Math. Program., Ser. A (2017) 161:135–158
DOI 10.1007/s10107-016-1004-8

FULL LENGTH PAPER

Approximability of average completion time scheduling on unrelated machines

René Sitters[1,2]

Received: 29 August 2012 / Accepted: 15 March 2016 / Published online: 21 April 2016

Abstract We show that minimizing the average job completion time on unrelated machines is $\mathcal{APX}$-hard if preemption of jobs is allowed. This provides one of the last missing pieces in the complexity classification of machine scheduling with (weighted) sum of completion times objective. The proof is based on a mixed integer linear program. This means that verification of the reduction is partly done by an ILP-solver. This gives a concise proof which is easy to verify. In addition, we give a deterministic 1.698-approximation algorithm for the weighted version of the problem. The improvement is made by modifying and combining known algorithms and by the use of new lower bounds. These results improve on the known $\mathcal{NP}$-hardness and 2-approximability.

Keywords Scheduling · $\mathcal{APX}$-hardness · Approximation algorithms · Average completion time · Quadratic programming

Mathematics Subject Classification 90B35 Scheduling theory, deterministic · 68W25 Approximation algorithms · 68Q25 Analysis of algorithms and problem complexity

1 Introduction

In the last two decades extensive research has been done on approximation algorithms for machine scheduling problems with the objective of minimizing the average (or

✉ René Sitters
r.a.sitters@vu.nl; sitters@cwi.nl

1 Vrije Universiteit, Amsterdam, The Netherlands

2 Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands

total) job completion time. For many of the studied problems, the complexity of finding close to optimal solutions is well-understood by now. We say that a minimization problem can be approximated within a factor $\alpha \geq 1$ if there is a polynomial time algorithm which computes for any instance a feasible solution whose (expected) value is at most α times the value of an optimal solution. Such an algorithm is called a (randomized) α-approximation algorithm. A paper by Schuurman and Woeginger [18] of 1999 gives an interesting (and still accurate) overview of scheduling problems for which the approximability is unknown. Arguably, the most interesting are the ones with precedence constraints. The area is still wide open here (see also [14]). The approximability of classical machine scheduling problems without precedence constraints is much better understood and this holds especially for the objective of minimizing average completion time. Almost all problems have been shown to be either polynomial time solvable, or to have a polynomial time approximation scheme (PTAS), which means that it is possible to approximate it within any arbitrarily small factor $1 + \epsilon$. A few were proven to be $\mathcal{APX}$-hard. The first polynomial time approximation scheme in this class of problems was given by Skutella and Woeginger [21] for the problem of minimizing total weighted completion time on identical machines. Soon after, this was extended to many other problems [1,4]. See also [2] for an extensive overview.

In this paper, we consider the problem of minimizing total (weighted) completion time on unrelated machines where preemption of jobs is allowed. In the standard 3-field notation introduced by Graham et al. [8] this is denoted by $R|pmtn|\sum C_j$ and, for the weighted case, by $R|pmtn|\sum w_j C_j$. Formally, an instance of our problem is given by numbers m and n and numbers $w_j \in \mathbb{Q}^+$ for all $j \in \{1, 2 \ldots, n\}$ and $p_{ij} \in \mathbb{Q}^+ \cup \{\infty\}$ for all $i \in \{1, 2 \ldots, m\}$ and $j \in \{1, 2 \ldots, n\}$. Job j requires p_{ij} *processing time* if it is completely scheduled on machine i. If $p_{ij} = \infty$ then job j cannot be processed by machine i (we shall use *process time* in stead of processing time for the duration that a job is processed in a given schedule). If job j is processed for a duration δ on machine i then the processed *fraction* will be δ/p_{ij}. A schedule is an assignment of jobs to machines over time such that all jobs are completely processed. We do not allow a machine to work on more than one job at a time or two machines to work on the same job simultaneously. For a given schedule we denote the completion time of job j by C_j and the objective value is the sum of the weighted completion times, i.e., $\sum_{j=1}^{n} w_j C_j$. We distinguish between the problem with unit job weights, $R|pmtn|\sum C_j$, and the more general problem with arbitrary job weights, denoted by $R|pmtn|\sum w_j C_j$.

For our problem, the paper [1] gives a PTAS assuming that the number of machines is constant, and Hoogeveen et al. [10] show APX-hardness of the non-preemptive variant. But for our problem as described above, no such result was known. Techniques for designing polynomial time approximation schemes have failed so far. In this paper we prove that already the unweighted version is $\mathcal{APX}$-hard. Hence, it cannot be efficiently approximated within an arbitrarily small constant, unless $\mathcal{P} = \mathcal{NP}$. Additionally, we give a 1.698-approximation algorithm. So far, the smallest approximation ratio was 2 (see [15,17,22]).

The $\mathcal{APX}$-hardness reduction was given without proof of correctness in the conference version [19]. That paper also gives a 1.81-approximation algorithm and hints at the possible improvement that we present here.

$\mathcal{APX}$-hardness The complexity result is interesting for at least three reasons. Firstly, this provides one of the last missing pieces in the complexity classification of machine scheduling with (weighted) sum of completion times objective [2]. The only remaining open problems are those with precedence constraints.

Secondly, the problem $R|pmtn|\sum C_j$ has the peculiar property that the preemptive version is much harder to solve than the non-preemptive version which, for this problem, was shown to be solvable by any weighted bipartite matching algorithm almost forty years ago [3,11]. Intuitively, preemption should make problems easier, just as LP's are in general easier than ILP's. Indeed, for almost all scheduling problems the non-preemptive version is at least as hard to solve as its preemptive counterpart. In fact, $R|pmtn|\sum C_j$ appears the only scheduling problem for which the preemptive version is $\mathcal{APX}$-hard and the non-preemptive version is solvable in polynomial time.

A third interesting aspect is the proof itself, which is partly written in the form of a mixed integer program. That means, we claim the optimal value of the mixed ILP but do not prove it. We leave it for the reader to check the optimal solution by an ILP-solver. Writing down a complete proof would make it unnecessarily long and basically boils down to solving the mixed ILP by hand. Since ILP solvers are widely available and any solver can handle our simple mixed ILP problem, this short form is a much cleaner, more reliable proof than the much longer alternative.[1] Although, ILP solvers may give incorrect solutions, this not a real issue here since the mixed ILP has only two binary variables and can thus be reduced to four LP's (see the discussion at the end of the proof of Lemma 4).

Of course, not every complex reduction may be simplified using such a mathematical programming approach, but if possible, then such an approach may be preferred. Arguably, the answer given by an ILP solver for a relatively easy mixed ILP is more reliable than the answer given by one or two people having checked a case analysis of many pages.

Algorithm The second contribution of this paper is a significant improvement over the best known approximation ratio for the problem. This applies even to the weighted version of the problem. Several 2-approximations are known. Skutella [22] gave a 2-approximation algorithm using a convex quadratic program. Schulz and Skutella [17] and Queyranne and Sviridenko [15] gave $(2+\epsilon)$-approximations using interval-indexed linear programs. One difficulty of improving on the constant 2 is finding good lower bounds on the optimal solution. Here, we give two new lower bounds and show how a modification of the convex program used in [22] reduces the approximation ratio to 1.81. In [17], Schulz and Skutella raised the question whether a stronger LP-formulation could improve on the factor 2. We show that a similar modification leads to the same improved ratio. Further, we show that if we apply both our new algorithm and the algorithm by Queyranne and Sviridenko [15] and take the best of the two solution then this yields a 1.698-approximate solution.

Both improved approximation factors follow from new lower bounds on the optimal solution. The proofs for these two lower bounds are quite technical and probably not

[1] An unpublished earlier version of the proof contains a case-analysis of roughly five pages instead of the one-page program that we give here.

the most interesting parts of the paper and are therefore placed at the end of the corresponding sections. What does make this result interesting is that given these new bounds, the improved ratios follow easily from known algorithms.

2 Inapproximability

The problem $R|pmtn|\sum C_j$ is known to be $\mathcal{NP}$-hard [20]. Here, we give a reduction which shows that the problem is even $\mathcal{APX}$-hard. The reduction itself is simpler than the one used in [20]. The simplification is essential to obtain an approximation preserving reduction. However, this simplification also makes the reduction very fragile and proving correctness of the reduction becomes non-trivial. Especially the preemptive setting makes it hard to prove correctness. Fortunately, an ILP solver is a perfect remedy for this step.

The reduction We reduce from the *maximum bounded 3-dimensional matching* problem which was proven to be $\mathcal{APX}$-hard by Kann [12].

Maximum bounded 3-dimensional matching (3DM-B):

An instance is given by pairwise disjoint sets $A = \{a_1, \ldots, a_m\}$, $B = \{b_1, \ldots, b_m\}$, and $C = \{c_1, \ldots, c_m\}$ and a set $T \subseteq A \times B \times C$ of cardinality $n \geq m$ such that each element of A, B and C appears in at most three triples from T. A solution is a set $S \subseteq T$ of pairwise disjoint triples. The goal is to maximize $|S|$.

Lemma 1 *For any instance of* 3DM- B, *we have* $m \leq n \leq 3m$ *and the optimal value is at least* $n/7$.

Proof By assumption $m \leq n$. Since each element appears in at most three triples there are at most $3m$ triples. Further, each triple has a non-empty intersection with at most six other triples. Therefore, any maximal set S of pairwise independent triples has size at least $n/(6+1)$. □

For our reduction we assume that $n = 3m$. This is without loss of generality since if the instance has n' triples with $n' < 3m$ then we can take any triple t_j from T and add $3m - n'$ copies of t_j. This does not change the optimal value. Now, elements may appear in more than three triples but we shall not use that property anymore. From now we may assume that

$$n = 3m \text{ and } \textsc{Opt} \geq n'/7 \geq m/7 = n/21. \tag{1}$$

To simplify notation we denote triples of T by $t_j = (x_j, y_j, z_j) \in \{1, \ldots, n\}^3$ instead of $t_j = (a_{x_j}, b_{y_j}, c_{z_j})$.

Given any instance I_M of 3DM- B we construct an instance I_R of the scheduling problem with $3m + n$ machines and $(15/\lambda + 1)n + 2m$ jobs, where $\lambda = 1/4$. In fact, any $\lambda \leq 1/4$ with $1/\lambda$ integer will do and we prefer to keep λ in the notation as it is more insightful to think of $1/\lambda$ as a large integer.

Machines:

The machines are $\mathcal{A}_1, \cdots, \mathcal{A}_m, \mathcal{B}_1, \cdots, \mathcal{B}_m, \mathcal{C}_1, \cdots, \mathcal{C}_m$, and $\mathcal{T}_1, \ldots, \mathcal{T}_n$. The first $3m$ correspond to the elements of A, B and C and the last n correspond to the triples T.

$\mathcal{B}$-jobs:

For each machine $\mathcal{B}_i$ we define one job with processing time 6 on that machine and which cannot be processed on any other machine.

$\mathcal{C}$-jobs:

For each machine $\mathcal{C}_i$ we define one job with processing time 11 on that machine and which cannot be processed on any other machine.

$\mathcal{T}$-jobs:

For each machine $\mathcal{T}_i$ we define $15/\lambda$ jobs with processing time 16 on that specific machine and which cannot be processed on any other machine.

Triple-jobs:

For each triple $t_j = (x_j, y_j, z_j)$ we define one *triple-job* j with processing times 18, 15, and 12 on, respectively, machine $\mathcal{A}_{x_j}$, $\mathcal{B}_{y_j}$, and $\mathcal{C}_{z_j}$. The processing time on machine $\mathcal{T}_j$ is λ and it cannot be processed on any of the other $3m + n - 4$ machines.

Correctness of the reduction Let us first consider the instance without the triples jobs, i.e., only with the $\mathcal{B}$-,$\mathcal{C}$-, and $\mathcal{T}$-jobs. An optimal schedule is easily found since each of these jobs fits on only one machine: The $\mathcal{B}$-jobs on the $\mathcal{B}$-machines complete at time 6 and the $\mathcal{C}$-jobs on the $\mathcal{C}$-machines complete at time 11. Their total completion time is $6m + 11m$. The total completion time of the $\mathcal{T}$-jobs is $n(16 + 2 \cdot 16 + \cdots + 15/\lambda \cdot 16)$. We denote this schedule by $\sigma_{\mathcal{BCT}}$ and its total completion time by $C_{\mathcal{BCT}}$. We do not need to know its exact value but do note that $C_{\mathcal{BCT}} = c_\lambda n$ for some constant c_λ (depending on λ).

For each triple $t_j = (x_j, y_j, z_j)$ we define a schedule that we call the *optimal schedule of triple-job* j: Job j is processed on machine $\mathcal{A}_{x_j}$ between time 0 and 6 and on machine $\mathcal{B}_{y_j}$ between time 6 and 11, and on machine $\mathcal{C}_{z_j}$ between time 11 and 15 (the fractions done on these $\mathcal{A}$-, $\mathcal{B}$- and $\mathcal{C}$-machines are, respectively, 6/18, 5/15, and 4/12, which equals 1/3 in either case). Note that we can add any triple-job j to $\sigma_{\mathcal{BCT}}$ and complete it at time 15 without changing $\sigma_{\mathcal{BCT}}$. The following lemma is obvious.

Lemma 2 *Instance I_M has a matching of size k if and only if there exists a schedule for I_R in which exactly k triple-jobs get their optimal schedule.*

If a triple job is scheduled completely at the beginning of its triple machine then it completes at time λ. However, this will delay all other $15/\lambda$ jobs on the machine by λ. One could say that, in this case, the *contribution* of the job in the total completion time is $\lambda + (15/\lambda)\lambda = 15 + \lambda$. The meaning of contribution is made precise later and we will show (see Claim 1) that if a triple job contributes less than $15 + \lambda$, then not only must it complete before time $15 + \lambda$ but also, its schedule must have a substantial overlap with its optimal schedule. We prove this by using a mixed ILP to compute a lower bound on the contribution of a triple job for all possible schedules. For this, it clearly suffices to restrict the analysis to schedules of the triple job on the interval $[0, 15 + \lambda]$. For the easy of analysis, however, we shall consider the slightly larger interval $[0, 16]$.

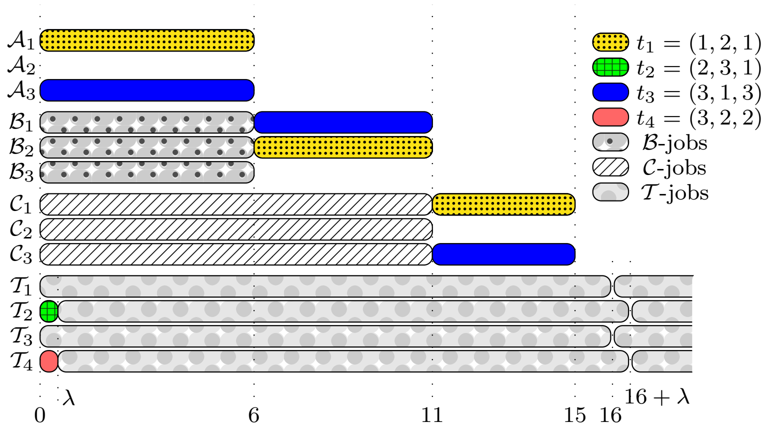

Fig. 1 An optimal schedule for the given instance. The maximum matching has cardinality $k = 2$ (in this example, $m = 3$ and $n = 4$ but in the proof we assume that $n = 3m$)

We say that the schedule of a triple job j is *nearly optimal* if it is scheduled on machine $\mathcal{A}_{x_j}$ for strictly more than 3 time units between 0 and 6, on machine $\mathcal{B}_{y_j}$ for strictly more than 2.5 units between 6 and 11, and on $\mathcal{C}_{z_j}$ for strictly more than 2.5 units between 11 and 16. Then, k triple-jobs can get their optimal schedules simultaneously if and only if they can get nearly optimal schedules simultaneously. We can strengthen Lemma 2 as follows.

Lemma 3 *Instance I_M has a matching of size k if and only if there exists a schedule for I_R in which exactly k triple-jobs get a nearly optimal schedule.*

$\mathcal{APX}$-hardness follows almost directly from the following Lemma, which we prove by solving a mixed ILP. Note that λ and c_λ are constants and $C_{\mathcal{BCT}} = c_\lambda n$ and $k = \Omega(n)$.

Lemma 4 $\textsc{Opt}(I_M) = k \;\Leftrightarrow\; \textsc{Opt}(I_R) = C_{\mathcal{BCT}} + (15 + \lambda)n - \lambda k.$

Proof Assume that $\textsc{Opt}(I_M) = k$. First we give a schedule with value $C_{\mathcal{BCT}} + (15 + \lambda)n - \lambda k$ and then we prove that no better schedule exists. Since we have equality on both sides it is enough to prove one direction only.

We construct a feasible schedule for I_R as follows. We schedule the $\mathcal{B}$- and $\mathcal{C}$-jobs as $\sigma_{\mathcal{BCT}}$. We give k triple-jobs their optimal schedule and place any other triple-job j at the beginning of machine $\mathcal{T}_j$. The $\mathcal{T}$-jobs are scheduled subsequently and as early as possible in the obvious way (see Fig. 1). The completion time of a triple-job j which does not get its optimal schedule is λ. Then, all $\mathcal{T}$-jobs on the machine $\mathcal{T}_j$ are delayed by λ which gives a total extra cost for these $\mathcal{T}$-jobs of $\lambda 15/\lambda = 15$. The total completion time of the schedule is $C_{\mathcal{BCT}} + 15k + \lambda(n-k) + 15(n-k) = C_{\mathcal{BCT}} + (15+\lambda)n - \lambda k$.

Next we prove that no better schedule exists. Consider an *optimal* schedule σ for instance I_R. We define for every triple-job j its *contribution* C'_j in the total completion time of σ as follows. It is the completion time of job j plus a lower bound on the total time that it shifts $\mathcal{B}$-, $\mathcal{C}$-, and $\mathcal{T}$-jobs ahead in time compared with schedule $\sigma_{\mathcal{BCT}}$.

Precisely, it is the completion time C_j plus the time it is processed on machine $\mathcal{B}_{y_j}$ between time 0 and 6, plus the time it is processed on machine $\mathcal{C}_{z_j}$ between time 0 and 11, plus $15/\lambda$ times the time it is processed on machine $\mathcal{T}_j$ between 0 and 16. Notice that the value C'_j is defined by the schedule of job j only. The total completion time of σ is at least

$$\textsc{Opt}(I_R) \geq C_{\mathcal{BCT}} + \sum_{j=1}^{n} C'_j. \tag{2}$$

Claim 1 (i) $C'_j \geq 15$, *for any* $j \in \{1, \ldots, n\}$.
(ii) $C'_j \geq 15 + \lambda$ *(with* $\lambda = 1/4$*) if the schedule of* j *is* not *nearly optimal.*

We shall prove this claim below. First we show that the proof of the lemma follows directly from Claim 1. If $\textsc{Opt}(I_M) = k$ then by Lemma 3 at most k jobs can get their nearly optimal schedule. From (2) and Claim 1 we obtain

$$\begin{aligned}\textsc{Opt}(I_R) &\geq C_{\mathcal{BCT}} + \sum_{j=1}^{n} C'_j \\ &\geq C_{\mathcal{BCT}} + 15k + (15+\lambda)(n-k) \\ &= C_{\mathcal{BCT}} + (15+\lambda)n - k\lambda.\end{aligned}$$

Proof of Claim 1 We formulate a mixed integer linear program that computes a lower bound on C'_j in case (i) and (ii). Both properties hold for any $\lambda \leq 1/4$ but we assume $\lambda = 1/4$ here.

Since C'_j depends on the schedule of job j only, it is enough to consider one specific triple-job j with its machines $\mathcal{A}_{x_j}$, $\mathcal{B}_{y_j}$, $\mathcal{C}_{z_j}$ and $\mathcal{T}_j$. Thus, in the sequel we consider only four machines and one job. We ignore all other machines and jobs and remove the index j from the notation. We refer to the machines as the $\mathcal{A}$-, $\mathcal{B}$-, $\mathcal{C}$-, and $\mathcal{T}$-machine and refer to job j simply as *the job*.

Let C and C' be, respectively, the completion time and contribution of the job for our fixed schedule σ. If the job does not complete before time 16 then $C' \geq C \geq 16$ in which case (i) and (ii) hold. So assume from now that it completes before time 16. We label the time interval from 0 to 6, from 6 to 11, and from 11 to 16 by, respectively, interval 1,2, and 3. For $i = 1, 2, 3$, the variables $\mathtt{a}_i, \mathtt{b}_i, \mathtt{c}_i$ are defined as the *time* that the job is scheduled on, respectively, the $\mathcal{A}$-, $\mathcal{B}$-, and $\mathcal{C}$-machine within interval i. Let τ be the time that the job is scheduled on the $\mathcal{T}$-machine. Note that $\mathtt{C}' = \mathtt{C} + \mathtt{b}_1 + \mathtt{c}_1 + \mathtt{c}_2 + 15/\lambda \cdot \tau$, with $\lambda = 1/4$. The objective is to minimize C'.

$$\min \quad \mathtt{C}' = \mathtt{C} + \mathtt{b}_1 + \mathtt{c}_1 + \mathtt{c}_2 + 60\tau. \tag{3}$$

Let us define variables for the row sums and column sums, i.e.,

$$\begin{aligned}\mathtt{a} &= \mathtt{a}_1 + \mathtt{a}_2 + \mathtt{a}_3 \\ \mathtt{b} &= \mathtt{b}_1 + \mathtt{b}_2 + \mathtt{b}_3 \\ \mathtt{c} &= \mathtt{c}_1 + \mathtt{c}_2 + \mathtt{c}_3\end{aligned}$$

$$\begin{aligned} \mathtt{I}_1 &= \mathtt{a}_1 + \mathtt{b}_1 + \mathtt{c}_1 \\ \mathtt{I}_2 &= \mathtt{a}_2 + \mathtt{b}_2 + \mathtt{c}_2 \\ \mathtt{I}_3 &= \mathtt{a}_3 + \mathtt{b}_3 + \mathtt{c}_3 \end{aligned} \tag{4}$$

A lower bound on C is the total time that the job is processed. Hence,

$$\mathtt{C} \geq \mathtt{a} + \mathtt{b} + \mathtt{c} + \tau. \tag{5}$$

Another constraint is that the fractions should add up to 1. Hence,

$$\mathtt{a}/18 + \mathtt{b}/15 + \mathtt{c}/12 + \tau/(1/4) = 1.$$

Multiplying everything by 180 gets rid of the fractions:

$$10\mathtt{a} + 12\mathtt{b} + 15\mathtt{c} + 720\tau = 180. \tag{6}$$

In the first interval we can schedule at most 6 units and in the second and third both at most 5. This gives

$$\mathtt{I}_1 \leq 6, \quad \mathtt{I}_2 \leq 5 \quad \text{and} \quad \mathtt{I}_3 \leq 5. \tag{7}$$

If the job completes before time 6 then $\mathtt{I}_2 = \mathtt{I}_3 = 0$. If it completes after time 6 then $C \geq 6 + \mathtt{I}_2 + \mathtt{I}_3$. Let binary variable $\mathtt{B}_1$ be 0 if the job completes before time 6 and $\mathtt{B}_1 = 1$ otherwise. Then the following bounds are valid [The coefficient 10 herein is just a large enough number: see Eq. (7)]

$$C \geq 6\mathtt{B}_1 + \mathtt{I}_2 + \mathtt{I}_3 \text{ and } \mathtt{I}_2 + \mathtt{I}_3 \leq 10\mathtt{B}_1, \tag{8}$$

Similarly, let binary variable $B_2 = 0$ if the job completes before time 11 and let $\mathtt{B}_2 = 1$ otherwise. Then the following bounds are valid (the coefficient 5 herein is just a large enough number: see Eq. (7)).

$$C \geq 11\mathtt{B}_2 + \mathtt{I}_3 \text{ and } \mathtt{I}_3 \leq 5\mathtt{B}_2. \tag{9}$$

All variables are non-negative:

$$\mathtt{B}_1, \mathtt{B}_2 \in \{0, 1\} \text{ and all variables } \geq 0. \tag{10}$$

If one solves[2] the mixed ILP given by (3)–(10) by any correct mixed ILP-solver then one finds that the optimal solution corresponds with *the optimal schedule of the job*. Hence, the optimal value is exactly 15.

[2] We used the free solver lp_solve.

If the job is *not* scheduled nearly optimal then at least one of the following constraints holds:

$$\mathtt{a}_1 \leq 3, \quad \mathtt{b}_2 \leq 2.5, \quad \mathtt{c}_3 \leq 2.5.$$

If we add any of the three constraints above then the optimal value is 15.25. The corresponding solution given by the solver is to put the job completely on the $\mathcal{T}$-machine, i.e., $\tau = 0.25$ and $\mathtt{B}_1 = \mathtt{B}_2 = 0$. We conclude that if the job is not scheduled nearly optimal then the optimum is exactly $15 + \lambda$ with $\lambda = 1/4$.

Note that in general, the arithmetic in ILP solvers is not exact rational arithmetic but floating-point arithmetic. As a side-effect the results come without any formal guarantee. Note that our program has only two binary variables and can thus be translated into four LP's. LP solvers with precise arithmetics can be used to verify the optimal solutions. Further, one may even formulate the duals of the LP's and check (by hand) feasibility of corresponding dual solutions. □

Lemma 4 gives the gap that proves $\mathcal{APX}$-hardness. This follows directly from the fact that λ and c_λ are constants and $C_{\mathcal{BCT}} = c_\lambda n$ and $k = \Omega(n)$. Let us prove this formally. We show that any polynomial time approximation scheme (PTAS) for the scheduling problem implies a polynomial time approximation scheme for the 3DM- Bproblem.

Assume that $\textsc{Opt}(I_M) = k$ and assume that we have a $(1 + \epsilon)$-approximation for instance I_R for some $\epsilon > 0$. From (1), we may assume that $k \geq n/21$ and we argued before that $C_{\mathcal{BCT}} = c_\lambda n$ for some constant c_λ. Hence,

$$\begin{aligned} C_{\mathcal{BCT}} + (15 + \lambda)n &= c_\lambda n + (15 + \lambda)n \\ &= (c_\lambda + 15 + \lambda)n \\ &\leq (c_\lambda + 15 + \lambda)21k \\ &= d_\lambda k, \end{aligned} \tag{11}$$

where $d_\lambda = 21(c_\lambda + 15 + \lambda)$. By Lemma 4 ($\Rightarrow$) and Eq. (11), the approximate solution has value at most

$$\begin{aligned} &(1 + \epsilon)\left(C_{\mathcal{BCT}} + (15 + \lambda)n - \lambda k\right) \\ &\quad = C_{\mathcal{BCT}} + (15 + \lambda)n - \lambda k + \epsilon\left(C_{\mathcal{BCT}} + (15 + \lambda)n - \lambda k\right) \\ &\quad \leq C_{\mathcal{BCT}} + (15 + \lambda)n - \lambda k + \epsilon(d_\lambda k - \lambda k) \\ &\quad = C_{\mathcal{BCT}} + (15 + \lambda)n - \lambda k(1 - \epsilon d_\lambda/\lambda + \epsilon) \end{aligned}$$

Lemma 4 implies that there is a matching of size at least $k(1 - \epsilon d_\lambda/\lambda + \epsilon)$. Hence, this approximates $\textsc{Opt}(I_M)$ within a multiplicative factor $(1 - \epsilon d_\lambda/\lambda + \epsilon)$. The ratio goes to 1 if ϵ goes to zero. We conclude that any polynomial time approximation scheme (PTAS) for computing the optimal value of the scheduling problem implies a PTAS for computing the maximum 3-dimensional matching.

Theorem 1 *The problem $R|pmtn|\sum_j C_j$is $\mathcal{APX}$-hard.*

Some remarks on the numbers in the reduction. The numbers in the reduction are balanced carefully but there is some slack. For the $\mathcal{T}$ jobs, any constant of at least $15+\lambda$ satisfies. We only need that a triple job which completes before time $15+\lambda$ and which is (partially) scheduled on a triple machine, delays *all* $\mathcal{T}$ jobs on that machine. Further, we do need the processing times of a triple job on its $\mathcal{A}$-, $\mathcal{B}$-, and $\mathcal{C}$-machine (18,15, and 12) to be decreasing. Reducing the differences between these processing times gives a smaller gap for the inapproximability. On the other hand, the differences cannot be too large. For example if the $\mathcal{C}$-machine is much faster than the $\mathcal{A}$-machine, then scheduling a triple job completely at the beginning of its $\mathcal{C}$-machine becomes beneficial. The processing times of the $\mathcal{B}$- and $\mathcal{C}$-jobs are chosen such that the optimal schedule of a triple job divides the job in three equal fractions. Although we do not need these to be exactly $1/3$, the largest gap for the inapproximability is obtained when the three fractions are roughly the same. Replacing the triple 18, 15, 12 by, for example, 6, 5, 4, or by 7, 6, 5 would still yield a valid reduction but then, adjusting other (processing) times accordingly leads to more fractional numbers than are used now. One could say that we selected nice numbers within a certain range.

So far, we proved $\mathcal{APX}$-hardness only for the model in which p_{ij} may be infinite, i.e., job j cannot be processed on machine i. It is easy to show that we can replace each infinite processing time in the reduction by a large polynomially bounded processing time P such that the reduction is still valid. In fact, we conjecture that by a slight modification of the reduction one can show that the problem is $\mathcal{APX}$-hard even when we restrict to instances with all mn processing times in $\{1, 2, \dots, P\}$ for some constant P.

3 Approximability

In this section we present two approximation algorithms for the weighted version: $R|pmtn|\sum_j w_j C_j$. The first has an approximation ratio less than 1.81 and is based on the 2-approximation given by Skutella [22]. The second has a ratio less than 1.698 which is obtained by applying the first algorithm and an algorithm by Queyranne and Sviridenko [15] and then taking the best of the two solutions. The presented algorithms are randomized but derandomization is easy.

Preliminaries The *speed* of machine i for job j is $s_{ij} = 1/p_{ij}$ and we say that j is processed with speed s_{ij} at time t if it is processed at time t on machine i. Given a preemptive schedule σ we define $f_j^\sigma(t)$ as the speed at which job j is processed at time t and call f_j^σ the *speed function* of job j in σ. The *mean busy time* M_j^σ of job j (introduced in [5]) is defined as the average time at which it is processed. More precisely,

$$M_j^\sigma := \int_0^T f_j^\sigma(t)t\,dt, \tag{12}$$

where T is any upper bound on the completion time of j. An equivalent formulation is the following. For $\alpha \in (0, 1]$ let $C_j^\sigma(\alpha)$ be the moment in time when an α-fraction of j is completed; α-points were first used in the context of approximation in [9].

$$M_j^\sigma := \int_0^1 C_j^\sigma(\alpha)\, d\alpha,$$

Lawler and Labetoulle (Theorem 2 in [13]) showed that there is always an optimal schedule with $O(m^2 n)$ preemptions. We may therefore restrict our analysis to schedules with a polynomially bounded number of preemptions and $C_j^\sigma(\alpha)$ is well-defined in that case. Further, we define for given σ the *process time* P_j^σ as the total time that job j is processed.

New lower bound One difficulty in improving on the approximation ratio of 2 is to find new lower bounds. If all machines are identical then $M_j^\sigma + P_j^\sigma/2$ is a lower bound on the completion time C_j^σ of a job j. This bound no longer holds in the unrelated machine model (for example, take $m = n = 2$ and let $w_1 = z$, $p_{11} = 1$, $p_{21} = z$, $w_2 = 1$, $p_{12} = 1/z$, and $p_{22} = z$, where z is a large number. Then it is optimal to schedule job 1 on machine 1 in the interval [0, 1] and job 2 starts on machine 2 and continues on machine 1 from time 1. If $z \to \infty$, then $C_2^\sigma, M_2^\sigma, P_2^\sigma \to 1$). The ratio's in [15,17] and [22] are based on the weaker bound $C_j^\sigma \geq M_j^\sigma$ and the bound $C_j^\sigma \geq P_j^\sigma$. The next theorem gives a new relation between the three concepts: completion time, mean busy time and process time. The proof is given in Sect. 3.1.1

Theorem 2 *For any instance I of $R|pmtn|\sum w_j C_j$ and feasible preemptive schedule σ for I there exists a feasible preemptive schedule σ' for I such that $M_j^{\sigma'} + P_j^{\sigma'} < 1.81 C_j^\sigma$ for any job j.*

The intuition behind this theorem is as follows. In the extreme case that $M_j^\sigma + P_j^\sigma \approx 2C_j^\sigma$, as is true for $j = 2$ in the example above, we have $M_j^\sigma \approx C_j^\sigma$ and $P_j^\sigma \approx C_j^\sigma$. But then, a small fraction of the job is accountable for most of the processing time. By stretching the whole schedule by a factor slightly more than 1 we can remove that small fraction from the schedule. The processing time is reduced significantly while there is only little increase in completion and mean busy time. The technique of stretching and rescheduling part of the jobs has been used before. See for example [16].

Corollary 1 (First lower bound) *Let OPT_I be the optimal value for instance I of $R|pmtn|\sum w_j C_j$. Then*

$$\min_\sigma \left(\sum_j w_j (P_j^\sigma + M_j^\sigma) \right) < 1.81 \text{OPT}_I. \tag{13}$$

3.1 The first algorithm

The algorithm in this section is based on the 2-approximation by Skutella [22]: First, a convex quadratic program is solved and then the solution is rounded randomly. The only real difference with the algorithm in [22] is the objective function. It will become obvious that the same result can be obtained by using the interval-indexed LP-formulation by Schulz and Skutella [17]. However, if we had presented our proof based on [17] then it is not that obvious that the formulation of [22] gives the same results. Another reason to use [22] is that it has a nice compact formulation and the rounding is slightly easier. Of course, quadratic programs cannot necessarily be solved exactly in polynomial time but for our analysis it is enough to know that they can be solved up to an arbitrary small additive number $\epsilon > 0$.

Next, we give a concise description of the algorithm, we show that the new objective function remains convex, and we show how Corollary 1 leads to the improved ratio.

To simplify notation we introduce for each machine i a total order $\prec_i$ on the set of jobs by setting $j \prec_i k$ if $w_j/p_{ij} > w_k/p_{ik}$ or if $w_j/p_{ij} = w_k/p_{ik}$ and $j < k$. Consider the following quadratic program (QP). Here x_{ij}, corresponds with the fraction of job j processed on machine i.

$$\text{(QP) minimize } \sum_{j=1}^{n} w_j \left(P_j^{\text{QP}} + M_j^{\text{QP}}\right)$$

$$\text{subject to } P_j^{QP} = \sum_{i=1}^{m} x_{ij} p_{ij} \qquad \text{for all } j, \tag{14}$$

$$M_j^{QP} = \sum_{i=1}^{m} x_{ij} \left(\sum_{k \prec_i j} x_{ik} p_{ik} + x_{ij} p_{ij}/2 \right) \qquad \text{for all } j,$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \text{for all } j,$$

$$x_{ij} \geq 0 \qquad \text{for all } i, j. \tag{15}$$

Lemma 5 *Let* $\text{O}\small{\text{PT}}^{QP}$ *be the* optimal value *of program QP for some given instance* I. *Then, for any feasible schedule* σ *for* I,

$$\text{O}\small{\text{PT}}^{QP} \leq \sum_j w_j (P_j^\sigma + M_j^\sigma).$$

Proof Given σ, let x_{ij} be the fraction of j that is processed on machine i in σ. Then for any j, P_j^{QP} is exactly the process time of job j in σ. Next we show that $\sum_{j=1}^n w_j M_j^{QP} \leq \sum_{j=1}^n w_j M_j^\sigma$. Given the values x_{ij}, the total weighted mean busy time on machine i is minimized by placing jobs in order of non-increasing w_j/p_{ij} [6]. Thus, $\sum_{j=1}^n w_j M_j^\sigma$ is at least the total weighted mean busy time of this *pseudo* schedule $\hat{\sigma}$ (we call $\hat{\sigma}$ a pseudo schedule since a job is possibly processed simultaneously on several machines). Let $s = \sum_{k \prec_i j} x_{ik} p_{ik}$ be the starting time of job j in $\hat{\sigma}$. Then, by (12), the contribution of machine i to the mean busy time of j is

$$\int_s^{s+x_{ij}p_{ij}} 1/p_{ij} t \, dt = \frac{1}{2p_{ij}} \left[t^2\right]_s^{s+x_{ij}p_{ij}}$$

$$= s x_{ij} + x_{ij}^2 p_{ij}/2 = x_{ij} \left(\sum_{k \prec_i j} x_{ik} p_{ik} + x_{ij} p_{ij}/2 \right).$$

Taking the sum over all jobs and machines shows that the total weighted completion time of $\hat{\sigma}$ is exactly $\sum_{j=1}^n w_j M_j^{QP}$. □

We rewrite (QP) in matrix notation and adopt the notation from [22]. Define $c_{ij} = w_j p_{ij}$ and, for any $i \in \{1, 2, \ldots, m\}$, let $i_1, i_2, \ldots, i_n$ be the indices of the jobs according to $\prec_i$. Let $c, x \in \mathbb{R}^{mn}$ be, respectively, the vector of all c_{ij}'s and x_{ij}'s ordered by increasing i and then, for each i, in the order $\prec_i$. The $mn \times mn$ matrix A is given by

$$A = \begin{pmatrix} A_1 & 0 & 0 & 0 \\ 0 & A_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_m \end{pmatrix}, \text{ where } A_i = \begin{pmatrix} w_{i_1} p_{i_1} & w_{i_2} p_{i_1} & \cdots & w_{i_n} p_{i_1} \\ w_{i_2} p_{i_1} & w_{i_2} p_{i_2} & \cdots & w_{i_n} p_{i_2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{i_n} p_{i_1} & w_{i_n} p_{i_2} & \cdots & w_{i_n} p_{i_n} \end{pmatrix}.$$

Precisely, the entry (j, k) of submatrix A_i is $w_{i_{\max\{j,k\}}} p_{i_{\min\{j,k\}}}$. Now we can rewrite (QP) as

$$\begin{aligned} &\text{minimize } c^T x + \frac{1}{2} x^T A x \\ &\text{subject to } \sum_{i=1}^{m} x_{ij} = 1, && \text{for all } j, \\ &\qquad\qquad x \geq 0. && (16) \end{aligned}$$

The quadratic program used in [22] has objective function $\frac{1}{2} c^T x + \frac{1}{2} x^T A x$ and this is the only difference. It was shown in that paper that matrix A is positive semidefinite.

Lemma 6 (Skutella [22]) *Matrix A is positive semidefinite.*

The quadratic program (QP) is not a proper relaxation of $R|pmtn|\sum_j w_j C_j$ in the sense that the optimal value may be strictly larger than the optimal value of the scheduling problem. However, the advantage of this objective function is that we do not lose anything in the next step of the algorithm: randomized rounding.

Randomized rounding

Given a solution for (QP) with values x_{ij}, ($i \in \{1, 2, \ldots, m\}$, $j \in \{1, 2, \ldots, n\}$), we assign each job j independently at random to one of the machines, where job j is assigned to machine i with probability x_{ij}. On each machine i we place the assigned jobs in the order $\prec_i$. The obtained schedule is non-preemptive and the expected completion time $\mathbb{E}[C_j]$ of job j is easily expressed in the values x_{ij} as follows.

$$\begin{aligned} \mathbb{E}[C_j] &= \sum_{i=1}^{m} x_{ij} \Big(\sum_{k \prec_i j} x_{ik} p_{ik} + p_{ij} \Big) \\ &= \sum_{i=1}^{m} x_{ij} p_{ij} + \sum_{i=1}^{m} x_{ij} \sum_{k \prec_i j} x_{ik} p_{ik} \\ &< P_j^{\mathrm{QP}} + M_j^{\mathrm{QP}}. \end{aligned}$$

Corollary 2 *Algorithm 1 is a randomized* 1.81*-approximation algorithm.*

Algorithm 1: A randomized 1.81-approximation algorithm

```
1 Solve (QP)→ values $x_{ij}$;
2 Assign job $j$ to machine $i$ with probability $x_{ij}$. On each machine $i$ place jobs in order $\prec_i$.
```

Proof From Corollary 1 and Lemma 5 we have that for any instance I

$$\text{O}\small{\text{PT}}^{QP} < 1.81 \min_{\sigma} \sum_j w_j C_j^{\sigma}.$$

On the other hand, randomized rounding gives

$$\mathbb{E}\left[\sum_j w_j C_j\right] = \sum_j w_j \mathbb{E}[C_j] < \sum_j w_j \left(P_j^{\text{QP}} + M_j^{\text{QP}}\right) = \text{O}\small{\text{PT}}^{QP}.$$

□

The following corollary follows immediately from the non-preemptiveness of the constructed schedule.

Corollary 3 *For any instance of $R|pmtn|\sum w_j C_j$, the value of the optimal non-preemptive schedule is no more than* 1.81 *times the value of the optimal preemptive schedule.*

The algorithm can be derandomized using the method of conditional probabilities. All we need to show is that we can bound the expected completion times on the condition that some jobs are already assigned to machines. In that case, the matrix A in (16) is adjusted by removing the rows and columns of the already assigned jobs and therefore it remains positive semidefinite. The changes in the vector c are slightly more involved but have no effect on the convexity of the adjusted program. The procedure is exactly the same as in [22] and we refer to that paper for more details.

Instead of using the quadratic program QP one may just as well use the interval-indexed LP-formulation and randomized rounding technique given by Schulz and Skutella [17]. This LP uses variables y_{ijt} indicating the fraction of interval t on machine i used by job j. Given an LP-solution, a job is assigned to interval t on machine i with probability $y_{ijt}\delta_t/p_{ij}$, where δ_t is the length of interval t. Given the assignment, a feasible schedule is obtained by processing on each machine the jobs as early as possible in the order of the intervals assigned to the jobs. We refer to [17] for a precise formulation. It is shown that any solution to the LP can be rounded such that the expected completion time of any job is at most $P_j^{LP} + M_j^{LP}$, where P_j^{LP} and M_j^{LP} are the process time and mean busy time of the job in the (pseudo) schedule defined by the LP-solution. On the other hand, for any feasible schedule σ with corresponding values y_{ijt} we have $P_j^{LP} = P_j^{\sigma}$ and $M_j^{LP} \leq (1+\epsilon)M_j^{\sigma}$, where ϵ depends on the size of the intervals. To turn the $(2+\epsilon)$-approximation into a 1.81-approximation, all we need to do is to change the objective function to $P_j^{LP} + M_j^{LP}$.

3.1.1 Proof of Theorem 2.

The proof is constructive. Note that the construction is not part of the algorithm. Given any value $\beta > 1$ and schedule σ we can construct a new schedule σ' by removing a $1 - 1/\beta$ fraction of every job and stretching the remaining schedule by a factor β. Then, the fraction of any job in the new schedule will be exactly $1/\beta \cdot \beta = 1$. Note that we can decide for each job *independently* what parts to remove. The optimal choice for β that follows from our analysis below is $\beta = 1.089$. It would be enough to do the analysis just for this value but the analysis is easier to read by keeping β variable. Moreover, we will do a similar analysis in Sect. 3.2.3 with a different value of β.

We now describe how σ' is constructed. For each job j we keep that $1/\beta$ fraction that has smallest process time in σ. We say that this is the *good part* of job j (this part may not be unique but we fix one arbitrarily). Then we stretch the whole schedule by a factor β.

We shall only analyze one arbitrary job j in one arbitrary schedule σ and this enables us to simplify the notation. We remove the index j and σ from the notation and refer to job j simply as *the job* (as was done in the $\mathcal{APX}$-hardness proof). For example, M, and P are the mean busy time and process time of *the* job (j) in *the* schedule (σ) and $f(t)$ is its speed function. Since there is only one job we assume w.l.o.g. that the completion time of the job is 1. Let $\mathcal{T} \subset [0, 1]$ be the (set of) time interval(s) in which the good part of the job is processed. Denote the total length of $\mathcal{T}$ by $\tau = |\mathcal{T}|$, for some $\tau < 1$. Note that

$$0 < \tau \leq 1/\beta < 1$$

since we picked the fastest $1/\beta$ fraction and assumed $C = 1$. By stretching σ by a factor β, the set $\mathcal{T}$ is mapped onto a set $\mathcal{T}'$ of size $|\mathcal{T}'| = \beta\tau$. In the new schedule, the job is processed completely during $\mathcal{T}'$.

Let $M_{\mathcal{T}}$ be the contribution of the good part of the job in the mean busy time, i.e.,

$$M_{\mathcal{T}} := \int_{t \in \mathcal{T}} f(t) t \, dt,$$

Claim 2 *Let P' and M' be, respectively, the total process time and mean busy time of the job in σ'. Then,*

$$P' = \beta\tau \text{ and } M' = \beta^2 M_{\mathcal{T}} \leq \beta - \frac{(\beta^2 - \beta)\tau^2}{2(1 - \tau)}.$$

Proof The process time of the good part of the job in σ is τ. After stretching it has length $\beta\tau$. To see the second equality, note that there is a factor β increase in time and a factor β increase in the fraction considered (from $1/\beta$ to 1). More precisely,

$$M' = \int_{y \in \mathcal{T}'} f(y/\beta) y \, dy = \int_{t \in \mathcal{T}} f(t) \beta t \, d(\beta t) = \beta^2 M_{\mathcal{T}}.$$

To prove the inequality of the claim we maximize $M_{\mathcal{T}}$ over all speed functions g for which the length of the good part is τ. In other words, we solve the following problem: Given $\beta > 1$ and $0 < \tau \leq 1/\beta$ find (an upper bound for)

$$\Omega_{\beta,\tau} := \sup_{g} \int\limits_{t \in \mathcal{T}_g} g(t)\, t \, dt,$$

where g is non-negative and is defined on [0, 1] and $\int_{t=0}^{t=1} g(t)\, dt = 1$ and the good part is $\mathcal{T}_g$ with $|\mathcal{T}_g| = \tau$. Clearly, we may restrict to speed functions g that are non-decreasing. Consequently, $\mathcal{T}_g = [1-\tau, 1]$ and we can rephrase our problem as finding (an upper bound for)

$$\Omega_{\beta,\tau} := \sup_{g} \int\limits_{t=1-\tau}^{1} g(t)\, t\, dt \tag{17}$$

such that

$$(i) \int\limits_{t=0}^{t=1-\tau} g(t)\, dt = 1 - 1/\beta \quad \text{and} \quad (ii) \int\limits_{t=1-\tau}^{1} g(t)\, dt = 1/\beta.$$

First we fix some constant $s \geq 0$ and add to (i) and (ii) the restriction that $g(1-\tau) = s$. The value s is bounded by the monotonicity of g. Constraints (i) and (ii) imply

$$\frac{1 - 1/\beta}{1 - \tau} \leq s \leq \frac{1/\beta}{\tau}. \tag{18}$$

Since we assumed that the function is non-decreasing, the supremum follows from the extreme situation where the speed is s everywhere on $[1-\tau, 1]$ and the remaining fraction $1/\beta - s\tau$ is done instantly at time 1. The supremum in that case is

$$s\tau \cdot (1 - \tau/2) + (1/\beta - s\tau) \cdot 1 = 1/\beta - s\tau^2/2. \tag{19}$$

Expression (19) is maximized for minimum value of s. We substitute the lower bound from (18) and get

$$M_{\mathcal{T}} \leq \Omega_{\beta,\tau} = 1/\beta - \frac{(1 - 1/\beta)\tau^2}{2(1-\tau)}.$$

□

From Claim 2 we see that for given value $\beta > 1$ we have

$$P' + M' \leq \max_{\tau \in (0, 1/\beta]} F_\beta(\tau), \text{ where } F_\beta(\tau) = \left[\beta\tau + \beta - \frac{(\beta^2 - \beta)\tau^2}{2(1-\tau)}\right]. \tag{20}$$

We shall prove that for fixed $\beta > 1$ the function $F_\beta(\tau)$ is maximized for

$$\tau^*(\beta) = 1 - \sqrt{(\beta - 1)/(\beta + 1)}. \tag{21}$$

Before proving (21), let us see how this completes the proof of Theorem 2. Minimizing $F_\beta(\tau^*(\beta))$ over β one finds that the minimum is attained for $\hat{\beta} \approx 1.089$. Note that we do not need to prove here that this is the minimum. It suffices to verify that $F_{1.089}(\tau^*(1.089)) < 1.81$, which completes the proof of Theorem 2.

Now we prove (21). The function is continues and differentiable for $\tau < 1$ and goes to $-\infty$ for $\tau \to -\infty$ or $\tau \uparrow 1$.

$$\frac{\partial F_\beta(\tau)}{\partial \tau} = \beta - (\beta^2 - \beta)\frac{\tau - \tau^2/2}{(1-\tau)^2}. \tag{22}$$

Setting this to zero gives

$$\begin{aligned} 0 &= \beta(1-\tau)^2 - (\beta^2 - \beta)(\tau - \tau^2/2) \\ &= \beta - (\beta^2 + \beta)\tau + (\beta^2/2 + \beta/2)\tau^2. \end{aligned}$$

This is a quadratic function of τ. Let us write $u = \beta^2 + \beta$. Then

$$\begin{aligned} \tau &= (-b \pm \sqrt{b^2 - 4ac})/2a, \ \text{ where } a = u/2, \ b = -u, \ \text{ and } c = \beta. \\ \tau &= (u \pm \sqrt{u^2 - 2u\beta})/u = 1 \pm \sqrt{1 - 2\beta/u} = 1 \pm \sqrt{(\beta - 1)/(\beta + 1)}. \end{aligned}$$

Since $\tau < 1$ we conclude that $\tau^*(\beta) = 1 - \sqrt{(\beta - 1)/(\beta + 1)}$.

3.2 A second algorithm: take the best of two solutions

The $(2+\epsilon)$-approximation algorithm by Queyranne and Sviridenko [15] can be used to improve on the constant of 1.81. Remember that $C_j^\sigma(\alpha)$ is defined as the moment in time when an α-fraction of j is completed.

Algorithm QS: Find a $(1+\epsilon)$-approximation σ for minimizing $\sum_j w_j M_j$ by solving an LP. Then, take $1/\gamma$ at random from $(0, 1]$ with distribution function $f(x) = 2x$ and for each job j remove all that is scheduled after time $C_j^\sigma(1/\gamma)$. Stretch the remaining schedule by a factor γ.

It turns out that algorithm QS performs quite well for the instance where Algorithm 1 attains its worst case ratio. Hence, it is interesting to consider the following algorithm.

Algorithm 2: A randomized 1.698-approximation algorithm

```
1 Apply Algorithm 1 and Algorithm QS and take the best solution.
```

Analysis of Algorithm QS

The sum of weighted mean busy times can be minimized approximately by solving a linear program and then turning it into a feasible schedule by using the polynomial time algorithm of Gonzalez and Sahni [7] for the preemptive open shop problem. Let us denote for any integer $t \geq 1$, the time interval $[t-1, t]$ as slot t. In the LP below, the variable x_{ijt} represents the fraction of job j processed on machine i within slot t.

$$\begin{aligned}
\text{(LP) minimize } & \sum_{j=1}^{n}\left(w_j \sum_{i=1}^{m}\sum_{t=1}^{T} x_{ijt}\cdot t\right) & & \\
& \sum_{i=1}^{m}\sum_{t=1}^{T} x_{ijt} = 1 & & \text{for all } j, \\
& \sum_{i=1}^{m} x_{ijt}p_{ij} \leq 1 & & \text{for all } j, t, \\
& \sum_{j=1}^{n} x_{ijt}p_{ij} \leq 1 & & \text{for all } i, t, \\
& x_{ijt} \geq 0 & & \text{for all } i, j, t. \qquad (23)
\end{aligned}$$

The first constraint ensures that each job is completely processed, the second constraint ensures that each job spends at most one time unit in each slot and the third constraints implies that in each slot the total processing time is at most one on each machine. If the values x_{ijt} are taken from a feasible schedule, then $M_j \geq \sum_{i=1}^{m}\sum_{t=1}^{T} x_{ijt}(t-1)$ for any j since each fraction x_{ijt} is processed in the interval $[t-1, t]$. Now, assume w.l.o.g. that $p_{ij} \geq 2/\epsilon$ for all i, j. Then, for any job j

$$\sum_{i=1}^{m}\sum_{t=1}^{T} x_{ijt}t = 1 + \sum_{i=1}^{m}\sum_{t=1}^{T} x_{ijt}(t-1) \leq 1 + M_j \leq (1+\epsilon)M_j.$$

Hence, the optimal LP-value is at most $1+\epsilon$ times the minimum total weighted mean busy time. For each slot t, the values x_{ijt} define an instance of the open shop problem $O|pmtn|C_{\max}$, and the classical algorithm of Gonzalez and Sahni [7] can be used to find a feasible preemptive schedule in each slot. The resulting schedule has a total weighted mean busy time of at most the LP-value and, hence, is a $1+\epsilon$-approximation for minimizing total weighted mean busy time. We assumed here that processing times are polynomially bounded but this assumption can be avoided by using intervals of geometrically increasing length instead of intervals of length 1. See [15] for more details.

The next step of the QS-algorithm is to apply algorithm Slow-Motion of Schulz and Skutella [16] to the optimal LP-schedule σ. That means, the LP-schedule is converted by stretching it by a random factor γ and processing each job within its new time slots as early as possible. Let C_j^{γ} be the completion time of j in the final schedule given the value of γ. Then $C_j^{\gamma} = \gamma \cdot C_j^{\sigma}(1/\gamma)$. Now, if $1/\gamma$ is chosen at random from (0, 1] with distribution $f(x) = 2x$ then

$$\mathbb{E}[C_j] = \int_{x=0}^{1} C_j^{1/x} f(x)dx = \int_{x=0}^{1} \frac{1}{x} C_j^{\sigma}(x) f(x)dx = 2 \int_{x=0}^{1} C_j^{\sigma}(x)dx = 2M_j^{\sigma}.$$

Let ρ be an optimal schedule for minimizing the total weighted completion time. Then, the expected sum of weighted completion times given by this algorithm is

$$\sum_j w_j \mathbb{E}[C_j] = 2\sum_j w_j M_j^{\sigma} \leq 2(1+\epsilon)\sum_j w_j M_j^{\rho}. \tag{24}$$

The proof follows now from $M_j^{\rho} < C_j^{\rho}$. Derandomization is easy by enumerating over γ.

3.2.1 Comparing the two algorithms

In this paragraph we give some intuition for why Algorithm 2 performs strictly better than Algorithm 1. Consider an optimal schedule ρ w.r.t. minimizing the total weighted completion time for some instance I. Let $\lambda_I \leq 1$ be such that

$$\sum_j w_j M_j^{\rho} = \lambda_I \sum_j w_j C_j^{\rho}. \tag{25}$$

From (24) it follows that the approximation ratio of the QS algorithm for this instance is at most $2(1+\epsilon)\lambda_I$. Hence, it performs good for small values of λ_I. On the other hand, Algorithm 1 performs good if λ_I is close to 1. We shall give a short (hand-waving) explanation for this. In the analysis of Algorithm 1 we defined ρ' from ρ by keeping the *good part* of jobs and removing the rest and stretching the schedule by a factor $\hat{\beta}$. If $\lambda_I \approx 1$ then, for an *average job* j we have $M_j^{\rho} \approx C_j^{\rho}$ which means that almost the whole job j is processed in a small interval. This implies $M_j^{\rho'} \approx \hat{\beta} M_j^{\rho}$ and $P_j^{\rho'} \approx 0$. Then, $M_j^{\rho'} + P_j^{\rho'} \approx \hat{\beta} M_j^{\rho} < \hat{\beta} C_j^{\rho}$ and the approximation factor for the *average* job j is approximately $\hat{\beta} \approx 1.089$.

3.2.2 A second lower bound

Theorem 3 *For any instance I and feasible preemptive schedule σ for I there exists a feasible preemptive schedule σ' for I such that $P_j^{\sigma'} + M_j^{\sigma'} < 2\gamma C_j^{\sigma} - 2M_j^{\sigma}$ for any job j, where $\gamma = 1.698$.*

A proof is given in Sect. 3.2.3. Now let σ be an optimal schedule for minimizing the total weighted completion time of I and let $\mathrm{O{\scriptsize PT}}_I$ be its value and let σ' be as defined by Theorem 3. Then, taking the weighted sum over all jobs gives

$$\sum_j w_j P_j^{\sigma'} + \sum_j w_j M_j^{\sigma'} < 2\gamma \mathrm{OPT}_I - 2\sum_j w_j M_j^{\sigma}.$$

This leads to the following lower bound on the optimal value.

Corollary 4 (second lower bound) *Let* OPT_I *be the optimal value for instance* I *of* $R|pmtn|\sum w_j C_j$*. Then*

$$2\min_{\sigma}\sum_j w_j M_j^{\sigma} + \min_{\sigma'}\sum_j w_j \left(P_j^{\sigma'} + M_j^{\sigma'}\right) < 2\gamma \mathrm{OPT}_I,$$

where the minima are taken over all feasible schedules σ *and* σ'*.*

The left minimum is an upper bound on the value given by Algorithm QS, up to a factor $1+\epsilon$ [see (24)], and the right minimum is an upper bound on the value given by Algorithm 1. By choosing ϵ small enough we get the the next corollary.

Corollary 5 *Algorithm 2 gives a* γ*-approximation for* $R|pmtn|\sum w_j C_j$*, were* $\gamma =$ 1.698.

3.2.3 Proof of Theorem 3.

The proof is similar to that of Theorem 2 but the optimization is more tedious. The schedule σ' is defined exactly the same. That means that σ' is defined from σ by keeping only the good part of jobs and stretching the remaining schedule by a factor $\beta > 1$. The value of β that results from the analysis is now $\beta = 1.29$. The optimization over β is omitted here. It is enough to verify the equations starting from (31) where we plug in the value of β.

Again, we consider one arbitrary job j and drop the index j and σ in the sequel. We assume that the completion time of the job is $C = 1$. Also, $\mathcal{T}$ is defined as before as the (set of) time interval(s) in which the good part of the job is processed. Again, $\tau = |\mathcal{T}|$ and we define $M_{\mathcal{T}}$, M' and P' as before. We will show that for $\beta = 1.29$ we have

$$2M + (M' + P') < 2 \cdot 1.698. \tag{26}$$

We have $P' = \beta\tau$, $M = \int_{t=0}^{1} f(t) \cdot t\, dt$. To show an upper bound on the left side of (26) we may assume that the speed function f is non-decreasing. Then, $M' = \beta^2 \int_{t=1-\tau}^{1} f(t) \cdot t\, dt$ and the problem becomes to find an (upper bound) for the value

$$\Psi_{\beta,\tau} := \sup_g \left(2\int_{t=0}^{1-\tau} g(t)\cdot t\, dt + (2+\beta^2)\int_{t=1-\tau}^{1} g(t)\cdot t\, dt \right) + \beta\tau, \tag{27}$$

where g is a speed function that satisfies

$$(i) \int_{t=0}^{t=1-\tau} g(t)\, dt = 1 - 1/\beta \ \text{ and } (ii) \int_{t=1-\tau}^{1} g(t)\, dt = 1/\beta.$$

Let $s > 0$ be a constant. First we find the supremum under the restriction that $g(1-\tau) = s$. The constraints (i) and (ii) are the same as in the proof of Theorem 2 and imply again (18):

$$s_L \leq s \leq s_U, \text{ where } s_L = \frac{1-1/\beta}{1-\tau} \text{ and } s_U = \frac{1}{\beta\tau}.$$

The second integral in this objective (27) is exactly the objective function that we had in the proof of Theorem 2. From (19) we see that the second integral satisfies

$$(2+\beta^2) \int\limits_{t=1-\tau}^{1} g(t) \cdot t \, dt \leq (2+\beta^2)(1/\beta - s\tau^2/2). \tag{28}$$

The first integral in (27) is maximized in the case where the speed is 0 in the interval $[0, t^*]$ and s in $[t^*, 1-\tau]$ where t^* is such that the complete fraction $1-1/\beta$ is processed between 0 and $1-\tau$. Hence, $t^* = 1-\tau-(1-1/\beta)/s$ (note that $s \geq s_L$ implies $t^* \in [0, 1-\tau)$). The value of the first integral in (27) in that case is

$$\begin{aligned} &2(1-1/\beta)(t^*+1-\tau)/2 \\ &= 2(1-1/\beta)(1-\tau-(1-1/\beta)/s+1-\tau)/2 \\ &= 2(1-1/\beta)(1-\tau-(1-1/\beta)/(2s)). \end{aligned}$$

We add this bound to (28) and to $\beta\tau$ and denote this by

$$G_{\beta,\tau}(s) := \left(2+\beta^2\right)\left(\frac{1}{\beta} - \frac{s\tau^2}{2}\right) + 2\left(1-\frac{1}{\beta}\right)\left(1-\tau-\frac{1-1/\beta}{2s}\right) + \beta\tau.$$

Then,

$$\Psi_{\beta,\tau} \leq \max\{G_{\beta,\tau}(s) \mid s_L \leq s \leq s_U\}.$$

We simplify notation and write $G_{\beta,\tau}(s) = a_1 + a_2\tau + a_3\tau^2 s + a_4/s$, where

$$a_1 = 2+\beta, \; a_2 = \beta + 2/\beta - 2, \; a_3 = -1-\beta^2/2, \text{ and } \; a_4 = -(1-1/\beta)^2.$$

Note that function $G_{\beta,\tau}(s)$ is strictly increasing for $0 < s \leq s' := \sqrt{\frac{a_4}{a_3}}/\tau$ and strictly decreasing for $s \geq s'$. Hence,

$$\Psi_{\beta,\tau} \leq \begin{cases} G_{\beta,\tau}(s_L) & \text{if} \quad s' \leq s_L \\ G_{\beta,\tau}(s') & \text{if} \quad s_L \leq s' \leq s_U \\ G_{\beta,\tau}(s_U) & \text{if} \quad s_U \leq s'. \end{cases} \tag{29}$$

We can exclude the third case since

$$s' \geq s_U \Rightarrow \sqrt{\frac{a_4}{a_3}} \geq 1/\beta \Rightarrow \beta^2|a_4| \geq |a_3| \Rightarrow (\beta-1)^2 \geq 1+\beta^2/2 \Rightarrow \beta \geq 4.$$

Hence, for $\beta = 1.29$ the inequality $s' \leq s_U$ is always true. Further,

$$s_L \leq s' \Leftrightarrow \frac{1-1/\beta}{1-\tau} \leq \sqrt{\frac{a_4}{a_3}}/\tau \Leftrightarrow \frac{\tau}{1-\tau} \leq \sqrt{\frac{a_4}{a_3}}/(1-1/\beta).$$

The righthand side equals $1/\sqrt{-a_3}$. Hence,

$$s_L \leq s' \Leftrightarrow \frac{\tau}{1-\tau} \leq \frac{1}{\sqrt{-a_3}} \Leftrightarrow \tau \leq \tau^* := \frac{1}{\sqrt{-a_3}+1}.$$

Now (29) becomes

$$\Psi_{\beta,\tau} \leq \begin{cases} G_{\beta,\tau}(s_L) & \text{if} \quad \tau \geq \tau^* \\ G_{\beta,\tau}(s') & \text{if} \quad \tau \leq \tau^*. \end{cases} \tag{30}$$

At this point we removed the parameter s from the optimization and we continue with maximizing the righthand side of (30) over τ. We start with the second bound of (30). Substituting s' gives

$$G_{\beta,\tau}(s') = a_1 + (a_2 + 2\sqrt{a_3 a_4})\tau.$$

For $\beta = 1.29$ we have

$$a_1 = 3.29,\, a_2 \approx 0.8403,\, a_3 \approx -1.8320,\, a_4 \approx -0.05053,\, \tau^* = 0.4248 \tag{31}$$

This gives

$$G_{\beta,\tau}(s') \approx 3.29 + 0.2317\tau$$

For $\tau \leq \tau^*$ the maximum is attained for $\tau = \tau^*$.

$$G_{\beta,\tau^*}(s') \approx 3.29 + 0.2317 \cdot 0.4248 < 3.39.$$

Now we maximize the first bound of (30) over τ. This value will be slightly larger and hence determines the approximation ratio. Substituting $s_L = (1-1/\beta)/(1-\tau)$ gives

$$G_{\beta,\tau}(s_L) = a_1 + a_2\tau + a_3(1-1/\beta)\frac{\tau^2}{1-\tau} + \frac{a_4}{1-1/\beta}(1-\tau)$$

We rewrite this in a form that is easy to maximize.

$$
\begin{aligned}
G_{\beta,\tau}(s_L) &= a_1 + a_2\tau + a_3(1-1/\beta)\left(\frac{1}{1-\tau} + 1 - \tau\right) + \frac{a_4}{1-1/\beta}(1-\tau)\\
&= \left(a_3(1-1/\beta) + \frac{a_4}{1-1/\beta} - a_2\right)(1-\tau) + a_3(1-1/\beta)\frac{1}{1-\tau} + C_\beta\\
&= a(1-\tau) + \frac{b}{1-\tau} + C_\beta,
\end{aligned}
$$

where $a \approx -1.4769$ and $b \approx -0.4118$ and C_β is independent of τ. Now it is easy to see that it is maximized for $1-\tau' = \sqrt{b/a} \approx 0.5280$, i.e., $\tau' \approx 0.4720$. Note that $\tau' \in [\tau^*, 1/\beta]$. Hence,

$$
\max_{\tau\in[\tau^*,1/\beta]} G_{\beta,\tau}(s_L) = G_{\beta,\tau'}(s_L) < 3.396 = 2\cdot 1.698. \tag{32}
$$

This completes the proof of Theorem 3.

References

1. Afrati, F., Bampis, E., Chekuri, C., Karger, D., Kenyon, C., Khanna, S., Milis, I., Queyranne, M., Skutella, M., Stein, C., Sviridenko, M.: Approximation schemes for minimizing average weighted completion time with release dates. In: FOCS '99, pp. 32–44 (1999)
2. Afrati, F., Milis, I.: Designing PTASs for min-sum scheduling problems. Discret. Appl. Math. **154**, 622–639 (2006)
3. Bruno, J., Coffman Jr., E.G., Sethi, R.: Scheduling independent tasks to reduce mean finishing time. Commun. ACM **17**, 382–387 (1974)
4. Chekuri, C., Khanna, S.: A PTAS for minimizing weighted completion time on uniformly related machines. In: ICALP. Lecture Notes in Computer Science, vol. 2076, pp. 848–861 (2001)
5. Goemans, M.: Improved approximation algorithms for scheduling with release dates. In: Proceedings of 8th Symposium on Discrete Algorithms, pp. 591–598. New Orleans, Louisiana, USA (1997)
6. Goemans, M., Queyranne, M., Schulz, A., Skutella, M., Wang, Y.: Single machine scheduling with release dates. SIAM J. Discret. Math. **15**, 165–192 (2002)
7. Gonzalez, T., Sahni, S.: Open shop scheduling to minimize finish time. J. ACM **23**, 665–679 (1976)
8. Graham, R., Lawler, E., Lenstra, J., Kan, A.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discret. Math. **5**, 287–326 (1979)
9. Hall, L., Shmoys, D.B., Wein, J.: Scheduling to minimize average completion time: off-line and on-line algorithms. In: Proceedings of 7th Symposium on Discrete Algorithms, pp. 142–151 (1996)
10. Hoogeveen, J., Schuurman, P., Woeginger, G.: Non-approximability results for scheduling problems with minsum criteria. INFORMS J. Comput. **13**, 157–168 (2001)
11. Horn, W.: Minimizing average flow time with parallel machines. Oper. Res. **21**, 846–847 (1973)
12. Kann, V.: Maximum bounded 3-dimensional matching is MAX SNP-complete. Inf. Process. Lett. **37**, 27–35 (1991)
13. Lawler, E., Labetoulle, J.: On preemptive scheduling of unrelated parallel processors by linear programming. J. ACM **25**, 612–619 (1978)

14. Queyranne, M., Schulz, A.S.: Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. SIAM J. Comput. **35**(5), 1241–1253 (2006). doi:10.1137/S0097539799358094
15. Queyranne, M., Sviridenko, M.: A (2+epsilon)-approximation algorithm for the generalized preemptive open shop problem with minsum objective. J. Algorithms **45**(2), 202–212 (2002)
16. Schulz, A., Skutella, M.: Random-based scheduling: new approximations and LP lower bounds. pp. 119–133 (1997)
17. Schulz, A., Skutella, M.: Scheduling unrelated machines by randomized rounding. SIAM J. Discret. Math. **15**, 450–469 (2002)
18. Schuurman, P., Woeginger, G.: Polynomial time approximation algorithms for machine scheduling: ten open problems. J. Sched. **2**, 203–213 (1999)
19. Sitters, R.: Approximability of average completion time scheduling on unrelated machines. In: Halperin, D., Mehlhorn, K. (eds.) Proceedings of 16th European Symposium on Algorithms. Lecture Notes in Computer Science, vol. 5193, pp. 768–779. Springer (2008)
20. Sitters, R.: Complexity of preemptive minsum scheduling on unrelated parallel machines. J. Algorithms **57**(1), 37–48 (2005)
21. Skutella, M., Woeginger, G.: A PTAS for minimizing the weighted sum of job completion times on parallel machines. In: Proceedings of 31st Symposium Theory of Computing, pp. 400–407 (1999)
22. Skutella, M.: Convex quadratic and semidefinite programming relaxations in scheduling. J. ACM **48**(2), 206–242 (2001). doi:10.1145/375827.375840