time of a set of independent jobs on identical parallel machines by an in-depth analysis of the solution-structure of two well-known longest first

sequencing heuristics (LPT and RLPT). It is proved that the RLPT-heuristic is outperformed and dominated by the LPT-heuristic in terms of $C_{\min}$-maximization. However, at least in case of two machines experimental results show that the relative difference of the respective minimum completion times is negligible despite a theoretical worst case performance of $1/m$.

Unlike LPT, the RLPT-heuristic always generates cardinality-balanced schedules and is therefore a feasible procedure for the constrained version of the problem discussed in this paper. Here, an interesting direction for future research is to transfer the results on the worst-case performance of the RLPT-heuristic from the unconstrained problem to the constrained one. Thereby, two basic versions of the constrained problem are conceivable: cardinality-balance has to be fulfilled after the assignment of each rank or -in a weaker sense- only after the last rank.

Acknowledgments The author would like to thank the anonymous referees for their valuable suggestions which considerably helped to improve the presentation of results in this paper.

References

Arnold BC, Balakrishnan N, Nagaraja HN (1992) A first course in order statistics. Wiley, New York

Azar Y, Epstein L (1997) On-line machine covering. Lect Notes Comput Sci 1284:23–36

Coffman EG Jr, Langston MA (1984) A performance guarantee for the greedy set-partitioning algorithm. Acta Informatica 21:409–415

Coffman EG Jr, Sethi R (1976) Algorithms minimizing mean flow time: schedule-length properties. Acta Informatica 6:1–14

Csirik J, Kellerer H, Woeginger G (1992) The exact LPT-bound for maximizing the minimum completion time. Oper Res Lett 11:281–287

Deuermeyer BL, Friesen DK, Langston MA (1982) Scheduling to maximize the minimum processor finish time in a multiprocessor system. SIAM J Algebraic Discret Methods 3:190–196

Friesen DK, Deuermeyer BL (1981) Analysis of greedy solutions for a replacement part sequencing problem. Math Oper Res 6:74–87

Haouari M, Jemmali M (2008) Maximizing the minimum completion time on parallel machines. 4OR Q J Oper Res 6:375–392

He Y, Tan ZY (2002) Ordinal on-line scheduling for maximizing the minimum machine completion time. J Comb Optim 6:199–206

Tsai L-H (1992) Asymptotic analysis of an algorithm for balanced parallel processor scheduling. SIAM J Comput 21:59–64

Woeginger GJ (1997) A polynomial-time approximation scheme for maximizing the minimum machine completion time. Oper Res Lett 20:149–154