Journal of Scheduling (2020) 23:163–176
https://doi.org/10.1007/s10951-018-0597-6

The Longest Processing Time rule for identical parallel machines revisited

Federico Della Croce[1,2] · **Rosario Scatamacchia**[1]

Published online: 18 December 2018

Abstract
We consider the $P_m||C_{\max}$ scheduling problem where the goal is to schedule n jobs on m identical parallel machines ($m < n$) to minimize makespan. We revisit the famous Longest Processing Time (LPT) rule proposed by Graham in 1969. LPT requires to sort jobs in non-ascending order of processing times and then to assign one job at a time to the machine whose load is smallest so far. We provide new insights into LPT and discuss the approximation ratio of a modification of LPT that improves Graham's bound from $\left(\frac{4}{3} - \frac{1}{3m}\right)$ to $\left(\frac{4}{3} - \frac{1}{3(m-1)}\right)$ for $m \geq 3$ and from $\frac{7}{6}$ to $\frac{9}{8}$ for $m = 2$. We use linear programming to analyze the approximation ratio of our approach. This performance analysis can be seen as a valid alternative to formal proofs based on analytical derivation. Also, we derive from the proposed approach an $O(n \log n)$ time complexity heuristic. The heuristic splits the sorted job set in tuples of m consecutive jobs $(1, \ldots, m; m+1, \ldots, 2m;$ etc.) and sorts the tuples in non-increasing order of the difference (slack) between largest job and smallest job in the tuple. Then, given this new ordering of the job set, list scheduling is applied. This approach strongly outperforms LPT on benchmark literature instances and is competitive with more involved approaches such as COMBINE and LDM.

Keywords Identical parallel machine scheduling · LPT rule · Linear programming · Approximation algorithms

1 Introduction

We consider the $P_m||C_{\max}$ scheduling problem [as denoted in the three-field classification by Graham et al. (1979)] where the goal is to schedule n jobs on m identical parallel machines M_i $(i = 1, \ldots, m)$ to minimize the makespan. $P_m||C_{\max}$ is strongly NP-hard (Garey and Johnson 1979) and has been intensively investigated in the literature both from a theoretical and a practical point of view. For an exhaustive discussion, we refer, among others, to books by Leung et al. (2004), Pinedo (2016) and to the comprehensive survey by Chen et al. (1999). The pioneering approximation algorithm for the problem is the Longest Processing Time (LPT) rule proposed by Graham (1969). It requires to sort the jobs in non-ascending order of their processing times p_j $(j = 1, \ldots, n)$ and then, given the sorted job set, to assign one job at a time to the machine whose load is smallest so far. This assignment of jobs to machines is also known as list scheduling (LS). Several properties have been established for LPT in the last decades (Graham 1969; Coffman Jr. and Sethi 1976; Chen 1993; Blocher and Sevastyanov 2015). Among the various approaches explicitly based on the LPT rule, we mention also Dosa (2004), Dosa and Vizvari (2006). We recall the main theoretical results for LPT in the next section. LPT generally exhibits much better performance in practice than the expected theoretical ratios, especially as the number of jobs gets larger. Frenk and Rinnooy Kan (1987) also showed that LPT is asymptotically optimal under mild conditions on the probability distribution of the job processing times. Due to its simplicity and practical effectiveness, LPT became a cornerstone for the design of more involving exact or heuristic algorithms.

We mention other popular approximation algorithms which exploit connections of $P_m||C_{\max}$ with bin packing: MULTIFIT (Coffman Jr. et al. 1978), COMBINE (Lee and Massey 1988) and LISTFIT (Gupta and Ruiz-Torres 2001). Such algorithms provide better worst-case performance than LPT

✉ Federico Della Croce
federico.dellacroce@polito.it

Rosario Scatamacchia
rosario.scatamacchia@polito.it

[1] Dipartimento di Ingegneria Gestionale e della Produzione, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Turin, Italy

[2] CNR, IEIIT, Turin, Italy

but at the cost of higher running times. More recently, other more involved heuristics are available (see, e.g., Paletta and Ruiz-Torres 2015) that require, however, much higher computational effort. In addition, we mention the Largest Differencing Method (LDM) proposed by Karmarkar and Karp (1982), an efficient polynomial-time algorithm for which Michiels et al. (2007) showed that an approximation ratio not superior to the LPT ratio holds. Also, polynomial-time approximation schemes (PTASs) were derived for the problem. The first PTAS was given by Hochbaum and Shmoys (1987). PTASs with improved running times were then provided by Alon et al. (1998), Hochbaum (1997), Jansen (2010). Recently, an improved PTAS has been proposed by Jansen et al. (2017).

The contribution of this work is twofold. First, we revisit the LPT rule and provide a simple algorithmic variant that manages to improve the long-standing approximation ratio derived by Graham (1969) keeping the same computational complexity. To establish our theoretical results, we also employ linear programming (LP) to analyze the worst-case performance of the proposed algorithm and to derive approximation bounds. In a sense, this paper can also be seen as a follow-up of the work of Mireault et al. (1997) where several LPs were used to determine the worst-case approximation ratio of LPT on two uniform machines. Recently a growing attention has been paid to the use of LP modeling for the derivation of formal proofs (see Chimani and Wiedera 2016; Abolhassani et al. 2016; Della Croce et al. 2018) and we also show here a successful application of this technique.

We then move from approximation to heuristics. By generalizing the proposed LPT-based approach, we obtain a simple algorithm running in $\mathcal{O}(n \log n)$ time which drastically improves upon LPT and can hence be regarded as a valuable alternative to the most popular constructive heuristic designed for this problem.

2 Notation and LPT properties

We first recall the main theoretical properties of LPT applied to $P_m||C_{\max}$. From now on, we will consider the job set $\mathcal{J} = \{1, \ldots, n\}$ sorted by non-increasing p_j ($p_j \geq p_{j+1}, \quad j = 1, \ldots, n-1$). We denote the solution values of the LPT schedule and the optimal makespan by $C_m^{LPT}(\mathcal{J})$ and $C_m^*(\mathcal{J})$, respectively, where index m indicates the number of machines. Also, similarly to Chen (1993), we denote by r_l^{LPT} the performance ratio $\frac{C_m^{LPT}(\mathcal{J})}{C_m^*(\mathcal{J})}$ of an LPT schedule with l jobs assigned to the machine yielding the completion time (the critical machine) and by j' the last job assigned to the machine that gives the corresponding makespan (the critical job). As usually employed in the worst-case performance analysis of LPT, from now on we assume that the critical job is the last one, namely $j' = n$. Otherwise, we would have other jobs scheduled after the critical job that do not affect the makespan provided by LPT but can contribute to increase the optimal solution value. We summarize hereafter bounds on $C_m^*(\mathcal{J})$ and properties of the LPT schedule available in the literature.

Proposition 1 (Pinedo 2016) *The following expressions hold:*

$$C_m^*(\mathcal{J}) \geq \max \left\{ \frac{\sum_{j=1}^{n} p_j}{m}, p_1 \right\}; \tag{1}$$

$$C_m^{LPT}(\mathcal{J}) = C_m^*(\mathcal{J}) \text{ if } p_{j'} > \frac{C_m^*(\mathcal{J})}{3}; \tag{2}$$

otherwise, i.e., if $p_{j'} \leq \frac{C_m^*(\mathcal{J})}{3}$,

$$\begin{aligned} C_m^{LPT}(\mathcal{J}) &\leq \frac{\sum_{j=1}^{j'} p_j}{m} + p_{j'}\left(1 - \frac{1}{m}\right) \\ &\leq C_m^*(\mathcal{J}) + p_{j'}\left(1 - \frac{1}{m}\right). \end{aligned} \tag{3}$$

Proposition 2 (Chen 1993) *For each job i assigned by LPT in position j on a machine, the following inequality holds*

$$p_i \leq \frac{C_m^*(\mathcal{J})}{j}. \tag{4}$$

Proposition 3 *The following tight approximation ratios hold for LPT:*

$$r_2^{LPT} \leq \frac{4}{3} - \frac{1}{3(m-1)}; \quad \textit{(Chen 1993)} \tag{5}$$

$$r_l^{LPT} \leq \frac{l+1}{l} - \frac{1}{lm} \quad l \geq 3. \quad \textit{(Coffman Jr. and Sethi 1976)} \tag{6}$$

Approximation bounds (6) were derived by Coffman Jr. and Sethi (1976) and are also known as the a posteriori generalized bounds. For $l = 3$, the corresponding approximation ratio of $\frac{4}{3} - \frac{1}{3m}$ is the well-known Graham's bound (Graham 1969) and constitutes the worst-case bound for LPT. A straightforward implication of Proposition 2 is the following:

Lemma 1 *If LPT provides a schedule where a non-critical machine processes at least k jobs before the critical job j', then* $\frac{C_m^{LPT}(\mathcal{J})}{C_m^*(\mathcal{J})} \leq \frac{k+1}{k} - \frac{1}{km}$.

Proof Denote by i the job in the kth position on the non-critical machine, with $p_i \geq p_{j'}$. Since we have $p_i \leq \frac{C_m^*(\mathcal{J})}{k}$ from Proposition 2 and $p_{j'} \leq \frac{C_m^*(\mathcal{J})}{k}$ also holds, we get from inequality (3) that $C_m^{LPT}(\mathcal{J}) \leq (\frac{k+1}{k} - \frac{1}{km}) C_m^*(\mathcal{J})$. □

3 LPT revisited

3.1 Results for LPT

We provide here further insights into the Longest Processing Time rule. We first elaborate on the approximation bound provided in Lemma 1. We state the following proposition.

Proposition 4 *If LPT schedules at least k jobs on a non-critical machine before assigning the critical job, then $\frac{C_m^{LPT}(\mathcal{J})}{C_m^*(\mathcal{J})} \leq \frac{k+1}{k} - \frac{1}{k(m-1)}$ for $m \geq k+2$.*

Proof First, we can assume that the critical machine processes at least two jobs; otherwise, the LPT solution would be optimal as $C_m^*(\mathcal{J}) \geq p_1$ due to Proposition 1. Also, due to Proposition 2, condition $p_n \leq \frac{C_m^*(\mathcal{J})}{k}$ holds. Denote by t_c the completion time of the critical machine before loading critical job n. We have $C_m^{LPT}(\mathcal{J}) = t_c + p_n$. Also, denote by t' the completion time of a non-critical machine processing at least k jobs and by t'' the sum of completion times of the other $(m-2)$ machines, namely $t'' = \sum_{j=1}^{n} p_j - t' - (t_c + p_n)$. Since the application of list scheduling to the sorted jobs, each of the $(m-2)$ machines must have a completion time not inferior to t_c. Hence, the following inequality holds

$$\frac{t''}{m-2} \geq t_c. \tag{7}$$

We now rely on linear programming to evaluate the worst-case performance ratio of LPT for any $k \geq 1$ and $m \geq k+2$. More precisely, we introduce an LP formulation where we can arbitrarily set the value $C_m^{LPT}(\mathcal{J})$ to 1 and minimize the value of $C_m^*(\mathcal{J})$. We associate nonnegative variables sum_p and opt with $\sum_{j=1}^{n} p_j$ and $C_m^*(\mathcal{J})$, respectively. We also consider completion times t_c, t', t'' and processing time p_n as nonnegative variables in the LP model. Since we have $p_n \leq \frac{C_m^*(\mathcal{J})}{k}$, we introduce an auxiliary slack variable sl to write the corresponding constraint in the LP model as $p_n + sl - \frac{opt}{k} = 0$. This implies the following LP model parametric in m and k:

$$\text{minimize} \quad opt \tag{8}$$
$$\text{subject to} \quad -m \cdot opt + sum_p \leq 0 \tag{9}$$
$$k \cdot p_n - t' \leq 0 \tag{10}$$
$$t_c - t' \leq 0 \tag{11}$$
$$(m-2)t_c - t'' \leq 0 \tag{12}$$
$$(t_c + p_n) + t' + t'' - sum_p = 0 \tag{13}$$
$$t_c + p_n = 1 \tag{14}$$
$$p_n + sl - \frac{opt}{k} = 0 \tag{15}$$
$$t_c, t', t'', p_n, sum_p, opt, sl \geq 0 \tag{16}$$

The minimization of the objective function (8), after setting without loss of generality the LPT solution value to 1 (constraint (14)), provides an upper bound on the performance ratio of LPT rule. Constraint (9) represents the bound $C_m^*(\mathcal{J}) \geq \frac{\sum_{j=1}^{n} p_j}{m}$, while constraint (10) states that the value of t' is at the least kp_n, since k jobs with a processing time not inferior to p_n are assigned to the non-critical machine. Constraint (11) states that the completion time of the critical machine before the execution of the last job is not superior to the completion time of the other machine processing at least k jobs. Constraint (12) fulfills inequality (7). Constraint (13) guarantees that variable sum_p corresponds to $\sum_{j=1}^{n} p_j$ and constraint (15) represents condition $p_n \leq \frac{C_m^*(\mathcal{J})}{k}$. Eventually, constraints (16) state that all variables are nonnegative. Note that model (8)–(16) is independent from the number of jobs n. By analyzing the solutions of model (8)–(16) for several given values of m and k, we were able to deduce that a feasible solution of the model, for any $k \geq 1$ and $m \geq 2$, is:

$$t_c = \frac{k(m-1)-1}{(k+1)m-k-2};\ p_n = \frac{m-1}{(k+1)m-k-2};$$
$$t' = \frac{k(m-1)}{(k+1)m-k-2};\ t'' = \frac{(m-2)(k(m-1)-1)}{(k+1)m-k-2};$$
$$opt = \frac{k(m-1)}{(k+1)m-k-2};\ sum_p = \frac{m(m-1)k}{(k+1)m-k-2};$$
$$sl = 0.$$

We can show by strong duality that such a solution is in fact optimal for any $m \geq k+2$. Plugging $p_n = \frac{opt}{k} - sl$ from constraint (15) in constraints (13) and (14), we get an equivalent reduced LP model. If we associate dual variables λ_i $(i = 1, \ldots, 6)$ with constraints (9)–(14), respectively, the corresponding dual formulation of the reduced problem is as follows:

$$\text{maximize} \quad \lambda_6 \tag{17}$$
$$\text{subject to} \quad -m\lambda_1 + \lambda_2 + \frac{\lambda_5 + \lambda_6}{k} \leq 1 \tag{18}$$
$$\lambda_1 - \lambda_5 \leq 0 \tag{19}$$
$$-k \cdot \lambda_2 - \lambda_5 - \lambda_6 \leq 0 \tag{20}$$
$$-\lambda_2 - \lambda_3 + \lambda_5 \leq 0 \tag{21}$$
$$\lambda_3 + (m-2)\lambda_4 + \lambda_5 + \lambda_6 \leq 0 \tag{22}$$
$$-\lambda_4 + \lambda_5 \leq 0 \tag{23}$$
$$\lambda_1, \lambda_2, \lambda_3, \lambda_4 \leq 0 \tag{24}$$

where the dual constraints (18)–(23) are related to the primal variables opt, sum_p, sl, t', t_c, t'', respectively. For any $m \geq k+2$, a feasible solution of model (17)–(24) is

$$\lambda_1 = \lambda_2 = \lambda_4 = \lambda_5 = \frac{-k}{(k+1)m - k - 2};$$
$$\lambda_3 = 0;\ \lambda_6 = \frac{k(m-1)}{(k+1)m - k - 2};$$

where condition $m \geq k+2$ is necessary to satisfy constraint (20). Since $opt = \lambda_6$ in the above solutions, by strong duality these solutions are both optimal. We hence have

$$\frac{C_m^{LPT}(\mathcal{J})}{C_m^*(\mathcal{J})} \leq \frac{1}{opt} = \frac{(k+1)m - k - 2}{k(m-1)} = \frac{(k+1)(m-1) - 1}{k(m-1)} = \frac{k+1}{k} - \frac{1}{k(m-1)}$$

which shows the claim. □

Notably, with respect to Lemma 1, the result of Proposition 4 for $k = 3$ provides already a better bound than Graham's bound and equal to $\frac{4}{3} - \frac{1}{3(m-1)}$ for $m \geq 5$. Also, with the results of Proposition 4 we can state the following result.

Proposition 5 *In $P_m||C_{\max}$ instances with $n \geq 2m+2$ and $m \geq 5$, LPT has an approximation ratio not superior to $\frac{4}{3} - \frac{1}{3(m-1)}$.*

Proof In instances with $n \geq 3m+1$, there exists at least one machine in the LPT schedule executing at least four jobs. Then, either the machine is the critical one or not. Correspondingly, either inequality (6) with $l \geq 4$ holds or Proposition 4 with $k \geq 3$ applies showing the above claim. Consider the remaining cases with $2m+2 \leq n \leq 3m$. We assume the critical job n in third position on a machine; otherwise, either bound on r_2^{LPT} holds or at least bound on r_4^{LPT} holds. This implies that LPT schedules at least another job in position ≥ 3 on a non-critical machine. Hence, the results of Proposition 4 with $k = 3$ apply. □

In instances with $n \geq 2m+2$ and $3 \leq m \leq 4$, we can combine the reasoning underlying model (8)–(16) with a partial enumeration of the optimal/LPT solutions and state the following result.

Proposition 6 *In $P_m||C_{\max}$ instances with $n \geq 2m+2$, LPT has an approximation ratio not superior to $\frac{4}{3} - \frac{1}{3(m-1)}$) for $3 \leq m \leq 4$.*

Proof See "Appendix." □

Consider now instances with $2m$ jobs at most. The following proposition holds.

Proposition 7 *In $P_m||C_{\max}$ instances with $n \leq 2m$, LPT has an approximation ratio not superior to $\frac{4}{3} - \frac{1}{3(m-1)}$.*

Proof We consider the case $n = 2m$ only. All other cases $n < 2m$ can be reduced to the previous one after adding $2m - n$ dummy jobs with null processing time.
It is well known (see, e.g., Graham 1969) that if each machine processes two jobs at most in an optimal schedule, the solution provided by LPT would be optimal. Hence, we consider the case where there is one machine processing at least three jobs in an optimal solution. This situation straightforwardly implies that job 1 has to be processed alone on a machine. Therefore, we have $C_m^*(\mathcal{J}) \geq C_{m-1}^*(\mathcal{J} \setminus \{1\})$ since the optimal makespan with m machines could be as well given by the machine processing only job 1.
On the other hand, to contradict the claim, LPT must have the critical machine processing more than two jobs; otherwise, we could use bound (5). This implies that job 1 is processed alone on a machine and cannot give the makespan; otherwise, LPT solution would be optimal due to expression (1). We thus have $C_m^{LPT}(\mathcal{J}) = C_{m-1}^{LPT}(\mathcal{J} \setminus \{1\})$. Combining these results with Graham's bound on the problem instance with $m-1$ machines and without job 1, we get

$$\frac{C_m^{LPT}(\mathcal{J})}{C_m^*(\mathcal{J})} \leq \frac{C_{(m-1)}^{LPT}(\mathcal{J} \setminus \{1\})}{C_{(m-1)}^*(\mathcal{J} \setminus \{1\})} \leq \frac{4}{3} - \frac{1}{3(m-1)}.$$

□

For instances with exactly $2m+1$ jobs, we provide the following proposition.

Proposition 8 *In instances with $n = 2m+1$, if LPT loads at least three jobs on a machine before the critical job, then the approximation ratio is not superior to $\frac{4}{3} - \frac{1}{3(m-1)}$.*

Proof If LPT schedules at least three jobs on a machine before critical job n, this means that job 1 is processed either alone on a machine or with critical job n only. In the latter case, the claim is showed through bound (5). Alternatively, job 1 is processed alone on machine M_1. Also, M_1 cannot give the makespan; otherwise, LPT would yield an optimal solution. This implies that $C_m^{LPT}(\mathcal{J}) = C_{m-1}^{LPT}(\mathcal{J} \setminus \{1\})$ and that a trivial upper bound on the LPT solution value is equal to $p_1 + p_n$ as the starting time of job n is not superior to p_1. In this case, if an optimal solution schedules job 1 with another job, we have $C_m^*(\mathcal{J}) \geq p_1 + p_n$, and thus, LPT would also give the optimal makespan. If an optimal solution schedules job 1 alone on M_1, then inequality $C_m^*(\mathcal{J}) \geq C_{m-1}^*(\mathcal{J} \setminus \{1\})$ holds. Combining these results with Graham's bound as in Proposition 7, we have

$$\frac{C_m^{LPT}(\mathcal{J})}{C_m^*(\mathcal{J})} \leq \frac{C_{(m-1)}^{LPT}(\mathcal{J} \setminus \{1\})}{C_{(m-1)}^*(\mathcal{J} \setminus \{1\})} \leq \frac{4}{3} - \frac{1}{3(m-1)}.$$

□

Summarizing the above properties, the following theorem holds.

Theorem 1 *LPT has an approximation ratio not superior to* $\frac{4}{3}-\frac{1}{3(m-1)}$ *in all instances with* $m \geq 3$ *and* $n \neq 2m+1$. *Also, LPT can actually hit Graham's bound of* $\frac{4}{3}-\frac{1}{3m}$ *for* $m \geq 2$ *only in instances with* $2m+1$ *jobs with the critical machine processing three jobs and all the other machines processing two jobs.*

3.2 Improving the LPT bound: Algorithm *LPT-REV*

We consider a slight algorithmic variation in LPT where a subset of the sorted jobs is first loaded on M_1, and then, LPT is applied to the remaining job set on all machines (including M_1). We denote this variant as $LPT(\mathcal{S})$ where $\mathcal{S}$ represents the set of jobs assigned altogether to a machine first. Consider the following algorithm, denoted by *LPT-REV*, which first applies LPT and then $LPT(\mathcal{S})$ for at most two suitable choices of the job set $\mathcal{S}$.

Algorithm *LPT-REV*

Input: $P_m||C_{max}$ instance with n jobs and m machines.

1: Apply LPT yielding a schedule with makespan z_1 and $k-1$ jobs on the critical machine before job j'.
2: Apply $LPT' = LPT(\{j'\})$ with solution value z_2.
3: **If** $m = 2$ **then** apply $LPT'' = LPT(\{(j'-k+1), \ldots, j'\})$ with solution value z_3 and return $\min\{z_1, z_2, z_3\}$.
4: **Else** return $\min\{z_1, z_2\}$.

In practice, *LPT-REV* algorithm applies LPT first and then re-applies LPT *after* having loaded on a machine first either its critical job j' or the tuple of the smallest k jobs $(j'-k+1), \ldots, j'$. (this latter case applies only for $m = 2$). Eventually, the algorithm takes the best of the computed solutions.

In the following, we will show how to improve the long-standing Graham's bound by means of algorithm *LPT-REV* for $m \geq 3$. To this extent, given Theorem 1, we just need to address the case where the LPT critical job j' is job $2m+1$. So we consider instances with either $j' = 2m+1 < n$ or $j' = 2m+1 = n$. The LPT schedules with $j' = 2m+1 < n$ will be considered in Sect. 3.2.1 by means of Proposition 9 and additional remarks. The LPT schedules with $j' = 2m+1 = n$ will be covered in Sect. 3.2.2 by means of Propositions 10–14. Finally, the approximation ratio of *LPT-REV* will be stated in Sect. 3.2.3 in Theorem 2.

3.2.1 LPT schedules with $j' = 2m+1 < n, m \geq 3$

If there exists a job $i > j'$ that is critical in the solution provided by LPT', the following proposition holds.

Proposition 9 *In* $P_m||C_{\max}$ *instances where there exists a job* i *such that* $j' < i \leq n$ *and* i *is critical in* LPT' *schedule, LPT-REV algorithm has a performance guarantee of* $\frac{4}{3}-\frac{7m-4}{3(3m^2+m-1)}$.

Proof Denote by $\beta \sum_{j=1}^{n} p_j$ the overall processing time of jobs $j'+1, \ldots, n$, with $0 < \beta < 1$. Due to Graham's bound, the following relation holds for LPT when only jobs $1, \ldots, j'$ are considered:

$$C_m^{LPT}(\mathcal{J}) \leq \frac{\sum_{j=1}^{j'} p_j}{m}\left(\frac{4}{3}-\frac{1}{3m}\right) = \frac{(1-\beta)\sum_{j=1}^{n} p_j}{m}\left(\frac{4}{3}-\frac{1}{3m}\right). \tag{25}$$

From (25), we have

$$\frac{C_m^{LPT}(\mathcal{J})}{C_m^*(\mathcal{J})} \leq \frac{C_m^{LPT}(\mathcal{J})}{\frac{\sum_{j=1}^{n} p_j}{m}} \leq (1-\beta)\left(\frac{4}{3}-\frac{1}{3m}\right). \tag{26}$$

We introduce a target LPT approximation ratio denoted as ρ and identify the value of β which gives such a bound. We have

$$(1-\beta)\left(\frac{4}{3}-\frac{1}{3m}\right) = \rho \implies \beta = 1-\frac{3m\rho}{4m-1}. \tag{27}$$

Consider now the solution provided by LPT' and denote by $t_{c'}$ the processing time of the jobs on the critical machine that are processed before the critical job i. Since the following relations hold

$$mt_{c'} + p_i \leq \sum_{j=1}^{n} p_j \leq mC_m^* \implies t_{c'} + \frac{p_i}{m} \leq C_m^*;$$

$$p_i \leq \beta \sum_{j=1}^{n} p_j \leq m\beta C_m^*,$$

we have, in combination with (27), that

$$\begin{aligned} C_m^{LPT'}(\mathcal{J}) &= t_{c'} + p_i = \left(t_{c'} + \frac{p_i}{m}\right) + p_i\left(1-\frac{1}{m}\right) \\ &\leq C_m^*(\mathcal{J}) + (m-1)\beta C_m^*(\mathcal{J}) \\ &\implies \frac{C_m^{LPT'}(\mathcal{J})}{C_m^*(\mathcal{J})} \leq 1 + (m-1)\beta \\ &= 1 + (m-1)\left(1-\frac{3m\rho}{4m-1}\right). \end{aligned} \tag{28}$$

Hence, algorithm *LPT-REV* has a performance guarantee equal to $\min\{1 + (m-1)(1 - \frac{3m\rho}{4m-1}); \rho\}$. This expression reaches its largest value when the two terms are equal, namely

$$1 + (m-1)\left(1 - \frac{3m\rho}{4m-1}\right) = \rho. \tag{29}$$

From condition (29), we derive

$$\rho = \frac{4m-1}{1+3m-\frac{1}{m}} = \frac{4}{3} - \frac{7m-4}{3(3m^2+m-1)} \geq \frac{C_m^{LPT\text{-}REV}(\mathcal{J})}{C_m^*(\mathcal{J})}.$$

□

It easy to check that the bound of Proposition 9 is strictly inferior to $\frac{4}{3} - \frac{1}{3(m-1)}$ for $m \geq 3$. If the critical job in LPT' schedule is a job $h \leq j'$, then *LPT-REV* cannot have a worse approximation ratio than the one reached in a reduced instance where all jobs after j' are discarded and the same analysis of Sect. 3.2.2 applies.

3.2.2 LPT schedules with $j' = 2m + 1 = n, m \geq 3$

We show here that the addition of LPT' allows us to improve Graham's bound to $\frac{4}{3} - \frac{1}{3(m-1)}$. We concentrate on instances with $2m+1$ jobs where LPT must couple jobs $1, \ldots, m$, respectively, with jobs $2m, \ldots, m+1$ on the m machines before scheduling job $2m+1$; otherwise, Theorem 1 would hold. Therefore, we will consider the following LPT schedules with pair of jobs on each machine M_i $(i = 1, \ldots, m)$:

$M_1 : p_1, p_{2m}$
$M_2 : p_2, p_{2m-1}$
$\ldots$
$M_{m-1} : p_{m-1}, p_{m+2}$
$M_m : p_m, p_{m+1}$

where job $2m+1$ will be assigned to the machine with the least completion time. We analyze the following two subcases.

Case 1: $p_{(2m+1)} \geq p_1 - p_m$

In this case, the last job $2m+1$ has a processing time greater than (or equal to) the difference $p_1 - p_m$. Consider LPT' heuristic. Since the LPT critical job is assumed to be the last one, i.e., $j' = 2m+1$, the heuristic will assign jobs $2m+1, 1, \ldots, m-1$ to machines $M_1, M_2, \ldots M_m$, respectively. Then, job m will be loaded on M_1 together with job $2m+1$. Since $p_{(2m+1)} + p_m \geq p_1$, job $m+1$ will be processed on the last machine M_m after job $m-1$. Now we have

$$p_{(m-1)} + p_{(m+1)} \geq p_{(2m+1)} + p_m \geq p_1$$

since $p_{(m-1)} \geq p_m$ and $p_{(m+1)} \geq p_{(2m+1)}$. Hence, job $m+2$ is loaded on machine $M_{(m-1)}$ with job $m-2$. Similarly as before, it follows that $p_{(m-2)} + p_{(m+2)} \geq p_{(2m+1)} + p_m$. Consequently, job $m+3$ is processed on $M_{(m-2)}$ after job $m-3$. By applying the same argument, LPT' will assign the second job in reversed order to each machine until job $2m-1$ is assigned to M_2. Eventually, job $2m$ will be assigned to M_1 since it will be the least loaded machine at that point. This because all other machines have the largest (resp. smallest) job with processing time not inferior to p_m (resp. $p_{(2m+1)}$). Summarizing, LPT' will provide the following schedule:

$M_1 : p_{(2m+1)}, p_m, p_{2m}$
$M_2 : p_1, p_{(2m-1)}$
$M_3 : p_2, p_{(2m-2)}$
$\ldots$
$M_{(m-1)} : p_{(m-2)}, p_{(m+2)}$
$M_m : p_{(m-1)}, p_{(m+1)}$

Assume now that the critical machine is M_1 with non-optimal completion time $C_m^{LPT'}(\mathcal{J}) = p_{(2m+1)} + p_m + p_{2m} > C_m^*(\mathcal{J})$ (or else *LPT-REV* would provide the optimal solution). The following proposition holds:

Proposition 10 *If $p_{(2m+1)} \geq p_1 - p_m$ and $C_m^{LPT'}(\mathcal{J}) = p_{(2m+1)} + p_m + p_{2m} > C_m^*(\mathcal{J})$, then $C_m^*(\mathcal{J}) \geq p_{(m-1)} + p_m$ in any optimal schedule.*

Proof We prove the claim by contradiction. We assume that an optimal schedule assigns jobs $1, 2, \ldots, m$ to different machines or else $C_m^*(\mathcal{J}) \geq p_{(m-1)} + p_m$ immediately holds. Correspondingly, since there exists a machine processing three jobs, the optimal makespan can be lower bounded by $p_m + p_{2m} + p_{(2m+1)}$. But as $p_m + p_{2m} + p_{(2m+1)} > C_m^*(\mathcal{J})$ holds, a contradiction on the optimality of the schedule is implied. □

The following proposition also holds.

Proposition 11 *If $p_{(2m+1)} \geq p_1 - p_m$ and $C_m^{LPT'}(\mathcal{J}) = p_{(2m+1)} + p_m + p_{2m}$, then algorithm LPT-REV has an approximation ratio not superior to $\frac{7}{6}$.*

Proof We again employ linear programming to evaluate the performance of LPT'. More precisely, we consider an LP formulation with nonnegative variables p_j $(j = 1, \ldots, n)$ denoting the processing times and a positive parameter $OPT > 0$ associated with $C_m^*(\mathcal{J})$. The corresponding LP model for evaluating the worst-case performance of LPT' heuristic is as follows:

$$\text{maximize} \quad p_{(2m+1)} + p_m + p_{2m} \tag{30}$$
$$\text{subject to} \quad p_{(m-1)} + p_m \leq OPT \tag{31}$$

$$p_{(2m-1)} + p_{2m} + p_{(2m+1)} \leq OPT \quad (32)$$
$$p_{(2m+1)} - (p_1 - p_m) \geq 0 \quad (33)$$
$$p_1 - p_{(m-1)} \geq 0 \quad (34)$$
$$p_{(m-1)} - p_m \geq 0 \quad (35)$$
$$p_m - p_{(m+1)} \geq 0 \quad (36)$$
$$p_{(m+1)} - p_{(2m-1)} \geq 0 \quad (37)$$
$$p_{(2m-1)} - p_{2m} \geq 0 \quad (38)$$
$$p_{2m} - p_{(2m+1)} \geq 0 \quad (39)$$
$$p_1, p_{(m-1)}, p_m, p_{(m+1)}, p_{(2m-1)}, p_{2m}, p_{(2m+1)} \geq 0 \quad (40)$$

The objective function value (30) represents an upper bound on the worst-case performance of the algorithm. Constraints (31)–(32) state that the optimal value $C_m^*(\mathcal{J})$ is lower bounded according to Proposition 10 and by the sum of the three jobs with the smallest processing times. Constraint (33) simply represents the initial assumption $p_{(2m+1)} \geq p_1 - p_m$. Constraints (34)–(39) state that the considered relevant jobs are sorted by non-increasing processing times, while constraints (40) indicate that the variables are nonnegative. We remark that parameter OPT can be arbitrarily set to any value > 0. Further valid inequalities (such as $p_{(2m+1)} + p_m + p_{2m} \geq p_1 + p_{(2m-1)}$ or $OPT \geq \frac{\sum_{j=1}^{n} p_j}{m}$) were omitted as they do not lead to any improvement on the worst-case performance ratio. Notice that the number of variables of the reduced LP formulation is constant for any value of m.

By setting w.l.o.g. $OPT = 1$ and solving model (30)–(40), we get an optimal solution value z^* equal to $1.1666\ldots = \frac{7}{6}$. Correspondingly, the approximation ratio is $\frac{z^*}{OPT} = \frac{7}{6}$. □

Consider now the case where the makespan of LPT' schedule is given by one of the machines $M_2, \ldots, M_m$. In such a case, a trivial upper bound on LPT' makespan is equal to $p_1 + p_{m+1}$. We state the following proposition.

Proposition 12 *If $p_{(2m+1)} \geq p_1 - p_m$ and the makespan of LPT' is not on M_1, then coupling LPT with LPT' gives a performance guarantee not superior to $\frac{15}{13}$ for $m = 3$ and $\frac{4}{3} - \frac{1}{2m-1}$ for $m \geq 4$.*

Proof We again consider an LP formulation with nonnegative variables p_j $(j = 1, \ldots, n)$, a positive parameter $OPT > 0$ and two nonnegative auxiliary variables α, y. We can evaluate the worst-case performance of $LPT + LPT'$ by the following LP model

$$\text{maximize} \quad y \quad (41)$$
$$\text{subject to} \quad \sum_{j=1}^{2m+1} p_j \leq m\,OPT \quad (42)$$
$$p_{(2m-1)} + p_{2m} + p_{(2m+1)} \leq OPT \quad (43)$$
$$p_{(j+1)} - p_j \leq 0 \quad j = 1, \ldots, 2m; \quad (44)$$
$$p_j + p_{(2m-j+1)} - \alpha \geq 0 \quad j = 1, \ldots, m; \quad (45)$$
$$p_{(2m+1)} + \alpha - y \geq 0 \quad (46)$$
$$p_{(2m+1)} - (p_1 - p_m) \geq 0 \quad (47)$$
$$p_1 + p_{(m+1)} - y \geq 0 \quad (48)$$
$$p_j \geq 0 \quad j = 1, \ldots, 2m+1; \quad (49)$$
$$\alpha, y \geq 0 \quad (50)$$

where y represents the solution value reached by $LPT + LPT'$ and α is the starting time of job $2m+1$ in LPT. Constraints (45) indicate that α corresponds to the smallest load on a machine after processing jobs $1, \ldots, 2m$. The solution value of LPT is therefore the sum $\alpha + p_{(2m+1)}$. The objective function (41) provides an upper bound on the makespan of $LPT + LPT'$ since it maximizes the minimum between $\alpha + p_{(2m+1)}$ and the makespan reached by LPT' through variable y and related constraints (46) and (48). Constraints (42) and (43) state that $C_m^*(\mathcal{J})$ is lower bounded by $\frac{\sum_{j=1}^{n} p_j}{m}$ and by $p_{(2m-1)} + p_{2m} + p_{(2m+1)}$. Constraints (44) indicate that jobs are sorted by non-increasing processing time, while constraint (47) represents condition $p_{(2m+1)} \geq p_1 - p_m$. Finally, constraints (49) and (50) indicate that all variables are nonnegative.

By setting w.l.o.g. $OPT = 1$, a feasible solution of model (41)–(50) for any value of m is:

$$y = \frac{8m-7}{3(2m-1)}; \quad \alpha = \frac{2(m-1)}{2m-1};$$
$$p_1 = \frac{5m-4}{3(2m-1)}; \quad p_2 = p_3 = \cdots = p_{(m-1)} = \frac{4m-5}{3(2m-1)};$$
$$p_m = p_{m+1} = \frac{m-1}{2m-1};$$
$$p_{m+2} = p_{m+3} = \cdots = p_{2m+1} = \frac{1}{3}.$$

We can show by strong duality that this solution is optimal for any $m \geq 4$. The dual model with variables λ_i $(i = 1, \ldots, 3m+5)$ associated with constraints (42)–(48) is as follows:

$$\text{minimize} \quad m\lambda_1 + \lambda_2 \quad (51)$$
$$\text{subject to} \quad \lambda_1 - \lambda_3 + \lambda_{(2m+3)} - \lambda_{(3m+4)} + \lambda_{(3m+5)} \geq 0 \quad (52)$$
$$\lambda_1 + \lambda_{(1+j)} - \lambda_{(2+j)} + \lambda_{(2m+2+j)} \geq 0 \quad j = 2, \ldots, m-1 \quad (53)$$
$$\lambda_1 + \lambda_{(m+1)} - \lambda_{(m+2)} + \lambda_{(3m+2)} + \lambda_{(3m+4)} \geq 0 \quad (54)$$
$$\lambda_1 + \lambda_{(m+2)} - \lambda_{(m+3)} + \lambda_{(3m+2)} + \lambda_{(3m+5)} \geq 0 \quad (55)$$
$$\lambda_1 + \lambda_{(1+j)} - \lambda_{(2+j)} + \lambda_{(4m+3-j)} \geq 0 \quad j = m+2, \ldots, 2m-2 \quad (56)$$
$$\lambda_1 + \lambda_2 + \lambda_{2m} - \lambda_{(2m+1)} + \lambda_{(2m+4)} \geq 0 \quad (57)$$
$$\lambda_1 + \lambda_2 + \lambda_{(2m+1)} - \lambda_{(2m+2)} + \lambda_{(2m+3)} \geq 0 \quad (58)$$
$$\lambda_1 + \lambda_2 + \lambda_{(2m+2)} + \lambda_{(3m+3)} + \lambda_{(3m+4)} \geq 0 \quad (59)$$

$$-\sum_{j=(2m+3)}^{(3m+2)} \lambda_j + \lambda_{(3m+3)} \geq 0 \quad (60)$$

$$-\lambda_{(3m+3)} - \lambda_{(3m+5)} \geq 1 \quad (61)$$

$$\lambda_1, \lambda_2, \ldots, \lambda_{(2m+2)} \geq 0 \quad (62)$$

$$\lambda_{(2m+3)}, \lambda_{(2m+4)}, \ldots \lambda_{(3m+5)} \leq 0 \quad (63)$$

Constraints (52)–(61) correspond to primal variables p_j, α, y, respectively. A feasible solution of model (51)–(63) for $m \geq 4$ is:

$$\lambda_1 = \frac{2}{2m-1};\ \lambda_2 = \frac{2m-7}{3(2m-1)};$$
$$\lambda_3 = \lambda_4 = \cdots = \lambda_{(m+1)} = 0;$$
$$\lambda_{(m+2)} = \frac{1}{2m-1};\ \lambda_{(m+3)} = \lambda_{(m+4)} = \cdots = \lambda_{2m} = 0;$$
$$\lambda_{(2m+1)} = \frac{2m-7}{3(2m-1)};$$
$$\lambda_{(2m+2)} = \frac{4(m-2)}{3(2m-1)};\ \lambda_{(2m+3)} = 0;$$
$$\lambda_{(2m+4)} = \lambda_{(2m+5)} = \cdots = \lambda_{(3m+1)} = \frac{-2}{2m-1};$$
$$\lambda_{(3m+2)} = \frac{-1}{2m-1};\ \lambda_{(3m+3)} = \frac{3-2m}{2m-1};$$
$$\lambda_{(3m+4)} = 0;\ \lambda_{(3m+5)} = \frac{-2}{2m-1}.$$

The corresponding solution value is $m\lambda_1 + \lambda_2 = \frac{8m-7}{3(2m-1)} = y$. Hence, for $m \geq 4$ we have:

$$\frac{\min\{C_m^{LPT}(\mathcal{J}), C_m^{LPT'}(\mathcal{J})\}}{C_m^*(\mathcal{J})} \leq \frac{y}{OPT} = \frac{8m-7}{3(2m-1)} = \frac{4(2m-1)-3}{3(2m-1)} = \frac{4}{3} - \frac{1}{2m-1}$$

For $m = 3$, an optimal solution of model (41)–(50) has value $y = 1.15385\ldots = \frac{15}{13}$.

The corresponding approximation ratio is then $\frac{y}{OPT} = \frac{15}{13}$. □

Case 2: $p_{(2m+1)} < p_1 - p_m$

The processing time of the last job is smaller than the difference $p_1 - p_m$. The following proposition holds.

Proposition 13 *If $p_{(2m+1)} < p_1 - p_m$, the solution given by LPT has a performance guarantee not superior to $\frac{15}{13}$ for $m = 3$ and $\frac{4}{3} - \frac{1}{2m-1}$ for $m \geq 4$.*

Proof We consider the worst-case LPT performance and notice that it can be evaluated through model (41)–(50) by simply reversing the inequality sign in constraint (47) and disregarding constraint (48). Correspondingly, dual model (51)–(63) is still implied with the differences that variable $\lambda_{(3m+5)}$ is discarded and that we have $\lambda_{(3m+4)} \geq 0$. The primal solutions turn out to be the same solutions stated in Proposition 12. Likewise, dual solutions slightly modify in the following variables entries which do not contribute to the objective function: $\lambda_{(3m+2)} = \frac{-3}{2m-1}$; $\lambda_{(3m+3)} = -1$; $\lambda_{(3m+4)} = \frac{2}{2m-1}$. Therefore, the approximation ratios stated in Proposition 12 hold. □

3.2.3 The resulting approximation ratio of *LPT-REV*

We can now state the following theorem for *LPT-REV*.

Theorem 2 *Algorithm LPT-REV runs in $\mathcal{O}(n \log n)$ time and has an approximation ratio not superior to $\frac{4}{3} - \frac{1}{3(m-1)}$ for $m \geq 3$.*

Proof Putting together the results of Propositions 11, 12, 13, it immediately follows that *LPT-REV* has an approximation ratio not superior to $\frac{4}{3} - \frac{1}{3(m-1)}$ for $m \geq 3$. Besides, it is well known that the running time of the LPT heuristic is in $\mathcal{O}(n \log n)$: sorting the jobs by processing time has complexity $\mathcal{O}(n \log n)$, while an efficient implementation of the underlying LS procedure may employ a Fibonacci's heap for extracting the machine with smallest load at each iteration with overall complexity $\mathcal{O}(n \log m)$. Since the proposed algorithm requires to run first LPT (to compute z_1) and then LS heuristic twice (to compute z_2 and z_3) after the job sorting, the resulting time complexity is $\mathcal{O}(n \log n)$. □

We notice that algorithm *LPT-REV* has a worst-case performance at least as good as the one of the more computationally demanding Largest Differencing Method of Karmarkar and Karp (1982), which runs in $\mathcal{O}(n \log n + nm \log m)$ and has an approximation ratio proved to be in the interval $\left[\frac{4}{3} - \frac{1}{3(m-1)}, \frac{4}{3} - \frac{1}{3m}\right]$ for $m \geq 3$ (Michiels et al. 2007).

3.2.4 *LPT-REV* performance analysis for $P_2||C_{\max}$

We remark that in the problem variant with two machines ($m = 2$), the approximation ratio of $\frac{4}{3} - \frac{1}{3(m-1)} = 1$ cannot be reached by any polynomial-time algorithm unless $\mathcal{P} = \mathcal{NP}$, since $P_2||C_{\max}$ is well known to be NP-hard. At the current state of the art, the approach proposed by He et al. (2000) for $P_2||C_{\max}$ is the best approximation algorithm running with linear time complexity and has an approximation bound of $\frac{12}{11}$. Although algorithm *LPT-REV* does not improve this bound, the following analysis shows that the proposed approach reaches an approximation ratio not superior to $\frac{9}{8}$ and thus improves upon the bound of $\frac{7}{6}$ implied for LPT [as well as for Karmarkar–Karp algorithm, as shown by Fischetti and Martello (1987)] when $m = 2$.

We proceed by assuming that the critical job in the LPT solution is the last one ($j' = n$). Otherwise, we would get an approximation ratio of $\frac{14}{13} < \frac{9}{8}$ by using the same reasoning employed in the proof of Proposition 9 taking into account in this case both LPT' and LPT''. Given Lemma 1 and bound (6), an approximation bound worse than $\frac{9}{8}$ could be reached only in instances with $n \leq 6$ and where LPT schedules no more than three jobs on each machine.
For $n \leq 4$, LPT will output an optimal solution according to Proposition 7. For $n = 6$, we can evaluate the worst-case LPT performance by model (64)–(70) (in "Appendix") since $n = 3m$. The corresponding optimal objective value for $m = 2$ is $opt = 0.8888\ldots = \frac{8}{9}$ which gives an approximation ratio equal to $\frac{1}{opt} = \frac{9}{8}$. For $n = 5$, we have the following proposition.

Proposition 14 *In $P_2||C_{\max}$ instances with $n = 5$, LPT-REV provides an optimal solution.*

Proof According to Proposition 8, LPT could not give the optimal makespan only if it assigns jobs 1 and 4 to M_1 and jobs 2 and 3 to M_2 before assigning the critical job 5. Considering this LPT schedule, we analyze all possible optimal solution configurations and show that *LPT-REV* always identifies them. We have the following two cases:

- *Case 1: jobs 1 and 2 are on the same machine in an optimal solution.*
  There exists an optimal solution which assigns jobs 3, 4, 5 to M_1 and jobs 1, 2 to M_2. In fact, any other schedule cannot provide a smaller makespan. The same solution is also provided by LPT''.
- *Case 2: jobs 1 and 2 are on different machines in an optimal solution*.
  If there exists an optimal solution with job 1 on M_1 and jobs 2 and 3 on M_2, such a solution coincides with the LPT schedule. If instead an optimal solution assigns jobs 1 and 3 to M_1, then it must assign jobs 2, 4 and 5 to M_2. If $p_3 = p_4$, clearly LPT also gives the optimal makespan. If $p_3 > p_4$, inequality $p_1 \leq p_2 + p_5$ must hold or else the optimal solution would be improved by exchanging job 3 with job 4 on the machines giving a contradiction. But then, inequality $p_1 \leq p_2 + p_5$ implies that LPT' solution is also optimal since it assigns jobs 5, 2 and 4 to M_1 and jobs 1 and 3 to M_2.

□

4 From approximation to heuristics: a new LPT-based approach

Moving from approximation algorithms to heuristics, we decided to check whether it could possible from *LPT-REV* to come up with a heuristic procedure competitive with the relevant literature with emphasis on algorithms running with very low computational complexity. We considered to this extent the benchmark literature instances provided by Iori and Martello (2008). All tests were conducted on an Intel i5 CPU @ 2.30 GHz with 8 GB of RAM, and all algorithms have been implemented in C++.

Iori and Martello (2008) considered two classical classes of instances from the literature: *uniform instances* proposed by França et al. (1994) and *non-uniform instances* proposed by Frangioni et al. (2004). In *uniform instances*, the processing times are integer uniformly distributed in the range $[a, b]$. In *non-uniform instances*, 98% of the processing times are integer uniformly distributed in $[0.9(b - a), b]$, while the remaining ones are uniformly distributed in $[a, 0.2(b - a)]$. For both classes, we have $a = 1; b = 100, 1000, 10000$. For each class, the following values were considered for the number of machines and jobs: $m = 5, 10, 25$ and $n = 10, 50, 100, 500, 1000$. For each pair (m, n) with $m < n$, 10 instances were generated for a total of 780 instances.
Preliminary testing showed that *LPT-REV* is not significantly superior to LPT, being able to improve only 142 out of 780 instances. Besides, it can be noted that in our theoretical results, LPT' was necessary to improve Graham's bound for $m \geq 3$, while LPT'' was necessary for $m = 2$ only. Remarkably, for $m \geq 3$ the relevant subcase was the one with $2m + 1$ jobs, $p_{2m+1} \geq p_1 - p_m$ and LPT' required to schedule job $2m + 1$ first and then to apply list scheduling first to the sorted job subset $\{1, \ldots, m\}$ and then to the sorted job subset $\{m + 1, \ldots, 2m\}$. This suggests a general greedy approach that considers not only the ordering of the jobs but also the differences in processing time within job subsets of size m. We propose a constructive procedure that splits the sorted job set in tuples of m consecutive jobs $(1, \ldots, m; m + 1, \ldots, 2m$; etc.) and sorts the tuples in non-increasing order of the difference between the largest job and the smallest job in the tuple. Then, list scheduling is applied to the set of sorted tuples. We denote this approach as SLACK.

In terms of computational complexity, since construction and sorting of the tuples can be performed in $\mathcal{O}(\left\lceil \frac{n}{m} \right\rceil \log \left\lceil \frac{n}{m} \right\rceil)$, the running time of SLACK is $\mathcal{O}(n \log n)$ due to the initial sorting of the jobs.
To get a clear picture of the performance of SLACK, we compared it to LPT, to LDM of Karmarkar and Karp (1982) and to COMBINE, the algorithm proposed by Lee and Massey (1988) that couples LPT with the MULTIFIT heuristic introduced by Coffman Jr. et al. (1978). Notice that with an increase in the computational effort, both COMBINE and LDM generally exhibit better performances than LPT (see, e.g., Lee and Massey 1988; Michiels et al. 2007) but still guarantee a very limited computational complexity.

Table 1 SLACK versus LPT/COMBINE/LDM: performance comparison on $P_m||C_{\max}$ instances from Iori and Martello (2008)

$[a, b]$	m	Non-uniform instances	SLACK versus LPT			SLACK versus COMBINE			SLACK versus LDM		
		#	W	E	L	W	E	L	W	E	L
1–100	5	50	31	16	3	30	16	14	0	44	6
	10	40	32	8	0	29	8	3	0	40	0
	25	40	23	17	0	23	17	0	1	39	0
1–1000	5	50	39	10	1	39	10	1	1	44	5
	10	40	40	0	0	33	0	7	3	36	1
	25	40	27	12	1	27	12	1	2	36	2
1–10000	5	50	39	10	1	39	10	1	2	39	9
	10	40	40	0	0	32	0	8	7	29	4
	25	40	28	10	2	28	10	2	5	30	5
$[a, b]$	m	Uniform instances	SLACK versus LPT			SLACK versus COMBINE			SLACK versus LDM		
		#	W	E	L	W	E	L	W	E	L
1–100	5	50	12	37	1	10	38	2	0	44	6
	10	40	14	20	6	9	21	10	2	27	11
	25	40	10	29	1	4	23	13	4	28	8
1–1000	5	50	32	15	3	31	15	4	1	30	19
	10	40	27	5	8	21	5	14	4	6	30
	25	40	24	12	4	17	6	17	3	10	27
1–10000	5	50	36	12	2	36	11	3	0	12	38
	10	40	37	0	3	30	0	10	2	1	37
	25	40	22	11	7	15	6	19	4	10	26
Overall		780	513	224	43	453	208	119	41	505	234
		SLACK	LPT			COMBINE			LDM		
Tot. Time (ms)		104	75			105			5453		

SLACK heuristic

Input: $P_m||C_{max}$ instance with m machines and n jobs with processing times p_j $(j = 1, \ldots, n)$.

1: Sort jobs by non-increasing p_j.

2: Consider $\left\lceil \frac{n}{m} \right\rceil$ tuples with size m given by jobs $1, \ldots, m; m+1, \ldots, 2m$, etc.. If n is not multiple of m, add dummy jobs with null processing time in the last tuple.

3: For each tuple, compute the associated slack, namely $p_1 - p_m$, $p_{(m+1)} - p_{2m}, \ldots, p_{(n-m+1)} - p_n$.

4: Sort tuples by non-increasing slack and then fill a list with consecutive jobs in the sorted tuples.

5: Apply list scheduling to this job ordering and return the solution.

The comparison is executed by counting how many times SLACK wins (W), is equivalent to (E), or loses (L) when compared to the relevant competitor. The results are reported in Table 1 where instances are aggregated by processing time range and number of machines as in Iori and Martello (2008). Running times of the heuristics are generally negligible; hence, we just report an aggregated entry summing up the CPU time over all instances.

SLACK algorithm strongly outperforms LPT rule in each instance category with the most impressive performance difference on non-uniform instances. Overall, on 780 benchmark literature instances, SLACK wins 513 times (65.8% of the cases) against LPT, ties 224 times (28.7%) and loses 43 times (5.5%) only. Given these results, SLACK heuristic can be regarded as a valuable alternative to the popular LPT rule.

Table 2 SLACK + NS versus LDM: performance comparison on $P_m||C_{\max}$ instances from Iori and Martello (2008)

$[a, b]$	m	Non-uniform instances	SLACK + NS versus LDM		
		#	W	E	L
1–100	5	50	7	38	5
	10	40	2	38	0
	25	40	1	39	0
1–1000	5	50	12	35	3
	10	40	16	24	0
	25	40	4	35	1
1–10000	5	50	24	22	4
	10	40	28	8	4
	25	40	19	18	3
$[a, b]$	m	Uniform instances	SLACK + NS versus LDM		
		#	W	E	L
1–100	5	50	1	48	1
	10	40	6	26	8
	25	40	5	32	3
1–1000	5	50	8	35	7
	10	40	10	11	19
	25	40	6	11	23
1–10000	5	50	10	20	20
	10	40	13	0	27
	25	40	6	10	24
Overall		780	178	450	152

The comparison between SLACK and COMBINE provides additional evidence on the effectiveness of SLACK. The proposed heuristic outperforms COMBINE on non-uniform instances and favorably compares to it on uniform instances. Finally, SLACK shows up to be competitive with LDM on non-uniform instances, while it is slightly inferior on uniform instances, yielding though a non-superior makespan in 70% of the instances. Notice that in the uniform instances the smaller the processing times distribution, the better the performances of SLACK. A possible explanation of this phenomenon is that , with larger processing times distributions, the slack between largest and smallest processing times of each tuple does not fully capture the jobs processing times variance and may negatively affect the performances of the algorithm. Besides, SLACK runs much faster than LDM, taking roughly 2% of the computational time required by that algorithm. Taking into account this last aspect, we performed a further test where we evaluated the performance of SLACK enhanced by a simple neighborhood search (NS) procedure. This procedure, called SLACK + NS, with an overall negligible increase in the CPU time (about 57 ms for the whole batch of instances), looks for the best swap SW_{ij} between any job i on the critical machine and any job j on any machine that possibly improves upon the makespan provided by SLACK. If an improvement is found, NS restarts and iterates until a local minimum is reached. The relevant results are provided in Table 2 and indicate that the performances of SLACK + NS are already globally superior to those of LDM (though still inferior on uniform instances), while the overall CPU time remains inferior by more than an order of magnitude.

5 Concluding remarks

We provided new insights into the well-known LPT rule for $P_m||C_{\max}$ problem and proposed a slight algorithmic revisiting which improves previously published approximation ratios for LPT. As second major contribution, from our approximation results we came up with a simple heuristic requiring very low computational complexity which strongly outperforms LPT on a large set of benchmark literature

instances and is competitive with more involved approaches such as COMBINE and LDM.

In our analysis of $P_m||C_{\max}$, we deployed a novel approach which relies on linear programming. The proposed LP reasoning could be considered a valid alternative to techniques based on analytical derivation and may as well find application in other combinatorial optimization problems. For example, an attempt in this direction has been recently proposed by Della Croce et al. (2018) for a multiperiod variant of the knapsack problem.

We remark that in this work we did not derive tight approximation bounds for *LPT-REV* algorithm. We reasonably expect that improved bounds can be stated and leave this issue to future research. Nonetheless, we found out $P_m||C_{\max}$ instances for $m \geq 3$ which provide a lower bound on the worst-case performance ratio of *LPT-REV* equal to $\frac{4}{3} - \frac{7}{3(3m+1)}$. These instances have $2m+2$ jobs with processing times:

$$p_j = 2m - \left\lfloor \frac{j+1}{2} \right\rfloor, \quad 1 \leq j \leq 2m-2;$$
$$p_j = m, \quad 2m-1 \leq j \leq 2m+2$$

It is easy to check that $C_m^{LPT\text{-}REV} = 4m-1$ and $C_m^* = 3m+1$ and that such values give the above performance ratio.

Acknowledgements The authors wish to thank an anonymous referee for pointing out the works on the LPT rule by Dosa (2004) and Dosa and Vizvari (2006). This work has been partially supported by "Ministero dell'Istruzione, dell'Università e della Ricerca" Award "TESUN-83486178370409 finanziamento dipartimenti di eccellenza CAP. 1694 TIT. 232 ART. 6."

6 Appendix: Proof of Proposition 6

Proof The relevant cases involve instances with $2m+2 \leq n \leq 3m$, where LPT schedules the critical job in third position on a machine. Also, notice that each non-critical machine must process at most three jobs to contradict the claim; otherwise, the results of Lemma 1 hold for $k \geq 4$. In addition, an optimal solution must assign at most three jobs to each machine. Otherwise, we would have $C_m^*(\mathcal{J}) \geq \sum_{j=n-3}^{n} p_j \implies p_n \leq \frac{C_m^*(\mathcal{J})}{4}$ which induces a $\left(\frac{5}{4} - \frac{1}{4m}\right)$ performance guarantee according to expression (3) (with $j' = n$).

Considering the above requirements for both LPT schedule and the optimal solution, we introduce different LP formulations which consider appropriate bounds on the completion times of the machines as well as on the optimal makespan according to the number of jobs and machines involved. We analyze two macro-cases defined by the two different LP modelings.

- Case a): $n = 3m$ ($m = 3$ $n = 9$; $m = 4$ $n = 12$);
- Case b): $2m+2 \leq n \leq 3m-1$ ($m = 3$ $n = 8$; $m = 4$ $n = 10, 11$).

6.1 Case a)

Since $n = 3m$, an optimal solution must process exactly three jobs on each machine to contradict the claim. This implies a lower bound on the optimum equal to $p_1 + p_{(3m-1)} + p_{3m}$ as well as that condition $p_1 \leq 2p_{3m}$ must hold (otherwise $p_n \leq \frac{C_m^*(\mathcal{J})}{4}$). Given the last condition, LPT couples jobs $1, 2, \ldots, m$, respectively, with jobs $2m, (2m-1), \ldots, (m+1)$ on the machines. A valid upper bound on $C_m^{LPT}(\mathcal{J})$ is hence given by $p_1 + p_{(m+1)} + p_{3m}$. In order to evaluate the worst-case LPT performance, we introduce a simple LP formulation with nonnegative variables p_j ($j = 1, \ldots, n$) related to job processing times and variable *opt* which again represents $C_m^*(\mathcal{J})$. As in model (8)–(16), we minimize the value of $C_m^*(\mathcal{J})$ after setting w.l.o.g. $C_m^{LPT}(\mathcal{J}) = 1$. The following LP model is implied:

$$\text{minimize} \quad opt \tag{64}$$
$$\text{subject to} \quad \sum_{j=1}^{3m} p_j \leq m \cdot opt \tag{65}$$
$$p_1 + p_{(3m-1)} + p_{3m} - opt \leq 0 \tag{66}$$
$$p_1 - 2p_{3m} \leq 0 \tag{67}$$
$$p_1 + p_{(m+1)} + p_{3m} \geq 1 \tag{68}$$
$$p_{(j+1)} - p_j \leq 0 \quad j = 1, \ldots, 3m-1; \tag{69}$$
$$p_j \geq 0 \quad j = 1, \ldots, 3m; \tag{70}$$

Constraints (65)–(70) represent the above-specified conditions together with the sorting of the jobs by decreasing processing time (constraint (69)). Solving model (64)–(70) by means of an LP solver (e.g., CPLEX) provides the optimal solution value of *opt* for any value of m. More precisely, we have $opt = 0.8571\cdots = \frac{6}{7}$ for $m = 3$ and $opt = 0.8421\cdots = \frac{16}{19}$ for $m = 4$. The claim is showed by the corresponding upper bounds $\frac{1}{opt}$ on the ratio $\frac{C_m^{LPT}(\mathcal{J})}{C_m^*(\mathcal{J})}$ which are equal to $\frac{7}{6}$ for $m = 3$ and to $\frac{19}{16} (< \frac{11}{9})$ for $m = 4$.

6.2 Case b)

As said above, LPT assigns three jobs to the critical machine. The possible values of n and m also imply that there is a non-critical machine in the LPT solution which schedules three jobs. As in *Case a)*, we introduce an LP formulation which reconsiders model (8)–(16). For the target non-critical machine loading three jobs, we distinguish the processing time of the last job assigned to the machine, represented by a nonnegative variable p', from the contribution of the other

two jobs, represented by nonnegative variable t'. Variables t_c, t'', opt have the same meaning as in model (8)–(16). First notice that $p' \geq p_{(n-1)}$ holds as the critical job is n and that neither job $n-1$ nor job n can contribute to the value of t'. Also, in any LPT schedule both t_c and t' are given by the sum of two jobs where the first job is between job 1 and job m. The following relations straightforwardly hold:

$$t_c \leq p_1 + p_{(m+1)};\ t' \geq p_m + p_{(n-2)}$$

Considering these conditions and the reasoning applied to model (8)–(16), we get the following LP model:

$$\text{minimize} \quad opt \tag{71}$$

$$\text{subject to} \quad \sum_{j=1}^{n} p_j - m \cdot opt \leq 0 \tag{72}$$

$$(t_c + p_n) + (t' + p') + t'' - \sum_{j=1}^{n} p_j = 0 \tag{73}$$

$$t_c - (t' + p') \leq 0 \tag{74}$$

$$(m-2)t_c - t'' \leq 0 \tag{75}$$

$$t_c + p_n = 1 \tag{76}$$

$$p_{(n-1)} - p' \leq 0 \tag{77}$$

$$p_m + p_{(n-2)} - t' \leq 0 \tag{78}$$

$$t_c - (p_1 + p_{(m+1)}) \leq 0 \tag{79}$$

$$p_{(j+1)} - p_j \leq 0 \quad j = 1, \ldots, n-1; \tag{80}$$

$$p_j \geq 0 \quad j = 1, \ldots, n; \tag{81}$$

$$t_c, t', t'', p', opt \geq 0 \tag{82}$$

Model (71)–(82) constitutes a backbone LP formulation for all subcases analyzed in the following.

6.2.1 $P_m||C_{\max}$ instances with $m = 4, n = 11$

To contradict the claim, an optimal solution must assign three jobs to three machines and two jobs to one machine. Assume first that the optimal solution processes the first three jobs on two machines. Since the optimal solution must schedule at least five jobs on two machines, a valid lower bound on $C_m^*(\mathcal{J})$ is equal to one half of the sum $(p_1 + p_2 + p_3 + p_{10} + p_{11})$. Thus, we can add constraint

$$opt \geq \frac{p_1 + p_2 + p_3 + p_{10} + p_{11}}{2}$$

to model (71)–(82). The corresponding LP optimal solution gives an upper bound on ratio $\frac{C_m^{LPT}(\mathcal{J})}{C_m^*(\mathcal{J})}$ equal to $\frac{1}{opt} = \frac{11}{9}$. Likewise, should the optimal solution schedule the first three jobs on three different machines, a valid lower bound on $C_m^*(\mathcal{J})$ would correspond to the following constraint

$$opt \geq \frac{p_1 + p_2 + p_3 + p_7 + p_8 + p_9 + p_{10} + p_{11}}{3}$$

since the optimal solution has to process at least eight jobs on three machines. If we add the last constraint to model (71)–(82), the LP optimal solution yields again a value of $\frac{1}{opt}$ equal to $\frac{11}{9}$.

6.2.2 $P_m||C_{\max}$ instances with $m = 4, n = 10$

An optimal solution assigns either three jobs to three machines and one job to the other machine, or three jobs to two machines and two jobs to the others. We again distinguish whether the optimal solution processes the first three jobs either on two or three machines. In the first case, since two machines must process at least four jobs in an optimal solution, a valid lower bound on $C_m^*(\mathcal{J})$ is equal to one half of the sum $(p_1 + p_2 + p_3 + p_{10})$. The optimal objective value of model (71)–(82) after adding constraint

$$opt \geq \frac{p_1 + p_2 + p_3 + p_{10}}{2}$$

provides an approximation ratio equal to $\frac{1}{opt} = \frac{11}{9}$. In the second case, as three machines must necessarily process at least seven jobs, we can add to the LP model the following constraint which represents a valid lower bound on $C_m^*(\mathcal{J})$:

$$opt \geq \frac{p_1 + p_2 + p_3 + p_7 + p_8 + p_9 + p_{10}}{3}$$

From the corresponding LP optimal solution, we get again $\frac{1}{opt} = \frac{11}{9}$.

6.2.3 $P_m||C_{\max}$ instances with $m = 3, n = 8$

We have to address the case where an optimal solution assigns two jobs to a machine and three jobs to the other two machines. This implies relations $C_m^*(\mathcal{J}) \geq p_1 + p_8$ and $C_m^*(\mathcal{J}) \geq \frac{p_1+p_2+p_6+p_7+p_8}{2}$ and the corresponding constraints to be added to model (71)–(82):

$$opt \geq p_1 + p_8;\ opt \geq \frac{p_1 + p_2 + p_6 + p_7 + p_8}{2}$$

For the solution provided by LPT, we now analyze all possible assignments of the jobs which can give t'. LPT separately schedules the first three jobs and job 4 with job 3 on the third machine. It immediately follows that if the target non-critical machine is the first one, then it must necessarily process job 6 as second job, i.e., $t' = p_1 + p_6$. Instead, if the non-critical machine is the second machine, then t' must be contributed by p_2 and p_5, i.e., $t' = p_2 + p_5$. To show this, first notice that the assignment $t' = p_2 + p_6$, which also implies $p' = p_7$ as well as having job 5 with job 3 and 4, cannot occur, as

it would prevent the critical machine from scheduling three jobs. Also, job 7 cannot be scheduled as second job on such non-critical machine. Eventually, a third possibility is to have $t' = p_3 + p_4$. Solving model (71)–(82) with the constraints related to the three possible assignments of t' provides the following upper bounds on ratio $\frac{C_m^{LPT}(\mathcal{J})}{C_m^*(\mathcal{J})}$:

$$t' = p_1 + p_6 \implies \frac{1}{opt} = \frac{15}{13}\left(< \frac{7}{6}\right);$$
$$t' = p_2 + p_5 \implies \frac{1}{opt} = \frac{7}{6};$$
$$t' = p_3 + p_4 \implies \frac{1}{opt} = \frac{7}{6}.$$

Putting together all the stated results, we get the claim. □

References

Abolhassani, M., Chan, T. H., Chen, F., Esfandiari, H., Hajiaghayi, M., Hamid, M., et al. (2016). Beating ratio 0.5 for weighted oblivious matching problems. In P. Sankowski & C. Zaroliagis (Eds.), *24th Annual European symposium on algorithms (ESA 2016)* (Vol. 57, pp. 3:1–3:18).

Alon, N., Azar, Y., Woeginger, G. J., & Yadid, Y. (1998). Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, *1*, 55–66.

Blocher, J. D., & Sevastyanov, D. (2015). A note on the coffman-sethi bound for LPT scheduling. *Journal of Scheduling*, *18*, 325–327.

Chen, B. (1993). A note on LPT scheduling. *Operation Research Letters*, *14*, 139–142.

Chen, B., Potts, C. N., & Woeginger, G. J. (1999). A review of machine scheduling: Complexity, algorithms and approximability. In D. Z. Du & P. M. Pardalos (Eds.), *Handbook of combinatorial optimization: Volume 1–3*. New York: Springer.

Chimani, M., & Wiedera, T. (2016). An ILP-based proof system for the crossing number problem. In P. Sankowski & C. Zaroliagis (Eds.), *24th annual European symposium on algorithms (ESA 2016)* (Vol. 57, pp. 29:1–29:13).

Coffman, E. G, Jr., Garey, M. R., & Johnson, D. S. (1978). An application of bin-packing to multiprocessor scheduling. *SIAM Journal on Computing*, *7*, 1–17.

Coffman, E. G, Jr., & Sethi, R. (1976). A generalized bound on LPT sequencing. *Revue Francaise d'Automatique Informatique, Recherche Operationelle Supplement*, *10*, 17–25.

Della Croce, F., Pferschy, U., & Scatamacchia, R. (2018). Approximation results for the incremental knapsack problem. In *Combinatorial algorithms: 28th international workshop, IWOCA 2017, Springer lecture notes in computer science* (Vol. 10765, pp. 75–87).

Dosa, G. (2004). Graham example is the only tight one for P || Cmax (in Hungarian). *Annales Univ Sci Budapest*, *47*, 207–210.

Dosa, G., & Vizvari, A. (2006). The general algorithm lpt(k) for scheduling identical parallel machines. *Alkamazott Matematikai Lapok*, *23*(1), 17–37. (in Hungarian).

Fischetti, M., & Martello, S. (1987). Worst-case analysis of the differencing method for the partition problem. *Mathematical Programming*, *37*, 117–120.

França, P. M., Gendreau, M., Laporte, G., & Müller, F. (1994). A composite heuristic for the identical parallel machine scheduling problem with minimum makespan objective. *Computers & Operations Research*, *21*, 205–210.

Frangioni, A., Necciari, E., & Scutellà, M. G. (2004). A multi-exchange neighborhood for minimum makespan parallel machine scheduling problems. *Journal of Combinatorial Optimization*, *8*, 195–220.

Frenk, J. B. G., & Rinnooy Kan, A. H. G. (1987). The asymptotic optimality of the LPT rule. *Mathematics of Operations Research*, *12*, 241–254.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability*. New York: W. H. Freeman.

Graham, R. L. (1969). Bounds on multiprocessors timing anomalies. *SIAM Journal on Applied Mathematics*, *17*, 416–429.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In P. L. Hammer, E. L. Johnson, & B. H. Korte (Eds.), *Discrete optimization II, annals of discrete mathematics* (Vol. 5, pp. 287–326).

Gupta, J. N. D., & Ruiz-Torres, A. J. (2001). A listfit heuristic for minimizing makespan on identical parallel machines. *Production Planning & Control*, *12*(1), 28–36.

He, Y., Kellerer, H., & Kotov, V. (2000). Linear compound algorithms for the partitioning problem. *Naval Research Logistics (NRL)*, *47*(7), 593–601.

Hochbaum, D. S. (Ed.). (1997). *Approximation Algorithms for NP-hard Problems*. Boston: PWS Publishing Co.

Hochbaum, D. S., & Shmoys, D. B. (1987). Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM*, *34*, 144–162.

Iori, M., & Martello, S. (2008). Scatter search algorithms for identical parallel machine scheduling problems. In *Metaheuristics for scheduling in industrial and manufacturing applications* (pp. 41–59).

Jansen, K. (2010). An eptas for scheduling jobs on uniform processors: Using an milp relaxation with a constant number of integral variables. *SIAM Journal on Discrete Mathematics*, *24*, 457–485.

Jansen, K., Klein, K. M., & Verschae, J. (2017). Improved efficient approximation schemes for scheduling jobs on identical and uniform machines. In *Proceedings of the 13th workshop on models and algorithms for planning and scheduling problems (MAPSP 2017)* (pp. 77–79).

Karmarkar, N., & Karp, R. M. (1982). The differencing method of set partitioning. Technical Report UCB/CSD 82/113, University of California, Berkeley.

Lee, C. Y., & Massey, J. D. (1988). Multiprocessor scheduling: Combining LPT and MULTIFIT. *Discrete Applied Mathematics*, *20*(3), 233–242.

Leung, J., Kelly, L., & Anderson, J. H. (2004). *Handbook of scheduling: Algorithms, models, and performance analysis*. Cambridge: CRC Press, Inc.

Michiels, W., Korst, J., Aarts, E., & van Leeuwen, J. (2007). Performance ratios of the Karmarkar–Karp differencing method. *Journal of Combinatorial Optimization*, *13*(1), 19–32.

Mireault, P., Orlin, J. B., & Vohra, R. V. (1997). A parametric worst-case analysis of the LPT heuristic for two uniform machines. *Operations Research*, *45*(1), 116–125.

Paletta, G., & Ruiz-Torres, A. J. (2015). Partial solutions and multifit algorithm for multiprocessor scheduling. *Journal of Mathematical Modelling and Algorithms in Operations Research*, *14*(2), 125–143.

Pinedo, M. L. (2016). *Scheduling: Theory, algorithms, and systems* (5th ed.). Berlin: Springer.