A tight linear time13/12 13 12 -approximation algorithm for the P 2 | | C max problem

Federico Della Croce, Rosario Scatamacchia, Vincent T'kindt

A tight linear time $\frac{13}{12}$-approximation algorithm for the $P2||C_{\max}$ problem

Federico Della Croce* † Rosario Scatamacchia ‡
Vincent T'kindt §

Abstract

We consider problem $P2||C_{\max}$ where the goal is to schedule n jobs on two identical parallel machines to minimize the makespan. We focus on constant factor approximation algorithms with complexity independent from the required accuracy. We exploit the famous Longest Processing Time (LPT) rule that 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 propose an approximation algorithm that applies LPT to a subset of $2k$ jobs, then considers a single step of local search on the obtained subschedule and finally applies list scheduling to the remaining jobs. This algorithm, when applied for $k = 5$, reaches a tight $\frac{13}{12}$-approximation ratio improving the ratio of $\frac{12}{11}$ proposed by He et al. in 2000. We use Mathematical Programming to analyze the approximation ratio of our approach. As a byproduct, we also show how to improve the FPTAS of Sahni for any $n > 1/\epsilon$ so as to reach an approximation ratio $(1+\epsilon)$ with time complexity $O(n + \frac{1}{\epsilon^3})$.

Keywords: Two Identical Parallel Machines Scheduling; Makespan; LPT rule; Mathematical Programming; Approximation.

1 Introduction

We consider the problem of scheduling n jobs on 2 identical parallel machines M_1 and M_2 to minimize the makespan $C_{\max}$. Each job j is defined by

*DIGEP, Politecnico di Torino, Italy, federico.dellacroce@polito.it
†CNR, IEIIT, Torino, Italy
‡DIGEP, Politecnico di Torino, Italy, rosario.scatamacchia@polito.it
§Université Francois-Rabelais, Laboratoire d'Informatique Fondamentale et Appliquée (EA 6300), ERL CNRS 7002, Tours, France, tkindt@univ-tours.fr

a processing time p_j and is required to be executed by one of the machines. We denote by c_j the completion time of job j in any given schedule and we have $C_{\max} = \max_{1 \leq j \leq n}(c_j)$. Using the standard three–field notation ([12]) this problem is denoted by $P2||C_{\max}$. It is NP-hard in the ordinary sense.

We tackle problem $P2||C_{\max}$ from the approximation point of view. For any problem instance, we denote by $C^*_{\max}$ the optimal makespan and by $C^X_{\max}$ the makespan provided by a generic algorithm X. We say that algorithm X has approximation ratio $\rho \geq 1$, if $C^X_{\max} \leq \rho \cdot C^*_{\max}$ for every instance of $P2||C_{\max}$.

A pioneering approximation algorithm for this problem is the Longest Processing Time (LPT) rule proposed by Graham ([11]) for the more general $P||C_{\max}$ problem with m machines. It requires to sort the jobs in non-ascending order of their processing times p_j $(j = 1, \ldots, n)$ and then 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). LPT has time complexity of $O(n \log n)$ due to the initial sorting of the jobs. Several properties have been established for LPT in the last decades [3, 5, 8, 11]. LPT generally exhibits much better performance in practice than the expected theoretical approximation ratios, especially as the number of jobs gets larger. Due to its simplicity and practical effectiveness, LPT became a cornerstone for the design of more involving exact or heuristic algorithms for problem $P||C_{\max}$. Very recently, a revisiting of the LPT rule has been proposed in [10].

We mention other popular approximation algorithms which exploit connections of $P||C_{\max}$ with bin packing: *Multifit* [7], *Combine* [20] and *Listfit* [13]. Such algorithms provide better worst case performance than LPT but at the cost of higher running times. Also, Polynomial Time Approximation Schemes (PTASs) were derived for the problem. The first PTAS was given Hochbaum and Shmoys ([16]). PTASs with improved running times were then provided in [2], [15], [17] and in [18]. In [22], a Fully Polynomial Time Approximation Scheme (FPTAS) was devised for $P2||C_{\max}$ (and for the more general $Pm||C_{\max}$, if the number of machines is fixed) which solves the problem with accuracy $(1 + \epsilon)$ in time $O(n^2/\epsilon)$. Such algorithms provide better worst case performance than LPT but at the cost of higher running times. The current best performing algorithm for $P2||C_{\max}$ running with polynomial time complexity independent from the accuracy and providing a constant approximation ratio of $\frac{12}{11}$ was presented in [14], while an approximation algorithm

with ratio limited to $\frac{9}{8}$ was given in [10].

In this paper, we propose an approximation algorithm with a constant ratio that first applies LPT to a subset of $2k$ jobs, then considers a single step of local search on the obtained subschedule and finally applies list scheduling to the remaining jobs. This algorithm, when applied for $k = 5$, reaches a tight $\frac{13}{12}$-approximation ratio improving the ratio of $\frac{12}{11}$ established in [14]. We use Mathematical Programming (MP) to analyze the approximation ratio of our approach. In a sense, the proposed approach resembles the reasoning employed in [21], where several LPs were used to determine the worst case approximation ratio of LPT rule on two uniform machines, and the reasoning in [10] where theoretical results were derived by means of Mathematical Programming techniques. Also, recently a growing attention has been given to Mathematical Programming as an alternative to mainstream proof systems based on analytical derivation (see, e.g., [1], [6], [9]). Finally, in Section 3, we show how to improve the running time of the FPTAS of Sahni ([22]) under mild restrictions.

2 A linear time $\frac{13}{12}$-approximation algorithm

2.1 Preliminaries

We consider the jobs are sorted by non-ascending values of their processing time, i.e. $p_j \geq p_{j+1}, \quad j = 1, \ldots, n-1$. We denote by *critical* the job that provides the makespan of a given schedule. The following proposition holds.

Proposition 1. *Consider a given algorithm H that assigns jobs $1, ..., 2k$ to the machines according to some policy and then applies LS to the remaining jobs $2k+1, ..., n$. If the critical job j is such that $j \geq 2k+1$, then*

$$\frac{C^H_{\max}}{C^*_{\max}} \leq 1 + \frac{1}{2(k+1)} = \rho$$

.

Proof. Assume w.l.o.g. that j is assigned to machine M_1 and denote by t_2 the completion time of machine M_2 before processing jobs $j, ..., n$. Then, as j is scheduled according to LS, we have $C^H_{\max} - p_j \leq t_2$ and $C^H_{\max} + t_2 = \sum_{i=1}^{j} p_i \leq \sum_{i=1}^{n} p_i \leq 2C^*_{\max}$. Correspondingly, we have $2C^H_{\max} - p_j \leq C^H_{\max} + t_2 \leq 2C^*_{\max}$, that is $C^H_{\max} \leq C^*_{\max} + \frac{p_j}{2}$. Hence, we have $\frac{C^H_{\max}}{C^*_{\max}} \leq \frac{C^*_{\max} + \frac{p_j}{2}}{C^*_{\max}} = 1 + \frac{p_j}{2C^*_{\max}}$. Besides, as $j \geq 2k+1$, in the optimal solution, one of the machines will be

assigned at least $(k+1)$ jobs with processing time not inferior to p_j, that is $C^*_{\max} \geq (k+1)p_j$. But then

$$\frac{C^H_{\max}}{C^*_{\max}} \leq 1 + \frac{p_j}{2C^*_{\max}} \leq 1 + \frac{p_j}{2(k+1)p_j} = 1 + \frac{1}{2(k+1)} = \rho.$$

□

The following corollary immediately follows.

Corollary 1. *Given a problem P_1 with n jobs, consider the subproblem P_{red} with the first $2k$ jobs only. If problem P_{red} is solved by an algorithm with approximation ratio $1 + \frac{1}{2(k+1)}$, then the same approximation ratio holds for P_1 by applying LS to the remaining subset of jobs.*

Proof. Indeed, if the critical job in $P_1 \in \{1, ..., 2k\}$, the approximation ratio cannot be superior to $(1 + \frac{1}{2(k+1)})$. Besides, if the critical job in $P_1 \in \{2k + 1, ..., n\}$, then the result of Proposition 1 holds. □

Remark 1. *In [11], a somewhat similar result was proved (and generalized to m machines) stating that, if problem P_{red} is solved to optimality, then the approximation ratio $1 + \frac{1}{2(k+1)}$ holds for P_1 by applying LS to the remaining subset of jobs. We remark, however, that requiring to solve problem P_{red} to optimality makes such algorithm inapplicable as soon as k becomes non negligible. For this reason, it has always been considered of interest, well after the publication of the findings in [11], to determine low complexity polynomial time algorithms providing constant time approximation ratio (see, for instance, the work in [14]).*

2.2 The approximation algorithm

We now turn to the presentation of a $\frac{13}{12}$-approximation algorithm (Algorithm A_1) which combines the LPT rule with a simple local search.

Algorithm A_1

1: **Input:** An instance I with n jobs and 2 machines, a parameter k.
2: Select the $2k$ largest processing time jobs of instance I inducing a reduced instance I' and apply LPT to I' obtaining schedule S'.
3: Search for the best swap $SW^1_{i,j}$ (if any) between any job i on machine M_1 and any job j on machine M_2 that improves the makespan of S'.
4: Search for the best swap $SW^2_{i,j,k}$ (if any) between any job i on machine M_1 and any pair of jobs j, k on machine M_2 that improves the makespan of S'.

5: Search for the best swap $SW^3_{i,j,k}$ (if any) between any job i on machine M_2 and any pair of jobs j, k on machine M_1 that improves the makespan of S'.
6: Apply the best swap (among $SW^1_{i,j}$, $SW^2_{i,j,k}$, $SW^3_{i,j,k}$) to S' reaching schedule S''.
7: Given S'', apply LS to the remaining $(n-2k)$ jobs and return the complete schedule.

In practice, Algorithm A_1 applies first LPT to the reduced instance I' composed by the $2k$ largest jobs yelding subschedule S'. Then, a single step of local search between pairs or triples of jobs is applied to I' yelding subschedule S''. Finally, starting from S'', LS is applied to the remaining $(n-2k)$ jobs.

Proposition 2. *Algorithm A_1 runs with complexity $O(k^2 \log k + n)$.*

Proof. We proceed by analyzing the execution time of each step of the algorithm. It is well known that finding the $k-th$ element in a vector of n elements can be done in linear time by adapting the median algorithm in [4]. Correspondingly, determining the largest $2k$ processing times can be done in $O(n)$ time. Step 2 requires then $O(n + k \log k)$ due to the additional application of LPT to instance I'. Computing the best swap between two jobs in Step 3 can be done in $O(k)$ by means of the following procedure. Denote by δ the difference between the completion times of the critical and non-critical machine and by $d_{i,j}$ the difference in processing time produced by $SW^1_{i,j}$ for any jobs i and j. Notice also that jobs are ordered by non-ascending processing time on both machines due to the application of LPT at Step 2. We first seek for swaps with $d_{i,j}$ as close as possible to $\frac{\delta}{2}$ by coupling the first job i $(= 1)$ on the critical machine with jobs $j = 1, 2, \ldots$ on the non-critical machine as long as $0 \leq d_{i,j} \leq \frac{\delta}{2}$. This determines the best job j, say job j', for the first job i. Then, we analyze the next possible swap by considering the second largest job on the critical machine $(i = 2)$ and jobs $j \geq j' + 1$ on the non critical machine. After processing all jobs on the non-critical machine, we then re-apply the same procedure where, for a given candidate i, we look for the first job j such that $d_{i,j} > \frac{\delta}{2}$. Overall, the best swap $SW^1_{i,j}$ is found in $O(k)$. In Step 4, we first compute and sort by non-ascending order of processing times all pairs (j, k) of jobs on the non-critical machine in $O(k^2 \log k)$. Then, we can apply the same procedure of Step 3 by comparing jobs on the critical machine with the array of $O(k^2)$ processing time entries of the ordered pairs of jobs (j, k). Therefore, the execution time of Step 4 is in $O(k^2 \log k)$. A similar reasoning also applies in Step 5. In Step 7, LS runs with complexity $O(n)$. The overall time complexity is hence $O(k^2 \log k + n)$. □

Theorem 1. *Algorithm A_1 applied to the largest jobs $1, 2, ..., 10$ (where dummy jobs with null processing times are added if $n < 10$), i.e. with parameter $k = 5$, reaches a tight $\frac{13}{12}$-approximation ratio.*

Proof. If $n < 10$, it is immediate to see that an equivalent instance with 10 jobs exists by adding $10 - n$ dummy jobs with null processing times. If $n > 10$, then, due to Corollary 1, it is sufficient to show that steps $2 - 6$ of Algorithm A_1 applied to the largest $2k = 10$ jobs provide approximation ratio not superior to $1 + \frac{1}{2(k+1)} = \frac{13}{12}$. To this extent, we rely on Mathematical Programming to evaluate the worst-case performance ratio of Algorithm A_1 on any instance with up to 10 jobs. We propose a complete enumeration approach that considers all possible LPT sequences, denoted by S_i^{LPT}, where by construction job 1 is assigned to machine M_1, while jobs 2 and 3 are assigned to M_2. Correspondingly, LPT rule may generate $2^7 = 128$ possible different S_i^{LPT} sequences depending on the processing times values, where the makespan may be either on M_1 or on M_2. Similarly, we consider all possible optimal S_j^{OPT} sequences where, without loss of generality, we assume job 1 is assigned to M_1. Correspondingly, $2^9 = 512$ possible different sequences may be optimal. Actually, this value can be reduced to 260 by eliminating all dominated jobs assignments. For instance, the assignment of jobs $1, \ldots, 5$ to M_1 and jobs $6, \ldots, 10$ to M_2 is always dominated by the assignment of jobs $1, \ldots, 4, 6$ to M_1 and jobs $5, 7, \ldots, 10$ to M_2 as in both cases the makespan is on machine M_1 and $p_1+p_2+p_3+p_4+p_5 \geq p_1+p_2+p_3+p_4+p_6$.

For every pair S_i^{LPT}, S_j^{OPT} of candidate solutions (for a total of 260x128 pairs), we generate two LP models (taking into account whether the makespan of LPT is either on M_1 or on M_2) that search for the jobs processing times that maximize the makespan determined by Algorithm A_1 provided that the optimal makespan is not superior to a given constant value denoted by C^* where, without loss of generality, we may arbitrarily set $C^* = 1$: any other assignment $C^* = \alpha$ would simply scale by a factor α the related processing times values and correspondingly the objective function value without affecting the approximation ratio.

More precisely, we consider an MP formulation with non-negative variables p_j ($j = 1, \ldots, 10$) denoting the processing times, non-negative variables $C_{\max}^{M_1}$ and $C_{\max}^{M_2}$ representing the completion time of M_1 and M_2, respectively, in the LPT schedule and a non-negative variable δ representing the difference between the above completion times. Finally, we introduce a non-negative variable $\hat{\delta}$ representing the maximum among the improvements reachable by the best possible swaps $SW_{i,j}^1$, $SW_{i,j,k}^2$ and $SW_{i,j,k}^3$, respectively, determined in steps $3 - 5$ of Algorithm A_1.

The processing times must satisfy the list scheduling constraints of the LPT solution and the requirement that the optimal makespan cannot exceed the constant parameter. Further constraints connecting variables p_j and $\hat{\delta}$ are also induced by the swaps considered in Steps 3-5 of Algorithm A_1. We consider here the case where LPT gives the makespan on M_1. Hence, the objective function value is given by the difference $(C_{\max}^{M_1} - \hat{\delta})$ to be maximized as we search for the worst-case. Let us denote by $w_{k,\ell}$ a 0/1 coefficient indicating if job ℓ is assigned to machine M_k in sequence S_i^{LPT}. Similarly, let us denote by $w_{k,\ell}^*$ a 0/1 coefficient indicating if job ℓ is assigned to machine M_k in the optimal sequence S_j^{OPT}. The following model is implied:

$$\text{Maximize } (C_{\max}^{M_1} - \hat{\delta}) \tag{1}$$

$$p_j \geq p_{j+1} \qquad j = 1, \ldots, 9 \tag{2}$$

$$\sum_{j=1}^{\ell-1} w_{1,j} p_j \leq \sum_{j=1}^{\ell-1} w_{2,j} p_j, \ \forall 2 \leq \ell \leq 10 \mid w_{1,\ell} = 1 \tag{3}$$

$$\sum_{j=1}^{\ell-1} w_{2,j} p_j \leq \sum_{j=1}^{\ell-1} w_{1,j} p_j, \ \forall 2 \leq \ell \leq 10 \mid w_{2,\ell} = 1 \tag{4}$$

$$\sum_{j=1}^{10} w_{1,j}^* p_j \leq C^* \tag{5}$$

$$\sum_{j=1}^{10} w_{2,j}^* p_j \leq C^* \tag{6}$$

$$C_{\max}^{M_1} = \sum_{j=1}^{10} w_{1,j} p_j \tag{7}$$

$$C_{\max}^{M_2} = \sum_{j=1}^{10} w_{2,j} p_j \tag{8}$$

$$\delta = C_{\max}^{M_1} - C_{\max}^{M_2} \tag{9}$$

$$\hat{\delta} \geq \min\{p_i - p_j, \delta - p_i + p_j\}, \ \ \forall i < j \mid w_{1,i} = w_{2,j} = 1 \tag{10}$$

$$\hat{\delta} \geq \min\{p_i - p_j - p_k, \delta - p_i + p_j + p_k\}, \ \ \forall i, j < k \mid w_{1,i} = w_{2,j} = w_{2,k} = 1 \tag{11}$$

$$\hat{\delta} \geq \min\{p_i + p_j - p_k, \delta - p_i - p_j + p_k\} \ \ \forall i < j, k \mid w_{1,i} = w_{1,j} = w_{2,k} = 1 \tag{12}$$

$$p_j \geq 0 \qquad j = 1, \ldots, 10 \tag{13}$$

$$\delta, \hat{\delta}, C_{\max}^{M_1}, C_{\max}^{M_2} \geq 0 \tag{14}$$

Here constraints (2) ensure that jobs are ordered by non-increasing processing times. Constraints (3) and (4) impose to the p_j variables the fulfillment of the List Scheduling requirement. Also, constraints (5) and (6) require that the completion times of both machines in the optimal solution is not superior to the optimal makespan $C_{\max}^*$. Then, constraint (7) indicates that the completion time of the critical machine is given by the sum of the processing times of the jobs assigned to that machine. Similarly, constraint

(8) provides the same information for the non-critical machine. Constraint (9) indicates that δ is the difference between the completion times of the two machines. Constraints (10)–(12) indicate that $\hat{\delta}$ must be not inferior to the value of the largest possible improvement reachable by the best possible swaps $SW^1_{i,j}$, $SW^2_{i,j,k}$ and $SW^3_{i,j,k}$, respectively. Notice that for conciseness we kept in constraints (10)–(12) a non-linear *min* notation that can be easily transformed into sets of linear constraints by means of *big-M* coefficients and the introduction of dedicated 0/1 variables. We report in Appendix 5 the explosion of constraint (10). A similar reasoning is employed in the modeling of constraints (11)–(12) which is omitted in the paper for sake of conciseness.

Finally, constraints (13) and (14) indicate that all variables are non-negative. Thus, the MP model to be solved turns out to be a MILP formulation.

Then, by iterating (twice, in order to handle the makespan of LPT either on M_1 or on M_2) the solution of the MILP model on all possible pairs S_i^{LPT}, S_j^{OPT} and taking the maximum value, we get the worst-case instance with up to 10 jobs.

After solving 2 x 260 x 128 = 66560 MILP models[1], we found that the worst-case is reached by the following example with vector of processing times $\{7/12, 5/12, 1/6, 1/6, 1/6, 1/6, 1/6, 1/6, 0, 0\}$. An LPT solution assigns jobs $1, 4, 6, 8$ to M_1 and jobs $2, 3, 5, 7, 9, 10$ to M_2 and has makespan = 13/12. Any swap of the type indicated in Steps 2-6 of A_1 does not lead to improvement. The optimal solution assigns jobs 1 and 2 to one machine and jobs $3, ..., 10$ to the other machine reaching makespan =1. Correspondingly, the approximation ratio is $\frac{13}{12}$. □

We can also state the following side result for algorithm A_1 applied to problem $P2||\sum_{i=1}^{2}(C_{M_i})^2$ where C_{M_i} refers to the completion time of the last job processed on M_i and the goal is to minimize the sum of the squares of the machine completion times.

Corollary 2. *For any $P2||\sum_{i=1}^{2} C^2_{M_i}$ instance, algorithm A_1 has a tight $\frac{145}{144}$-approximation ratio.*

[1]All LPTs with related optimal sequences, the generation code of model (1)-(14) embedding the extended linear formulation of constraints (10)–(12) and taking in input a given pair S_i^{LPT}, S_j^{OPT} and the MILP model to which corresponds the worst-case instance are available at: `https://drive.google.com/open?id=1IdII7LoSHhYPbmupRCTpThnmuSt-35gi`.

Proof. As indicated in [23], problems $P2||C_{\max}$ and $P2||\sum_{i=1}^{2} C_{M_i}^2$ are equivalent. Correspondingly, the tight instance provided above for problem $P2||C_{\max}$ constitutes also a worst-case instance for problem $P2||\sum_{i=1}^{2} C_{M_i}^2$. Since for this instance we have $C_{M_1} = \frac{13}{12}$, $C_{M_2} = \frac{11}{12}$ in the LPT schedule and $C_{M_1} = C_{M_2} = 1$ in the optimal solution, the approximation ratio of Algorithm A_1 is $\frac{(\frac{13}{12})^2+(\frac{11}{12})^2}{1+1} = \frac{145}{144}$. □

We remark that the result of Corollary 2 improves upon the approximation ratio of $\frac{50}{49}$ derived in [19].

3 Improving the FPTAS of Sahni

By exploiting Proposition 1 and Corollary 1, it is possible to improve upon the time complexity of $O(n^2/\epsilon)$ of the FPTAS proposed by Sahni ([22]). Consider a simple procedure that first runs the FPTAS in [22] to the subinstance only composed by the largest $2k$ jobs, with $k = \lceil \frac{1}{2\epsilon} - 1 \rceil$. Then, LS is applied to the remaining subset of jobs. The following proposition holds.

Proposition 3. *Given a $P2||C_{\max}$ instance with n jobs, an approximation ratio $(1+\epsilon)$ can be reached with complexity $O(\frac{1}{\epsilon^3} + n)$ for all $n > 1/\epsilon$.*

Proof. As the proposed procedure sets $k = \lceil \frac{1}{2\epsilon} - 1 \rceil$, the results of Proposition 1 and Corollary 1 guarantee an approximation ratio $1 + \frac{1}{2(k+1)} \leq 1 + \epsilon$. To bound the running time, notice that the FPTAS is applied to the subset of $2k$ jobs, with $k = \lceil \frac{1}{2\epsilon} - 1 \rceil$, thus yielding a time complexity of $O(\frac{(2k)^2}{\epsilon}) = O(\frac{1}{\epsilon^3})$. The additional linear contribution of n to the time complexity is due to the running of LS. □

We remark that the difference in terms of complexity of the proposed procedure, with respect to the FPTAS in [22], can be extremely large if $n >> \frac{1}{\epsilon}$. For instance, with $n = 10000$ and $\epsilon = 0.01$, while the FPTAS in [22] runs in $O(\frac{n^2}{\epsilon}) = O(10^{10})$, we get a time complexity of $O(\frac{1}{\epsilon^3} + n) = O(10^6 + 10^4)$ that represents a difference of more than 4 orders of magnitude.

4 Conclusions

We considered problem $P2||C_{\max}$ and showed that the well-known LPT rule followed by a single step of local search reaches in linear time a tight $\frac{13}{12}$-approximation ratio. As a byproduct, we also showed that for any $n > 1/\epsilon$ an approximation ratio $(1+\epsilon)$ can be reached by means of an algorithm running with complexity $O(n + \frac{1}{\epsilon^3})$.

In our analysis we deployed an approach relying on Mixed Integer Linear Programming modeling. The proposed MILP reasoning could be considered a valid alternative to techniques based on analytical derivation. An attempt in this direction has been recently proposed in [9] for a multiperiod variant of the knapsack problem. Due to the implications of Proposition 1, a generalization of the proposed approach for larger values of k, possibly combining LPT and other basic greedy rules such as, for instance, *Multifit* followed by a single step of local search, may possibly induce further improvement of the current result and is, therefore, definitely worthy of future investigation.

Acknowledgement 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".

References

[1] M. Abolhassani, T. -H Hubert. Chan, F. Chen, H. Esfandiari, M. Hajiaghayi, M. Hamid, and X. Wu. Beating Ratio 0.5 for Weighted Oblivious Matching Problems. In Piotr Sankowski and Christos Zaroliagis, editors, *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57, pages 3:1–3:18. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

[2] N. Alon, Y. Azar, G. J. Woeginger, and Y. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1:55–66, 1998.

[3] J. D. Blocher and D. Sevastyanov. A note on the Coffman–Sethi bound for LPT scheduling. *Journal of Scheduling*, 18:325–327, 2015.

[4] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7:448–461, 1973.

[5] B. Chen. A note on LPT scheduling. *Operation Research Letters*, 14:139–142, 1993.

[6] M. Chimani and T. Wiedera. An ILP-based Proof System for the Crossing Number Problem. In Piotr Sankowski and Christos Zaroliagis, editors, *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57, pages 29:1–29:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.

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

[8] E. G. Coffman(Jr.) and Ravi Sethi. A generalized bound on LPT sequencing. *Revue Francaise d'Automatique Informatique, Recherche Operationelle Supplement*, 10:17–25, 1976.

[9] F. Della Croce, U. Pferschy, and R. Scatamacchia. Approximation results for the incremental knapsack problem. In *Combinatorial Algorithms: 28th International Workshop, IWOCA 2017*, volume 10765 of *Springer Lecture Notes in Computer Science*, pages 75–87, 2018.

[10] F. Della Croce and R. Scatamacchia. The Longest Processing Time rule for identical parallel machines revisited. *Journal of Scheduling*, (forthcoming). DOI: 10.1007/s10951-018-0597-6.

[11] R. L. Graham. Bounds on multiprocessors timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416–429, 1969.

[12] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P. L. Hammer, E. L. Johnson, and B. H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287 – 326. 1979.

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

[14] Y. He, H. Kellerer, and V. Koto. Linear compound algorithms for the partitioning problems. *Naval Research Logistics*, 47:593–601, 2000.

[15] D. S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., 1997.

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

[17] K. Jansen. 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, 2010.

[18] K. Jansen, K. M. Klein, and J. Verschae. 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)*, Seeon Abbey, Germany, 2017.

[19] C. Koulamas and G. J. Kyparisis. An improved delayed-start LPT algorithm for a partition problem on two identical parallel machines. *European Journal of Operational Research*, 187:660–666, 2008.

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

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

[22] S. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM*, 23:116–127, 1976.

[23] R. Walter. A note on minimizing the sum of squares of machine completion times on two identical parallel machines. *Central European Journal of Operations Research*, 25:139–144, 2017.

5 Appendix: Extended linear formulation of constraint (10)

A linear formulation of constraint 10 can be expressed by introducing for each pair of jobs i, j the binary variables v'_{ij}, v''_{ij} and v'''_{ij}. Variable v'_{ij} is equal to 1 iff $p_i - p_j \leq \frac{\delta}{2}$, variable v''_{ij} is equal to 1 iff $\frac{\delta}{2} < p_i - p_j \leq \delta$ and variable v'''_{ij} is equal to 1 iff $\delta < p_i - p_j$. Correspondingly, $\forall i < j \mid w_{1,i} = w_{2,j} = 1$, the following set of big-M constraints (for a reasonable large value of M, e.g. $M = 1000$) are introduced.

$$v'_{ij} + v''_{ij} + v'''_{ij} = 1; \tag{15}$$

$$\frac{\delta}{2} - p_i + p_j \leq M v'_{ij} \tag{16}$$

$$-\frac{\delta}{2} + p_i - p_j \leq M(1 - v'_{ij}) \tag{17}$$

$$\delta - p_i + p_j \leq M(v'_{ij} + v''_{ij}) \tag{18}$$

$$-\delta + p_i - p_j \leq M v'''_{ij} \tag{19}$$

$$\hat{\delta} \geq p_i - p_j - M(1 - v'_{ij}) \tag{20}$$

$$\hat{\delta} \geq \delta - p_i + p_j - M(1 - v''_{ij}) \tag{21}$$

Indeed, constraint (15) indicates that either $v'_{ij} = 1$, or $v''_{ij} = 1$ or $v'''_{ij} = 1$.
Then, if $v'_{ij} = 1$, constraint (17) implies that $p_i - p_j \leq \frac{\delta}{2}$. Correspondingly, constraints (16, 18, 21) are inactive, while constraint (19) that implies that $p_i - p_j \leq \delta$ is dominated by constraint (17). Hence, the swap induces the makespan reduction $\hat{\delta} = p_i - p_j$ through constraint (20) in combination with the objective function (1).
Else, if $v''_{ij} = 1$, constraint (16) implies that $p_i - p_j \geq \frac{\delta}{2}$, while constraint (19) implies that $p_i - p_j \leq \delta$. Also, constraints (17, 18, 20) are inactive. Hence, the binding constraint is in this case constraint (21) that is satisfied as an equality, that is $\hat{\delta} = \delta - p_i + p_j$. Correspondingly, due to constraint (9), the new makespan will be on machine M_2 and its value in the objective function (1) will be $C^{M_2}_{max} + p_i - p_j$.
Else, $v'''_{ij} = 1$. In this case, constraint (16) induces $p_i - p_j \geq \frac{\delta}{2}$, and constraint (18) induces $p_i - p_j \geq \delta$, that is swap will not occur as it can only worsen the objective function value. Besides, constraints (17, 18, 20, 21) are inactive. Correspondingly, as $\hat{\delta}$ must be positive or null, it has negative coefficient in the objective function and is not further constrained, we have $\hat{\delta} = 0$.