Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 8(3): 7061–7082.
DOI: 10.3934/math.2023356
Received: 04 October 2022
Revised: 29 November 2022
Accepted: 08 December 2022
Published: 11 January 2023

Research article

On preemptive scheduling on unrelated machines using linear programming

Nodari Vakhania*

Centro de Investigación en Ciencias, Universidad Autónoma del Estado de Morelos, Cuernavaca, Morelos, México

* **Correspondence:** Email: nodari@uaem.mx.

Abstract: We consider a basic preemptive scheduling problem where n non-simultaneously released jobs are to be processed by m unrelated parallel machines so as to minimize maximum job completion time. An optimal LP-solution has been used to construct an optimal preemptive schedule for simultaneously released jobs in time $O(n^3)$. We propose fast $O(m)$ time algorithm that finds an optimal schedule in case the above LP-solution possesses "small enough" number of non-zero elements. We propose another linear program for non-simultaneously released jobs and show how an optimal schedule can be constructed also in time $O(m)$ from the optimal solution to that linear program. Based on another stronger linear program formulation, we extend the earlier known schedule construction procedure for non-simultaneously released jobs. The procedure is important, in particular, because there may exist no optimal schedule that agrees with an optimal LP-solution. An optimal LP-solution imposes a number of preemptions, and additional preemptions may occur during the schedule construction process, a job might be forced to be split on the same machine. We show that if no job split is allowed, even a restricted version of the problem on three unrelated machines is NP-hard. As a result, we obtain that, given an optimal LP-solution, it is NP-hard to find an optimal schedule that agrees with that LP-solution. As another side result, we obtain that it is NP-hard to find an optimal schedule with at most $m - 1$ preemptions.

Keywords: scheduling; unrelated machines; release time; linear programming; time complexity
Mathematics Subject Classification: 68Q17, 90B35, 90C05

1. Introduction

Scheduling unrelated machines to minimize maximum job completion time (the so-called makespan) is a well-known optimization problem. In a group of *unrelated* machines, a machine i has no universal speed characteristic (machine speed is job dependent), in contrast to a group of *uniform* machines, where each machine is characterized by a universal speed that extends to a whole set of jobs. A group of machines with the same speed is commonly referred to as a group of *identical machines* (the speed of every machine is the same for every job). The scheduling problem that we consider here is commonly abbreviated as $R|r_j; pmtn|C_{\max}$ (we use a standard three-field notation for scheduling problems, where the first, the second and the third fields specify machine environment, job parameters with some specific problem restrictions (if any), and the objective function, respectively). In this problem we are given n jobs to be performed by m parallel unrelated machines. Job j becomes available at its (integer) *release time* r_j and it requires an (integer) *processing time* p_{ij} on machine i, $i = 1, \dots, m$ (for the sake of simplicity, we will refer to jobs and machines by their corresponding indexes). A job, being processed by a machine, can be interrupted and resumed later on the same or on any other machine. In a *feasible schedule* a machine can process at most one job at a time and a job can be processed by at most one machine at a time (i.e., two parts of the same job cannot be processed in parallel by different machines), every job j is assigned to a machine no earlier than at time r_j and it is completely processed, i.e., $\sum_{i=1}^{m} t_{ij}/p_{ij} = 1$, where t_{ij} is the total amount of time that machine i spends on job j in that schedule. The objective is to find an *optimal schedule*, a feasible one in which the maximum job (machine) completion time $C_{\max}$ is the minimum possible.

In this paper we deal with the above described basic problem $R|r_j; pmtn|C_{\max}$ and its variations. In the restricted assignment problem $R|p_{ij} \in \{p_j, \infty\}; r_j; pmtn|C_{\max}$, a subset of allowable machines is specified for each job j, so that this job can only be processed in a fixed amount of time p_j on any machine from that subset; the processing time of job j on any other machine is an enough large magnitude such that job j cannot be assigned to any of these machines in any optimal schedule. In another variation of the basic problem, the part of each job assigned to a machine is to be performed without any interruption on that machine. We will give a more complete description of this latter version a bit later in this section. Before that, we overview briefly the earlier known related work.

A brief overview of relevant literature. Non-preemptive scheduling on already two identical machines of simultaneously released jobs $P||C_{\max}$ is NP-hard (reduction from the PARTITION problem). This is in contrast to the preemptive case, even on a group of uniform machines $Q|pmtn|C_{\max}$ Gonzalez and Sahni [1]). The setting with non-simultaneously released jobs on uniform machines $Q|r_j, pmtn|C_{\max}$ remains polynomially solvable Labetoulle et al. [2], as well as the version with unrelated machines but simultaneously released jobs $R|pmtn|C_{\max}$ Lawler and Labetoulle [3]. The authors in [3] give a linear programming formulation of the problem and show how an optimal solution to this LP can be used to construct an optimal solution to the latter problem. We are not aware of any previously existing work for preemptive scheduling of non-simultaneously released jobs to minimize the makespan. Lawler and Labetoulle [3] also use linear programming for the solution of a related problem $R|pmtn|L_{\max}$ with the objective to minimize the maximum job lateness $L_{\max}$. The best known polynomial-time approximation algorithms for non-preemptive problem $R||C_{\max}$ are based on rounding of an optimal LP-solution, see also [4–6]. A feasible LP-solution specifies parts of jobs to be processed by each machine; i.e., it *distributes* job parts to machines without specifying the starting times of the assigned job parts. If in a given LP-distribution, different parts of the same job are assigned to different machines, then this job will be preempted in any feasible schedule *respecting* that distribution: in such a schedule, the processing time of every job on every machine is determined by the former distribution. The scheduling stage is used to convert an LP-distribution to a feasible schedule respecting that distribution. This stage determines the start time of each assigned job part on the corresponding machine so that the same job is processed by at most one machine at a time.

Lawler and Labetoulle [3] adopted an open shop scheduling method of Gonzalez and Sahni [7] to construct an optimal preemptive schedule respecting an optimal LP-distribution, an optimal solution to the linear program presented also in [3]. The time complexity of the schedule construction procedure from [3] can roughly be bounded from the above by $O(n^3)$. Potts [4] (and later Lenstra et al. [5] and Shchepin and Vakhania [6]) also used linear programming in a 2-approximation algorithm for the non-preemptive case $R||C_{\max}$. The algorithm from [4] has polynomial dependence on n, but the dependence on m is exponential. Later Lenstra et al. [5] developed another 2-approximation algorithm that avoids an exponential-time dependence on m, and Shchepin and Vakhania [6] improved the approximation factor to $2 - 1/m$.

Our contributions. The algorithms from [4] and [6] are essentially based on the fact that an optimal distribution to the linear program from [4] has an enough small number of non-zero variables that yields at most $m - 1$ preempted jobs. This property may also hold for other linear programs that we consider here, but in principal, optimal LP-distributions to these linear programs may yield more than $m - 1$ preempted jobs. We present a fast $O(m)$ time algorithm in Section 3.1 that finds an optimal solution to the preemptive problem $R|pmtn|C_{\max}$ (with simultaneously released jobs) if the condition holds for an optimal distribution to linear program from [3]. In Section 3.2 we give a linear programming formulation of problem $R|r_j; pmtn|C_{\max}$ with non-simultaneously released jobs and specify how the procedure of Section 3.1 can be extended to find an optimal solution also in time $O(m)$ to this problem, given that the condition now holds for an optimal distribution to this new linear program.

If in an LP-distribution different parts of the same job are assigned to different machines, this job will have the corresponding preemptions in any feasible schedule respecting that distribution. Moreover, during the construction the latter schedule, additional preemptions of this job may occur: a part of it assigned to some machine might be forced to be *split* in further smaller parts to be processed independently of each other on that machine (so a split is a special kind of preemption, every split is a preemption but not every preemption is a split). A preemption, that is not a split, cannot be avoided whenever the corresponding schedule is obtained from the corresponding LP-distribution, whereas any job split is avoidable. No-split is an important restriction in a number of applications where it is undesirable to interrupt a currently running job on a machine due to additional machine reset and setup costs. We abbreviate by $R|r_j; pmtn; \quad no - split|C_{\max}$ the no-split version of problem $R|r_j; pmtn|C_{\max}$ (where the entire job part assigned to a machine cannot be split, i.e., this part is to be processed continuously on that machine), and we abbreviate by $R|p_{ij} \in \{p_j, \infty\}; r_j; pmtn; \quad no - split|C_{\max}$ the no-split version of the restricted problem $R|p_{ij} \in \{p_j, \infty\}; r_j; pmtn|C_{\max}$.

A natural question arises, if there is a polynomial-time algorithm that constructs an optimal solution without any split; i.e., if every job part assigned to a machine can be processed without any further interruption on that machine. This question was positively answered in the case of uniform machines and also for two unrelated machines. Gonzalez and Sahni [1] proposed an $O(n \log n)$ time algorithm for the no-split problem on uniform machines, and Gonzalez, Lawler and Sahni [8] showed that the no-split problem on two unrelated machines can be solved in linear time.

As we show in Section 4, the no-split problem for non-simultaneously released jobs, already on three unrelated machines, becomes NP-hard, even for the restricted assignment problem $R3|p_{ij} \in \{p_j, \infty\}; r_j; pmtn; \quad no - split|C_{\max}$. As a result, we obtain that, given an optimal LP-distribution, it is NP-hard to find an optimal schedule to problem

$R3|p_{ij} \in \{p_j, \infty\}; r_j; pmtn; \ \ no-split|C_{\max}$ that respects that LP-distribution. As another side result, we obtain that it is NP-hard to find an optimal preemptive schedule with at most $m-1$ preemptions to that problem. This result extends, in some sense, an earlier known result that scheduling identical machines with at most $m-2$ preemptions is NP-hard [9].

Unfortunately, our results of Section 5 imply that an optimal schedule to problem $R|p_{ij} \in \{p_j, \infty\}; r_j; pmtn|C_{\max}$ not necessarily respects an optimal LP-distribution. Moreover, there may exist no optimal schedule respecting an optimal LP-distribution. This it true for the liner program from [3], for the linear from Section 4 and also for a stronger linear program that we propose in Section 5. In Section 6 we extend the schedule construction procedure from [3] for non-simultaneously released jobs using an optimal distribution to our linear program from Section 5. The extended procedure constructs an optimal schedule respecting any feasible (not necessarily optimal) LP-distribution in polynomial time. The procedure is important, in particular, because of the result from Section 5 that no optimal schedule may respect an optimal LP-distribution.

Finally, we note that the use of linear programming is not restricted solely to scheduling problems on unrelated machines (where, as already mentioned, it remains the most efficient tool for both, exact and approximation solution), but it is also used to tackle more complex multiobjective shop scheduling problems. For example, Foumani and Smith-Miles [10] represent flow-shop scheduling problems as mixed integer linear programs and deal with by-objective criteria to balance between the minimization of makespan and carbon emissions. They call such bi-objective setting green flow-shop, since it aims not only to maximize the economical benefit (by minimizing the makespan) but also tries to keep the environment clean. Gong et al. [11] consider another green multiobjective shop scheduling problem where the labor cost is also involved.

2. A closer look at the earlier related work

In this section we describe in more detail the earlier obtained results on which our results rely. First, we present linear programs from the earlier cited references that have been used for scheduling unrelated machines. The following linear program LP1($C_{\max}$) was successfully used for an approximate solution of the non-preemptive version of the problem $R||C_{\max}$ with simultaneously released jobs, first in [4] and later in [6]:

Minimize $C_{\max}$
Subject to

$$\sum_{j=1}^{n} x_{ij} p_{ij} \le C_{\max}, \quad i = 1, \dots, m, \tag{2.1}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad x_{ij} \ge 0, \quad i = 1, ..., m, \quad j = 1, \dots, n. \tag{2.2}$$

In this linear program entry $x_{ij} = t_{ij}/p_{ij}$ represents the part of job j to be processed by machine i, for $j = 1, \dots, n$ and $i = 1, \dots, m$. These entries define the corresponding *distribution* of job parts on machines. Unlike a schedule which is a mapping that assigns to each job specific time interval(s) on one or more machines, a distribution, instead of these time intervals, defines only their lengths on the corresponding machines. Because of real assignment variables, a distribution may split a job in different parts and assign these parts to different machines. Note that a distribution involves no start times and only assigns job parts to machines (hence, there is an infinite number of feasible schedules respecting a given distribution). In particular, a solution to a linear program is a distribution that explicitly indicates which fraction of each job is assigned to each machine. We refer the reader to [6] for related formal definitions, concepts and properties.

As earlier noted, distribution to linear program $LP1(C_{\max})$ has a nice property that it possesses a large amount of integer (zero) entries so that it yields at most $m - 1$ preempted jobs. Then such distribution can be rounded to (an integer) feasible approximate non-preemptive solution to problem $R||C_{\max}$, as suggested by Potts [4], where a complete enumeration of at most $m - 1$ preempted jobs on m machines is carried out. This results in a 2-approximation solution in time, polynomial in n and exponential in m. Using a modified linear program combined with a binary search, a rounding that guarantees a 2-approximation solution in polynomial (in both n and m) time was achieved Lenstra et al. [5]. A new method of rounding an optimal distribution to linear program $LP1(C_{\max})$ proposed in Shchepin and Vakhania [6] yielded an improvement of the bound 2 to $2 - 1/m$. This latter bound is the best possible that can be obtained by rounding a distribution to a feasible non-preemptive schedule [6].

In an optimal distribution to linear program LP1($C_{\max}$), the total length of the parts of a job assigned to the machines can be longer than the minimized magnitude $C_{\max}$. Hence, a feasible preemptive schedule respecting that distribution may have the makespan larger than $C_{\max}$, a reason why it is not beneficial to use it for the solution of the preemptive case $R|pmtn|C_{\max}$. Linear program LP2($C_{\max}$) from Lawler and Labetoulle [3] bounds the total length of all the assigned parts of each job to different machines. It is the above specified linear program LP1($C_{\max}$) complemented by an additional set of the following n restrictions, one restriction for each job:

$$\sum_{i=1}^{m} x_{ij}p_{ij} \leq C_{\max}, \quad j = 1, \ldots, n. \tag{2.3}$$

Because of the additional n restrictions (2.3), the total number of basic (non-zero) variables is no more bounded by $m - 1$ (as we describe in Section 3, in case an optimal distribution to linear program LP2($C_{\max}$) yields no more than $m - 1$ preemptions, it can still be transformed to an optimal preemptive schedule with at most $m - 1$ preemptions). An optimal distribution to linear program $LP2(C_{\max})$ can be transformed to an optimal schedule to problem $R|pmtn|C_{\max}$ in polynomial time. Lawler and Labetoulle [3] adopted open shop scheduling technique from Gonzalez and Sahni [7] for constructing an optimal feasible schedule from an optimal distribution to linear program LP2($C_{\max}$) (note that an open shop instance can be already seen as a distribution). We describe this method in more detail in Section 6.

Suppose we have a feasible schedule for an instance of problem $R|pmtn|C_{\max}$ respecting an optimal distribution $\{x_{ij}\}$ to linear program LP2($C_{\max}$) with the makespan

$$C_{\max} = \max\{\max_i \sum_{j=1}^{n} x_{ij}p_{ij}, \max_j \sum_{i=1}^{m} x_{ij}p_{ij}.\}. \tag{2.4}$$

Then this schedule is clearly optimal. Such an optimal schedule is constructed in Lawler and Labetoulle [3].

As we will see in Section 5, for non-simultaneously released jobs, the bound (2.4) is no more attainable; i.e., for a given instance of problem $R|r_j, pmtn|C_{\max}$, there may exist no feasible schedule with makespan $C_{\max}$ respecting a given optimal distribution to linear program LP2($C_{\max}$) (e.g., consider an optimal distribution in which the sum of the job parts assigned to some machine is $C_{\max}$ but no job assigned to that machine is released at time 0). Furthermore, no feasible schedule respecting that optimal distribution might be optimal. This assertion is true not only for linear program LP2($C_{\max}$), but also for more enhanced linear programs that we give later on.

3. Scheduling distributions with limited number of preemptions

In this section we show how an optimal schedule can be constructed in case an optimal distribution to linear program LP2($C_{\max}$) has no more than m preempted job parts. In the next subsection we present fast $O(m)$ time procedure the case of simultaneously released jobs, and in the following subsection we introduce a new liner program and extend the procedure for the case where jobs have different release times.

3.1. Simultaneously released jobs

Suppose thus an optimal distribution to linear program LP2($C_{\max}$) yields no more than $m-1$ preemptions, i.e., there are at most m (preempted) job parts, at most one of it on a machine. We can convert such distribution to an optimal preemptive schedule for problem $R|pmtn|C_{\max}$ with at most $2m-4$ preemptions using the following procedure.

Step 0. Construct an optimal distribution $\{x_{ij}\}$ to linear program LP2($C_{\max}$).

Step 1. Sort the preempted jobs in distribution $\{x_{ij}\}$ in non-increasing order of their total processing times (the total processing time of job j in distribution $\{x_{ij}\}$ is $\sum_{i=1}^{m} x_{ij}p_{ij}$).

Repeat the Step 2 until all the jobs from the ordered list are considered:

Step 2. Let $i_1, \dots, i_k$, $i_1 < \cdots < i_k$, be the indices of the machines to which the parts of the next job from the list are assigned in distribution $\{x_{ij}\}$;
schedule the first part of job j at time 0 on machine i_1, schedule its second part at the completion time of the first part on machine i_2, and so on, schedule the last kth part of that job at the completion time of the preceding $(k-1)$th part on machine i_k

Step 3. Schedule the remaining entire jobs in the remained idle time slots of the partial schedule of Step 2 from time 0 in any order without creating an idle time in between the jobs; in case an overlapping of an entire job j with an earlier included preempted job part occurs, interrupt job j at the starting time of the former job part and resume it at its completion time.

Observation 1. *The above procedure finds an optimal schedule with at most* $2m-4$ *preemptions in time* $O(m)$.

Proof. First, note that at Step 2, no conflicts between the preempted parts of different jobs will occur since there is at most one preempted job part on any machine. Hence, the constructed schedule is feasible. By the construction, due to inequalities (1) and (2.3), the completion time of the last scheduled part of any job j cannot be larger than $C_{\max}$. Hence, the constructed schedule is also optimal. By our condition, distribution $\{x_{ij}\}$ yields at most $m-1$ preemptions. At Step 3 some additional preemptions may occur. Since there is at most one preempted job part on any machine, no split may occur on machines 1 and m, whereas at most one split may occur on the remaining $m - 2$ machines (for at most one entire job per machine) at Step 3. Hence, there are at most $2m - 4$ preemptions (note that this bound will be attained when there is only one preempted job distributed among all the m machines). As to the time complexity, since the total number of the preempted jobs to be processed in bounded from the above by m, the whole procedure takes time $O(m)$. □

3.2. Non-simultaneously released jobs

We extend the above procedure for the case of non-simultaneously released jobs. This time, we shall consider an optimal distribution to a new linear program LP3($C_{\max}$) that we introduce in this section.

An optimal distribution to linear program LP2($C_{\max}$) "implicitly assumes" that job parts can be assigned to the machines at arbitrary time moments. This does not apply if jobs are non-simultaneously released. Linear program LP3($C_{\max}$) is more "flexible" since it takes into account job release times. It is a modified version of linear program LP2($C_{\max}$) in which inequalities (2.3) are replaced by the following set of inequalities:

$$r_j + \sum_{i=1}^{m} x_{ij}p_{ij} \leq C_{\max}, \quad j = 1, \dots, n. \tag{3.1}$$

Note that linear program LP3($C_{\max}$) correctly reflects the desired restriction imposed by the release time of each job. Let us now consider an optimal distribution to this linear program (instead of the optimal distribution to linear program LP2($C_{\max}$)), and suppose that it satisfied the same condition as linear program of Section 4.1. We easily extend now the procedure from Section 4.1 for non-simultaneously released jobs. We again sort job parts in ascending order of the corresponding machine indexes. Instead of starting each first part of the next job from the list at time 0, it is scheduled at the release time of that job. The entire (non-preempted) jobs are scheduled in non-decreasing order of the their release times on each machine. Due to the modified inequalities (3.1) and the fact that there is at most one preempted job part on any machine, similarly as in the procedure of Section 4.1, no preempted job part will complete later than at time $C_{\max}$. Feasibility conditions are similarly verified and the time complexity remains to be $O(m)$.

4. NP-hardness results

Recall that the fast algorithms described in the previous section may create up to $m - 3$ splits. If however no splits are allowed, then the no-split problem in the restricted assignment setting already on 3 machines $R3|p_{ij} \in \{p_j, \infty\}; r_j; pmtn;\ no - split|C_{\max}$ becomes NP-hard:

Theorem 1. *$R|p_{ij} \in \{p_j, \infty\}; r_j; pmtn;\ nosplit|C_{\max}$ is NP-hard.*

Proof. We show that the decision version of $R|p_{ij} \in \{p_j, \infty\}; r_j; pmtn;\ nosplit|C_{\max}$ is NP-complete using the reduction from an NP-complete PARTITION problem. In the PARTITION problem we have k items with sizes $\{z_1, \dots, z_k\}$. We are asked if there exists a subset of these k items that sup up to $M/2$, where $M = \sum_{i=1}^{k} z_i$. Let us consider an arbitrary instance of the PARTITION problem with P_1 and P_2 being a partition of the k items with $\sum_{l \in P_1} z_l = \sum_{l \in P_2} z_l = M/2$, a solution to that PARTITION instance ($P_1 \cup P_2 = \{z_1, \dots, z_k\}$, $P_1 \cap P_2 = \emptyset$).

We now construct an instance of the decision version of our scheduling problem such that this instance will have a "yes" answer if and only if the corresponding PARTITION instance has a "yes" answer. Our scheduling instance consists of $4+k$ jobs on three machines, where $Z_1, \dots, Z_k$ are *partition* jobs. All partition jobs are released at time 3. The processing times of these jobs are such that $p_{2Z_j} = 2z_j/M$ and $p_{iZ_j} = \infty$, for $j = 1, \dots, k$ and $i = 1, 3$ (note that the total length of the partition jobs is 2). The processing times of the remaining four jobs J^1, J^2, J^3, J^4 are defined as follows. $p_{11} = p_{21} = p_{31} = 6$, $p_{13} = 2$, $p_{24} = 3$ and $p_{32} = 5$. All the unspecified processing times are infinity (large enough numbers). Job J^3 is released at time 4 and the remaining jobs are released at time 0 (except the partition ones which are released at time 3).

As it is easy to see, the processing time 6 of job J^1 is a lower bound on the optimal schedule makespan. In a schedule with this makespan (see Figure 1):

On machine 1, job J^3 cannot be started earlier than at its release time 4 and can be completed at time 6, hence job 1 starts at its release time 0 and competes at time 4. On machine 3, job J^2 is to occupy 5 time units. Hence, only one time unit is left where job J^1 can be processed on that machine. Therefore, one unit of time of job J^1 is to be processed on machine 2. All partition jobs are to be processed also on machine 2. None of the partition jobs can start earlier than at time 3 whereas job J^4 is to occupy 3 time units on machine 2. Since machine 1 is running job J^1 in interval $[0, 4)$, job J^4 is needs to occupy the first 3 time units on machine 2 and is to be followed by the partition jobs. Since the assigned to machine 3 part of job J^2 cannot be interrupted on that machine, the remaining unprocessed unit time of job J^1 can only be executed within the interval $[4, 5)$ on machine 2. Hence, exactly the intervals $[3, 4)$ and $[5, 6)$ are left for the partition jobs.

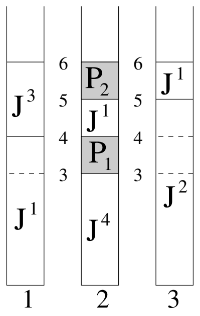


Figure 1. An optimal schedule with makespan 6. Dark regions represent partition jobs.

It should now be apparent that there exist a schedule of length 6 if and only if the partition instance has a "yes" answer, i.e., there exists a partition of set $\{z_1, \dots, z_k\}$ into sets P_1 and P_2 with equal length 1. In other words, an optimal schedule with makespan 6 provides a solution to PARTITION, and vice-versa, if PARTITION has a solution then the above optimal schedule can be constructed in polynomial time. □

Corollary 1. *Given an instance of problem $R|p_{ij} \in \{p_j, \infty\}; r_j; pmtn|C_{\max}$, it is NP-hard to find an optimal solution with at most $m-1$ preemptions.*

Proof. By Theorem 1, $R|r_j, pmtn - nosplit|C_{\max}$ is NP-hard. No split on any machine yields at most m preempted job parts (one job part on every machine), hence at most $m-1$ preemptions. □

Corollary 2. *Given an optimal distribution, it is NP-hard to find a schedule for problem* $R|p_{ij} \in \{p_j, \infty\}; r_j; pmtn; nosplit|C_{\max}$ *with the minimum makespan respecting that distribution.*

Proof. From the proof of Theorem 1, the PARTITION instance is to be solved if job J^2 is not allowed to be split on machine 3. The corollary follows as the distribution respected by the schedule of Figure 1 is optimal. □

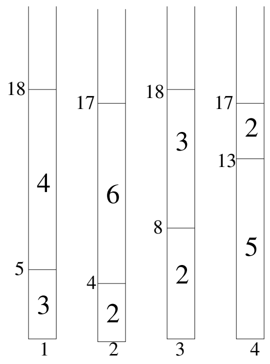
5. Weaknesses of linear programs

In this section we will have a closer look at the inherent relationship between optimal schedules and optimal distributions. For a given feasible schedule to an instance of the scheduling problem $R|r_j; pmtn|C_{\max}$ with makespan $C_{\max}$, let $\{x_{ij}\}$ be the distribution that respects this schedule. The values $C_{\max}$ and $\{x_{ij}\}$ form a feasible solution to linear programs LP3($C_{\max}$) and LP2($C_{\max}$). A less obvious question is, given a distribution to linear program LP3($C_{\max}$), whether there is an optimal schedule respecting that distribution with the makespan $C_{\max}$. Note that, if this assertion is true, then a feasible schedule respecting an optimal distribution will be optimal.

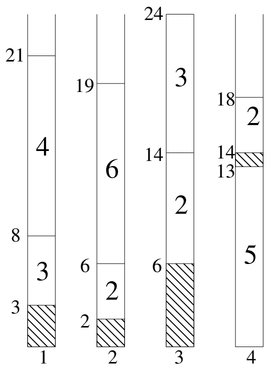
Thus a feasible schedule respecting an optimal distribution is trivially optimal if its makespan is $C_{\max}$. Lawler and Labetoulle [3] constructed schedules respecting optimal distributions to linear program LP2($C_{\max}$) with makespan $C_{\max}$ for simultaneously released jobs. In contrary to the case with simultaneously released jobs, an optimal schedule respecting an optimal distribution to linear program LP3($C_{\max}$) (and linear program LP2($C_{\max}$)) not necessarily attains makespan $C_{\max}$ if jobs are non-simultaneously released. Inequalities (1) do not actually reflect the restrictions imposed by job release times on the completion time of each machine, since it may not be possible to schedule a job assigned to a machine at the earliest idle-time moment on that machine. In particular, restrictions (1) are not necessarily satisfied in an optimal schedule, in which the completion time of a machine may be larger than $C_{\max}$ (recall an earlier mentioned simple scenario where no job assigned to a machine is released at time 0). Even restrictions (3.1) may not be satisfied in an optimal schedule.

We illustrate these points in this section using small problem instances. Our examples illustrate that in an optimal schedule constructed from an optimal distribution to linear program LP3($C_{\max}$), neither restrictions (1) nor restrictions (3.1) might be satisfied (the completion time of at least one job in that schedule can be larger than $C_{\max}$). More importantly, an optimal schedule not necessarily respects an optimal distribution.

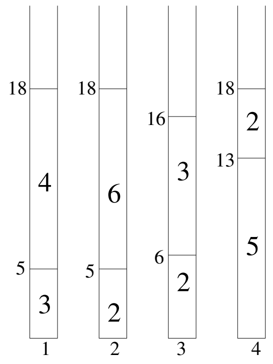
Example 1. Let us consider a problem instance where an optimal distribution assigns just two different job parts to the same machine (hence the schedule construction procedure from Section 4.2 already cannot be applied). We have five jobs $2, \dots, 6$ on four machines $1, \dots, 4$ such that:
$r_3 = 3,\ r_2 = 2,\ r_4 = 8,\ r_5 = 0,\ r_6 = 5$.
$p_{13} = p_{33} = 15$, $p_{22} = p_{32} = p_{42} = 16$, and $p_{14} = p_{26} = p_{45} = 13$. The processing times of these jobs on the remaining machines are infinities (large enough numbers). Note that in any feasible schedule, jobs 4, 6 and 5 can only be processed by machines 1, 2 and 4, respectively. Job 2 is to be distributed among machines 2, 3 and 4, and job 3 is to be distributed on machines 1 and 3.



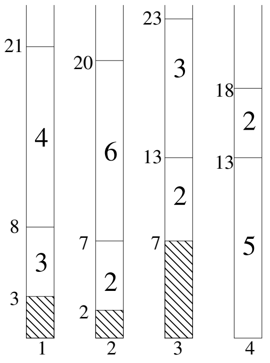
(a) A non-feasible schedule respecting distribution 1.



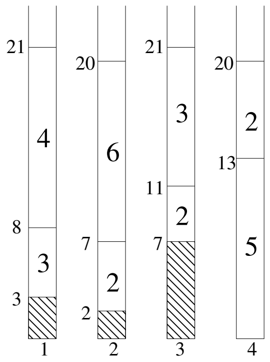
(b) A feasible schedule respecting distribution 1.



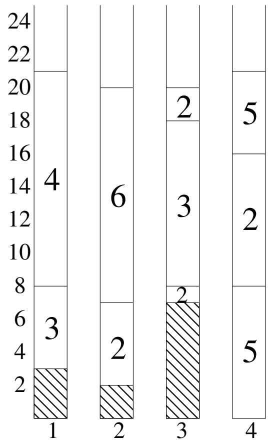
(c) A non-feasible schedule respecting distribution 2.



(d) A feasible schedule respecting distribution 2.



(e) A globally optimal schedule that respects a non-optimal distribution 3.



(f) A globally optimal schedule that respects a non-optimal distribution 4.

Figure 2. Schedules respecting different distributions for the problem instance of Example 1.

There are a few optimal distributions to linear program LP2($C_{\max}$) with $C_{\max} = 18$. We consider two of them which assign (preempted) parts of jobs 3 and 2 to machine 3. In distribution 1, the processing times are as follows:
$t_{13} = 5,\ t_{14} = 13, t_{22} = 4,\ t_{26} = 13, t_{32} = 8,\ t_{33} = 10$ and $t_{45} = 13,\ t_{42} = 4$.

Note that this is not an optimal distribution to linear program LP3($C_{\max}$) due to inequalities (3.1) where $C_{\max} = r_4 + p_{14} = 8 + 13 = 21$ is attained for job 4. A non-feasible schedule respecting distribution 1 is depicted in Figure 2a, and an optimal schedule respecting the same distribution with makespan 24 is depicted in Figure 2b.

An optimal distribution 2 is identical to distribution 1 except that $t_{22} = 5,\ \ t_{32} = 6$ and $t_{42} = 5$ (see Figure 2c). An optimal schedule respecting this distribution has the makespan 23, see Figure 2d.

As can see, distribution 2 possesses better properties, so that an optimal schedule respecting that distribution has a smaller makespan than an optimal schedule respecting distribution 1. At the same time, none of these schedules is (globally) optimal (see below). Moreover, one can easily verify that there exists no optimal schedule respecting any optimal distribution to linear program LP2($C_{\max}$). (From here on we refer to a schedule with minimum makespan respecting a given optimal distribution as an *optimal schedule respecting that distribution*; such schedule, may not be (globally) optimal, i.e., there may exist no optimal schedule respecting this distribution. Moreover, there may exist no optimal schedule respecting any optimal distribution.)

Now we consider a modification of the above considered distributions, a non-optimal distribution 3 with $C_{\max} = 20$, in which $t_{22} = 5$, $t_{32} = 4$, $t_{42} = 7$. A schedule with the makespan 21 respecting this distribution is depicted in Figure 2e. Distribution 3 is not optimal for linear program LP2($C_{\max}$) and it is not feasible to linear program LP3($C_{\max}$) (e.g., $r_4 + t_{14} = 21 > 20$, see inequalities 3.1). It is easy to see that the schedule of Figure 2e is (globally) optimal.

In Figure 2f another optimal schedule with the makespan 21 is depicted. This schedule respects distribution 4 with $C_{\max} = 21$, which is not optimal for linear program LP2($C_{\max}$) but it is (feasible and) optimal for linear program LP3($C_{\max}$) (due to inequalities (3.1), $C_{\max} = 8 + 13 = 21$ is attained for job 4, and 21 is also the load of machine 4). However, this schedule is not feasible for an instance of $R|r_j, pmtn - nosplit|L_{\max}$. Distribution 4 differs from the above optimal distributions on machines 3 and 4. Job processing times are distributed as follows:
$t_{13} = 5,\ t_{14} = 13, t_{22} = 5,\ t_{26} = 13, t_{32} = 3,\ t_{33} = 10$ and $t_{45} = 13,\ t_{42} = 8$. $\square$

As we saw from the above example, optimal schedules respecting different optimal distributions may have different makespan. Guessing an optimal distribution and also a “suitable” linear program is an important and also difficult task. Furthermore, as we argue in the next section, unlikely, there exists a universal “suitable” linear program for the studied scheduling problem, such that an optimal schedule respecting an optimal distribution to that linear program can be guaranteed to be (globally) optimal. Below we give a smaller problem instance illustrating similar points.

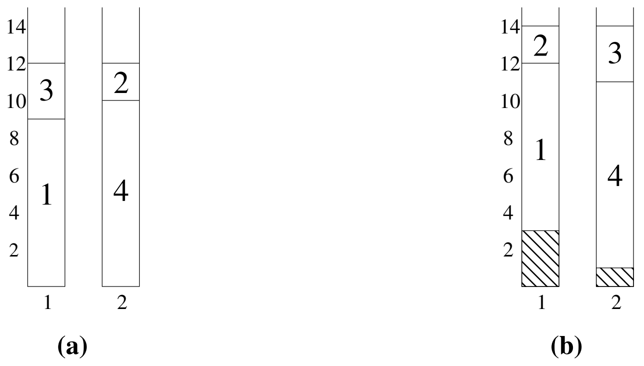


Figure 3. A non-feasible schedule respecting an optimal distribution (a) and an optimal schedule (b).

Example 2. As another example, consider a smaller problem instance with four jobs on two machines. Jobs 1 and 4 are released at times 3 and 1, respectively, and jobs 2 and 3 are released behind these jobs at time, say 5, i.e.,
$r_1 = 3,\ r_2 = 1$ and $r_3 = r_4 = 5$.

Job processing times are as follows:

$p_{11} = 9,\ p_{12} = \infty,\ p_{14} = \infty,\ p_{24} = 10,\ p_{13} = p_{23} = 3$ and $p_{12} = p_{22} = 2$.

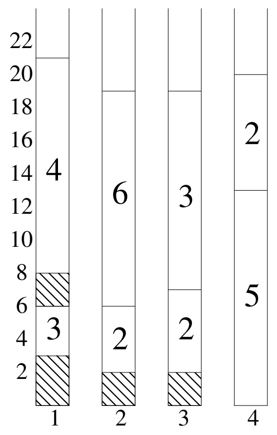
An optimal distribution with $C_{\max} = 12$ defines the processing times $t_{11} = 9,\ t_{13} = 3$ and $t_{24} = 10,\ t_{22} = 2$. This distribution is optimal for both linear programs LP2($C_{\max}$) and LP3($C_{\max}$). A non-feasible schedule with makespan 12 respecting this distribution is depicted in Figure 3a, whereas an optimal feasible schedule with makespan 14 is depicted in Figure 3b. The latter schedule respects another distribution with processing times $t_{11} = 9,\ t_{12} = 2$ and $t_{24} = 10,\ t_{23} = 3$ (in which the roles of jobs 2 and 3 are interchanged) and it is not optimal for linear programs LP2($C_{\max}$) and LP3($C_{\max}$). The latter distribution is however optimal for linear program LP4($C_{\max}$) that we introduce in the next section. □

Linear program LP3($C_{\max}$) properly reflects the desired restriction for each job, but it does not reflect actual restrictions imposed by job release times on machine start times due to the nature of inequalities (1), which deal with a mere sum of processing times of jobs assigned to each machine. This causes potential conflicts at the scheduling stage. As we saw in the previous section, there may exist no optimal schedule to problem $R|r_j; pmtn|C_{\max}$ respecting an optimal distribution to linear program LP3($C_{\max}$). An appropriate modification of restrictions (1) would require a kind of "prediction" of an actual completion time of each machine given the job parts assigned to this and other machines possessing parts of the same jobs. Such a prediction seems to be complicated since it would actually require the outcome of scheduling process on each machine. Nevertheless, we still may make some pertinent assumptions on an optimal scheduling strategy on each machine. In particular, we observe that no avoidable gap is created on any machine in an optimal schedule. It is easy to see that, among the job parts assigned to a machine, the first included one corresponds to an earliest released job in an optimal schedule. In general, whenever an idle time is unavoidable on a machine, an earliest released job is scheduled immediately after that idle time on that machine.

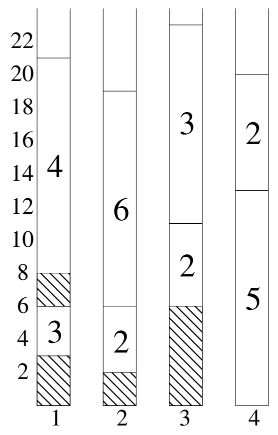
We use these observations in our second linear program LP4($C_{\max}$). Let, $J(> r)$ denote the set of jobs with the release time, no smaller than r. We rewrite m constraints (1) into nm constraints as follows.

$$r_j + \sum_{l \in J(>r)} x_{il} p_{il} \leq C_{\max}, \quad j = 1, \ldots, n, \quad i = 1, \ldots, m \tag{5.1}$$

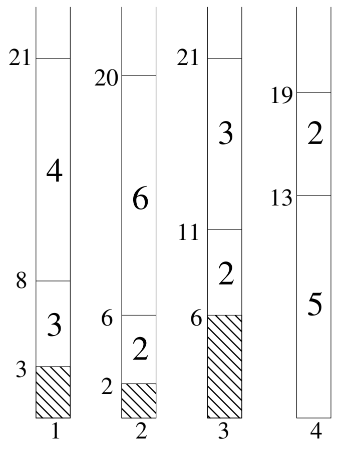
By incorporating these restrictions instead of constraints (1) into linear program LP3($C_{\max}$), we obtain linear program LP4($C_{\max}$).



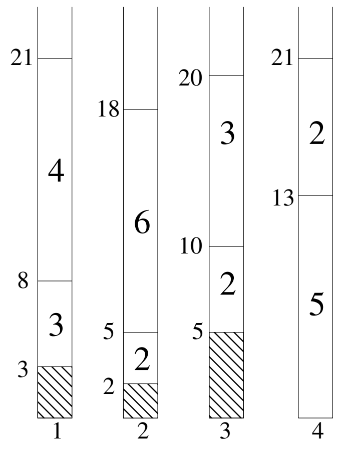
(a) A non-feasible schedule respecting the first optimal distribution.



(b) A feasible schedule respecting the first distribution.



(c) A globally optimal schedule respecting the second optimal distribution.



(d) A globally optimal schedule respecting the third optimal distribution.

Figure 4. Schedules constructed based on distributions to linear program LP4($C_{\max}$).

Example 1 (continuation). Returning to the problem instance of Example 1, we can easily observe that, for an optimal distribution to the new linear program LP4($C_{\max}$), $C_{\max} = 21$. In particular, distribution 4 from the previous section (Figure 2f) is optimal also for linear program LP4($C_{\max}$). There exist other optimal distributions to linear program LP4($C_{\max}$). In one of them,
$t_{13} = 3,\ t_{14} = 13, t_{22} = 4,\ t_{26} = 13, t_{32} = 5,\ t_{33} = 12$ and $t_{45} = 13,\ t_{42} = 7$,
see Figure 4a representing a non-feasible schedule that respects this distribution. A feasible schedule with makespan 23 respecting the same distribution is depicted in Figure 4b. In a slight modification of the latter optimal distribution, job 3 is redistributed on machines 1 and 3 so that its processing time on

machine 1 is increased by 2 and its processing time on machine 3 is decreased by the same amount. This results in a globally optimal schedule with makespan 21 depicted in Figure 4c. In another optimal distribution t_{22} is reduced to 3 and t_{42} is increased to 8. Another globally optimal schedule with makespan 21 respecting the latter optimal distribution is depicted in Figure 4d. □

As we can see, even for a very small sized problem instance, a number of different optimal distributions to the new linear program LP4($C_{\max}$) exist, some of them leading to an optimal schedule and some not. We again need to guess a correct optimal distribution among all possible optimal ones. Moreover, as we show below, not necessarily there exists a globally optimal schedule that respects an optimal distribution to the new linear program LP4($C_{\max}$), as it was the case for liner programs LP2($C_{\max}$) and LP3($C_{\max}$).

Example 1a. Consider a slight modification of the problem instance of Example 1 in which all job parameters remain the same except that $r_4 = 0$. This reduces the makespan of an optimal distribution to linear program LP4($C_{\max}$) from $C_{\max} = 21$ to $C_{\max} = 20$. Now job 4 can be partitioned on machine 1 in two parts, hence t_{13} can be increased to 7. An optimal schedule with makespan 20 respecting this optimal distribution is depicted in Figure 5a. □

Example 1b. For the second modification of Example 1, let $r_4 = 6$. For an optimal distribution to this modified instance $C_{\max} = 20$, which is the completion time of machine 4 (note that the completion time on machine 1, compared to that in the schedule of Figure 4b, is reduced by the length of the gap $[6, 8)$). A non-feasible schedule respecting an optimal distribution is depicted in Figure 5b. An optimal feasible schedule with makespan 23 respecting the same distribution is depicted in Figure 5c. The latter schedule is not globally optimal. An optimal schedule with makespan 21 is illustrated in Figure 5d; this schedule respects a distribution with the same makespan $C_{\max} = 21$. Observe that the latter distribution is not optimal for linear program LP4($C_{\max}$). □

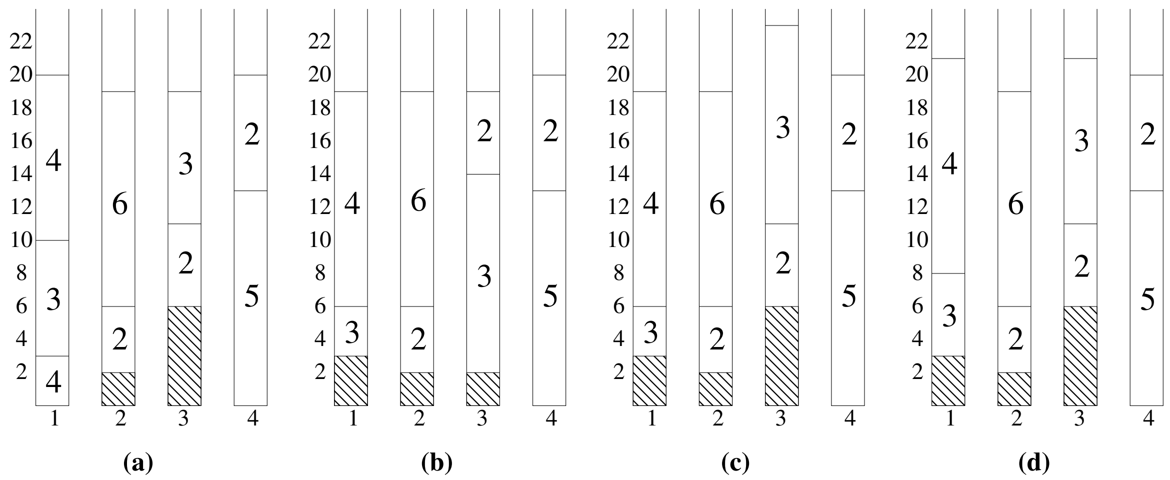


Figure 5. (a): An optimal schedule for the first modified instance; (b): a non-feasible schedule respecting an optimal distribution for the second modified instance; (c): an optimal schedule respecting the latter distribution; (d): and an optimal schedule, that respects a non-optimal distribution.

6. The schedule construction stage

In the next subsection we briefly describe the procedure of Lawler and Labetoulle [3] for the construction of an optimal schedule from an optimal distribution to linear program LP2($C_{\max}$) for problem $R|pmtn|C_{\max}$. In the following subsection 6.2 we show how this procedure can be extended to construct an optimal schedule respecting any feasible distribution to problem $R|r_j; pmtn|C_{\max}$.

6.1. The schedule construction procedure of Lawler and Labetoulle [3]

Lawler and Labetoulle [3] adopted open shop scheduling technique of Gonzalez and Sahni [7] for the construction of an optimal feasible schedule with makespan $C_{\max}$ respecting an optimal distribution to program LP2($C_{\max}$) for simultaneously released jobs (note that an open shop instance can already be seen as a distribution). Recall that an optimal distribution $\{x_{ij}\}$ defines an m x n non-negative processing time matrix $T = \{t_{ij}\} = x_{ij}p_{ij}$.

The so-called *decrementing sets* are iteratively formed from iteration 1 by selecting one entry in each tight row and in each tight column: a row/column is *tight* iff the sum of the entries in that row/column is exactly $C_{\max}$. The initial processing time matrix $T = T^1$, defined by an optimal distribution, is iteratively transformed into the m x n 0-matrix. At each iteration $h > 0$, the matrix T^{h-1} of the previous iteration is updated according to the formed decrementing set D^h at iteration h; in particular, each entry in matrix T^{h-1} corresponding to an element of set D^h is decreased by a suitably chosen (small enough) number δ^h resulting in the updated matrix T^h of iteration h. δ^h is chosen in such a way that the new matrix T^h possesses similar properties as its predecessor matrix T^{h-1}, i.e., the sum of the elements in each row and column of the updated matrix is no larger than $C_{\max} - \sum_{i=1}^{h} \delta^i$. The elements in decrementing set D^h define a partial schedule of iteration h of length δ^h according to the selected portions of processing times.

Let σ^h be the partial schedule generated by iteration h. Schedule σ^h is obtained by a mere merging of the partial schedules of each of the iterations $1, \dots, h$. As a result, the makespan of partial schedule σ^h is $\tau^{h+1} = \sum_{i=1}^{h} \delta^i$. We will refer to τ^{h+1} as the *scheduling time* of iteration $h + 1$, the time moment at which the partial schedule of iteration $h + 1$ starts.

The procedure halts at iteration h such that matrix T^h is a 0-matrix. At that iteration, σ^h is a complete feasible schedule. The optimality of this complete schedule is argued using the fact that its makespan is $C_{\max}$, a lower bound on the optimum schedule makespan. The authors in [3] also argue that in the initial matrix T and in each following matrix T^h there exists a decrementing set using a known Birkhoff and von-Neumann theorem. This theorem states that every doubly stochastic matrix is a linear combination of permutation matrices. By completing matrix T with additional slack rows and columns, an $m+n$ x $m+n$ matrix with the entries of *each* its row and column summing up to ($C_{\max}$) can be obtained. Dividing all the entries of this matrix by ($C_{\max}$), a doubly stochastic matrix is obtained. Then a permutation matrix from the theorem is to define a decrementing set.

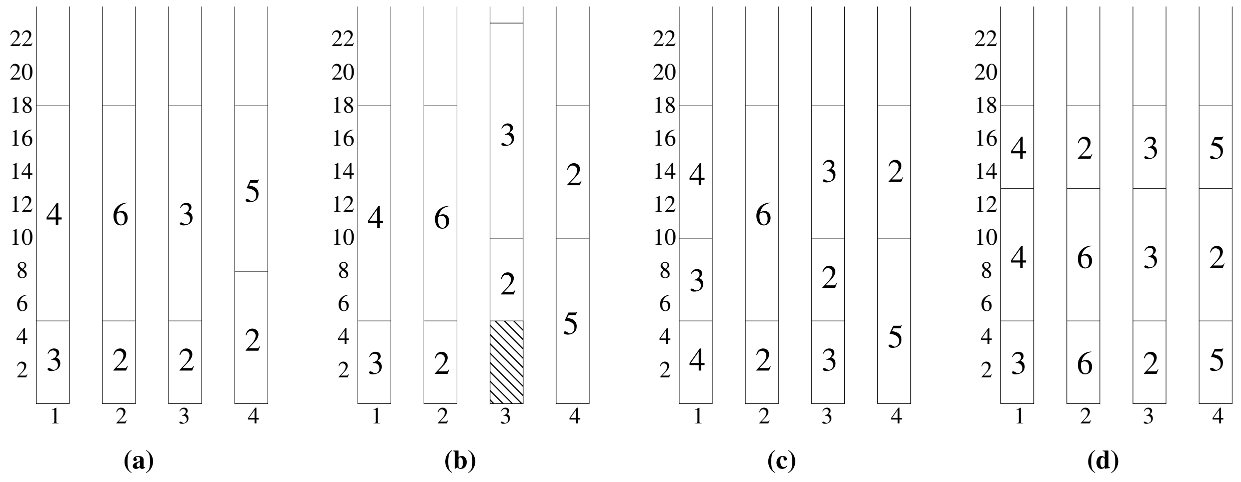


Figure 6. A non-feasible schedule respecting an optimal distribution (a), a feasible non-optimal schedule (b) and two optimal schedules (c) and (d) respecting the same distribution.

Example 3. Let us illustrate the scheduling method of Lawler and Labetoulle on a small instance of problem $R|pmtn|C_{\max}$. It is basically the instance of Example 1 adopted for the case when all jobs are simultaneously released. To maintain $C_{\max} = 18$, we increase the processing time of jobs 2 and 3 to 18 and decrease the processing time of job 5 to 10. It can be easily verified that in an optimal distribution to linear program LP2($C_{\max}$) with $C_{\max} = 18$, we have $t_{13} = 5,\ t_{14} = 13, t_{22} = 5,\ t_{26} = 13,\ t_{32} = 5,\ t_{33} = 13$, and $t_{42} = 8,\ t_{45} = 10$. A non-feasible schedule respecting this distribution is depicted in Figure 6a. This schedule can straightforwardly be converted to a feasible schedule of Figure 6b with makespan 23 by imposing a gap $[0, 5)$ on machine 3. This schedule is not optimal. An optimal one respecting the above optimal distribution is depicted in Figure 6c.

The scheduling method of Lawler and Labetoulle [3] will generate an optimal schedule as follows. The initial matrix $T = T^1$ of job processing times defined by the optimal distribution is represented in the first quarter of Table 1, where the entries corresponding to the elements of the decrementing set D^1 and the following decrementing sets are circled. δ^1 can be chosen to be equal to 5. The updated matrix T^2 and the entries corresponding to the elements of the decrementing set D^2 are shown in the second quarter of Table 1. The first partial schedule with length 5 can be seen as the initial (first) part of the schedule of Figure 6d corresponding to interval $[0, 5)$. The computations in the following iterations $h = 2, 3$ with $\delta^2 = 8$, $\delta^3 = 5$ are reflected in the next quarters of Table 1 and in the upper parts in the schedule of Figure 6d. Although schedule δ^3 of Figure 6d with makespan $C_{\max} = 5 + 8 + 5 = 18$ is somewhat similar to that of Figure 6c, it splits job 5 on machine 4 (hence it is not a feasible solution to problem $R|r_j, pmtn - nosplit|L_{\max}$). □

Table 1. Flowchart of the procedure with processing time matrices and the corresponding decrementing sets (marked with circles).

	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6	2	3	4	5	6
1	0	$\textcircled{5}$	13	0	0	0	0	$\textcircled{13}$	0	0	0	0	$\textcircled{5}$	0	0	0	0	0	0	0
2	5	0	0	0	$\textcircled{13}$	5	0	0	0	$\textcircled{8}$	$\textcircled{5}$	0	0	0	0	0	0	0	0	0
3	$\textcircled{5}$	13	0	0	0	0	$\textcircled{13}$	0	0	0	0	$\textcircled{5}$	0	0	0	0	0	0	0	0
4	8	0	0	$\textcircled{10}$	0	$\textcircled{8}$	0	0	5	0	0	0	0	$\textcircled{5}$	0	0	0	0	0	0

6.2. The extended procedure for non-simultaneously released jobs

Now we extend the schedule construction procedure from Section 6.1 for non-simultaneously released jobs. As we saw, an optimal schedule respecting an optimal distribution to any of linear programs that we considered is not necessarily optimal. Moreover, there may exist no globally optimal schedule respecting an optimal distribution, i.e., any such schedule may respect a non-optimal distribution.

Due to the above observations, we aim to find a schedule with the minimum makespan among all schedules respecting a given (not necessarily optimal) distribution, i.e., find an optimal schedule respecting that distribution. We generalize the schedule construction procedure from Section 6.1 maintaining an extra information on which job parts can be scheduled at every scheduling time τ^h. Note that such a care is to be taken only on the first scheduled part of each job. For that, we introduce an additional row 0 in processing time matrices.

Initially, in the extended matrix T^1, the entry t^1_{0j} in column j of row 0 is r_j; iteratively, $t^h_{0j} = \max\{0, r_j - \tau^h\}$. During the scheduling process, we impose an additional restriction that forbids scheduling of job j at time τ^h if $t^h_{0j} > 0$ (independently of whether the corresponding entries are from a tight row or tight column).

As a result, not necessarily a decrementing set will contain one entry from a tight row or a tight column. In particular, set D^h will contain no entry from a (tight) row i if among yet unscheduled job parts assigned to machine i no job is yet released by time τ^{h+1} (i.e., for any positive entry in row i, the corresponding entry in row 0 is positive); likewise, the decrementing set of iteration h will contain no entry from a (tight) column j if job j is not yet released by time τ^{h+1}.

We will refer to a row i from matrix T^h as *ready* at iteration h if it contains a positive entry $t^h_{ij} > 0$ such that $t^h_{0j} = 0$; we will refer to such an entry as *valid* for row i at iteration h.

Let $t^h_{0j} = 0$, and let $M_h(j)$ be the set of machines such that the entry t^h_{ij} is valid for $i \in M_h(j)$. Then the ready rows from set $M_h(j)$ are said to be *conflicting* by job j at iteration h. Note that two or more rows may be conflicting by two or more different jobs.

A decrementing set D^h of iteration h contains one entry from a ready row such that no two entries from the same column are included into that set (since the same job cannot be scheduled on different machines at a time); whenever a ready row i contains an element from a tight column j, the corresponding part of job j can be selected if entry t_{ij} is valid at iteration h.

It might not be possible to select an entry from every ready row at a given iteration h since two or more rows may be conflicting by the same jobs in such a way that all entries cannot be selected. For example, if we have two ready rows with valid entries only in, say, column j, then only one of these entries can be selected; likewise, if there are three ready rows with valid entries only in, say, columns j and j', then only two of these entries, one from column j and the other from column j', can be selected in decrementing set D^h. We break ties by selecting the corresponding entry from row i with the maximum current load, i.e., with the maximum $\sum_{l=1}^{n} t^h_{il}$. If such a row contains several valid entries, then the entry from a column j with the maximum remaining processing time, i.e., with the maximum $\sum_{l=1}^{m} t^h_{lj}$ is selected. Further ties are broken by selecting an entry from column l such that $t^{h-1}_{il} \in D^{h-1}$ (i.e., part of job l was included in the decrementing set of the previous iteration). The latter tie breaking rule avoids unnecessary preemption of an already running job. (At every iteration h, the rows can be considered in non-increasing order of their loads and the corresponding valid entries can be selected from a column j with the maximum remaining processing time.)

Let r be the minimum positive element in row 0 at iteration h in matrix T^h, i.e., $r = \min_j t^h_{0j}$. The jump δ^h at iteration h is now defined as the minimum between δ^h as defined in Lawler and Labetoulle [3] and r.

Since there may exist no more than n district job release times, the extended method yields an additional term n bounding the number of iterations and hence has the same polynomial time complexity as the method of Lawler and Labetoulle [3]. It is not difficult to see that τ^{h^*} is an optimal schedule makespan, where h^* is the last iteration in the procedure.

Note that the extended procedure will work as the one of Section 6.1 from the earliest scheduling time $\tau^{h'}$ such that $t^{h'}_{0j} = 0$, for all $j = 1, \dots, n$, since the corresponding decrementing sets will contains m elements with exactly one element from each tight column (by the construction, all entries in column j will be valid at any iteration h with $\tau^h \geq r_j$). In general, the extended procedure will work as the basic one if at every iteration there are m non-conflicting ready rows with an entry in a tight column so that the entries from each tight column are included into the corresponding decrementing sets. Otherwise, in case there is an entry in a tight column that was not selected at an iteration h, the corresponding entry in row 0 should have been positive, i.e., the corresponding job is not released by the current scheduling time τ^{h-1}. In other words, if an entry from a tight column j was not included in decrementing set D^h then there is no valid entry from that column at iteration h, equivalently, $r_j > \tau^{h-1}$. No feasible schedule may include such a job at time τ^{h-1}. Likewise, if there are only $l < m$ non-conflicting ready rows at iteration h, then there are only l jobs that can be feasibly scheduled at time τ^{h-1}. Our tie breaking rule will include an entry corresponding to each of these l jobs in decrementing set D^h, in total l entries from l rows corresponding to l most loaded machines will be included. Finally, note that among conflicting rows, ties can easily be broken by selecting valid entries from the rows corresponding to most loaded machines.

Example 4. Let us first illustrate the extended scheduling method for the scheduling instance from Theorem 1. Recall an optimal distribution respected by the schedule of Figure 1. Let us first assume that we have a solution to the corresponding partition instance, i.e., we have sets P_1 and P_2. Then, for the sake of simplicity, we can represent all partition jobs in one column marked as P, see Table 2 below. We update the entries in this column according to the made selections. All partition jobs from sets P_1 and P_2 are scheduled consequently without creating any machine idle time in two different (aggregated) iterations 2 and 4. In the first quarter of Table 2 an initial extended processing time matrix T^0 is presented. For this instance, all three rows are ready from the beginning of iteration 1. The valid entries corresponding to the elements in the decrementing set D_1 are circled (note that the corresponding entries in row 0 are 0). Row 1 and column 1 are tight, hence element t_{11} is circled. From rows 2 and 3 elements t_{24} and t_{32} are similarly selected. We have $\delta^1 = 3$ and $\tau^1 = 3$. The part of the schedule of Figure 1 corresponding to the interval $[0, 3)$ is the partial schedule σ^1 of iteration 1. Matrix T^1 is similarly represented in the second quarter of Table 2 (note that the entries in row 0 are updated correspondingly). Now, $\delta^2 = 1$ and hence $\tau^2 = 4$. The part of the schedule of Figure 1 corresponding to the interval $[0, 4)$ is partial schedule σ^2. Similarly, in the following iterations 3 and 4, $\delta^3 = 1, \tau^3 = 5$, and $\delta^4 = 1$ and $\tau^4 = 6$.

Schedule σ^4 is the resultant complete schedule of Figure 1 which extends through the interval $[0, 6)$. Since column 1 is tight, the corresponding parts of job J^1 are included in the decrementing sets in iterations 1, 2, 3 and 4 (until this job is completely scheduled). Note that $C_{\max} = 6$ is attained by both, job J^1 and machine 2 in the optimal distribution of Figure 1 (i.e., the total length of job J^1 and the load of machine 2, see inequalities 3.1 and 5.1). Hence, schedule σ^4 is optimal.

Table 2. Flowchart of the extended procedure with processing time matrices and the corresponding decrementing sets.

	1	2	3	4	P	1	2	3	4	P	1	2	3	4	P	1	2	3	4	P	1	2	3	4	P
0	0	0	4	0	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	④	0	2	0	0	①	0	2	0	0	0	0	②	0	0	0	0	①	0	0	0	0	0	0	0
2	1	0	0	③	2	1	0	0	0	②	①	0	0	0	1	0	0	0	0	③	0	0	0	0	0
3	1	⑤	0	0	0	1	②	0	0	0	1	①	0	0	0	①	0	0	0	0	0	0	0	0	0

Given an optimal distribution of Figure 1, the extended scheduling procedure will create an optimal schedule without the knowledge of a solution to the PARTITION instance. Then the jumps δ^i will be determined by the lengths of the selected partition jobs (without our aggregated presentation, the number of iterations would depend on k, in our case, it would be $k + 2$). Note that the procedure may split a partition job or/and job J^2, which is not allowed for setting $R|r_j, pmtn - nosplit|L_{\max}$. For instance, in this example, job J^2 was not split on machine 3 the corresponding parts of that job being selected in consecutive iterations 1, 2 and 3 (in fact, we solved an instance of problem $R|r_j, pmtn - nosplit|L_{\max}$ given a solution to the PARTITION instance, see Corollary 2). Finally, note that, depending on the made selections of the decrementing sets, the procedure may create different optimal schedules respecting the same distribution. □

Example 1 (continuation 2). Next, we illustrate the extended schedule construction procedure on the problem instance of our basic Example 1. We first construct an optimal schedule of Figure 2d respecting optimal distribution 2 (recall that this schedule is not globally optimal). Table 3 represents the flowchart of the procedure in its 10 iterations (we omit the table with all 0 entries of the last iteration 11). Initially in matrix T^1 all entries in row 0 are positive except that of column (job) 5. In particular, only row 4 is ready. Since 2 is the minimum entry in row 0, $\delta^1 = 2$, hence $\tau^1 = 2$. The partial schedule of iteration 1 corresponds to the segment $[0, 2)$ of the schedule in Figure 7. In the next matrix T^2 there arises one additional ready row 2 with the entry 5 in column 2 (recall that the minimum entry in column 0 of matrix T^1 was precisely in column 2). Now the minimum entry in row 0 is 3, hence $\delta^2 = \min\{5, 1\} = 1$ and $\tau^1 = 2 + 1 = 3$, see again Figure 2d. The entries in the decrementing set D^2 correspond to columns 2 and 5 (both, jobs 2 and 5 are released by time 2). Again, in the processing time matrix T^3 there arises one additional ready row 1 corresponding to job 3. The decrementing set D^3 contains now 3 jobs so that already 3 machines, 1, 2 and 4 become busy. At the next iteration 4 job 6 becomes available on machine 2, but the last unscheduled part of job 2 is scheduled on that machine by our tie breaking rule. The decrementing set of iteration 5 already contains 4 elements, hence all four machines become busy; $\delta^5 = 1$ and $\tau^5 = 2 + 1 + 2 + 2 + 1 = 8$, see Figure 7. At iteration 6 all entries in row 0 in matrix T^6 are already 0, hence all job parts become released. The procedure continues in the same fashion until it constructs a complete feasible schedule of Figure 2d respecting an optimal distribution 2 to linear program LP4($C_{\max}$).

Table 3. Flowchart of the extended procedure with processing time matrices and the corresponding decrementing sets applied to an optimal distribution.

	2	3	4	5	6
0	2	3	8	0	5
1	0	5	13	0	0
2	5	0	0	0	13
3	6	10	0	0	0
4	5	0	0	⑬	0

	2	3	4	5	6
0	0	1	6	0	3
1	0	5	13	0	0
2	⑤	0	0	0	13
3	6	10	0	0	0
4	5	0	0	⑪	0

	2	3	4	5	6
0	0	0	5	0	2
1	0	⑤	13	0	0
2	④	0	0	0	13
3	6	10	0	0	0
4	5	0	0	⑩	0

	2	3	4	5	6
0	0	0	3	0	0
1	0	③	13	0	0
2	②	0	0	0	13
3	6	10	0	0	0
4	5	0	0	⑧	0

	2	3	4	5	6
0	0	0	1	0	0
1	0	①	13	0	0
2	0	0	0	0	⑬
3	⑥	10	0	0	0
4	5	0	0	⑥	0

	2	3	4	5	6
0	0	0	0	0	0
1	0	0	⑬	0	0
2	0	0	0	0	⑫
3	⑤	10	0	0	0
4	5	0	0	⑤	0

	2	3	4	5	6
0	0	0	0	0	0
1	0	0	⑧	0	0
2	0	0	0	0	⑦
3	0	⑩	0	0	0
4	⑤	0	0	0	0

	2	3	4	5	6
0	0	0	0	0	0
1	0	0	③	0	0
2	0	0	0	0	②
3	0	⑤	0	0	0
4	0	0	0	0	0

	2	3	4	5	6
0	0	0	0	0	0
1	0	0	①	0	0
2	0	0	0	0	0
3	0	③	0	0	0
4	0	0	0	0	0

	2	3	4	5	6
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	②	0	0	0
4	0	0	0	0	0

Table 4. Flowchart of the extended procedure with processing time matrices and the corresponding decrementing sets applied to a non-optimal distribution.

	2	3	4	5	6
0	2	3	8	0	5
1	0	5	13	0	0
2	5	0	0	0	13
3	4	10	0	0	0
4	7	0	0	⑬	0

	2	3	4	5	6
0	0	1	6	0	3
1	0	5	13	0	0
2	⑤	0	0	0	13
3	4	10	0	0	0
4	7	0	0	⑪	0

	2	3	4	5	6
0	0	0	5	0	2
1	0	⑤	13	0	0
2	④	0	0	0	13
3	4	10	0	0	0
4	7	0	0	⑩	0

	2	3	4	5	6
0	0	0	1	0	0
1	0	①	13	0	0
2	0	0	0	0	⑬
3	④	10	0	0	0
4	7	0	0	⑥	0

	2	3	4	5	6
0	0	0	0	0	0
1	0	0	⑬	0	0
2	0	0	0	0	⑫
3	③	10	0	0	0
4	7	0	0	⑤	0

	2	3	4	5	6
0	0	0	0	0	0
1	0	0	⑨	0	0
2	0	0	0	0	⑧
3	0	⑩	0	0	0
4	7	0	0	①	0

	2	3	4	5	6
0	0	0	0	0	0
1	0	0	⑧	0	0
2	0	0	0	0	⑦
3	0	⑧	0	0	0
4	⑦	0	0	0	0

	2	3	4	5	6
0	0	0	0	0	0
1	0	0	①	0	0
2	0	0	0	0	0
3	0	①	0	0	0
4	0	0	0	0	0

Now we apply our schedule construction procedure to a non-optimal distribution to obtain a globally optimal schedule respecting that distribution. We illustrate this using a non-optimal distribution 3 from Example 1 (see Figure 2e). We represent the flowchart of the procedure in Table 4. Initially at iteration 1 we have one ready row 4 with a single valid entry 13 (corresponding to job 2) which is included into the decrementing set D^1; $\delta_1 = t^1_{02} = 2$ and $\tau^1 = 2$. At iteration 2 row 2 also gets ready with a single valid entry 5 corresponding to job 2; $D^2 = \{t^2_{22}, t^2_{45}\} = \{5, 11\}$, $\delta_2 = t^2_{03} = 1$ and $\tau^2 = 2 + 1 = 3$. At iteration 3 one additional row 1 with a single valid entry 5 corresponding to job 2 gets ready; $D^3 = \{t^3_{13}, t^3_{22}, t^3_{45}\} = \{5, 4, 10\}$; now $\delta_3 = t^3_{06} = 2$, but since the remaining processing time of job 2 is 13, the same as that of job 6, by our tie breaking rule we include t^3_{22} (and not t^3_{26}) in set D^3 (without creating unnecessary preemption of job 2). We respectively skip one iteration by setting $\delta_3 = 4$ and hence letting $\tau^3 = 2 + 1 + 4 = 7$. At iteration 4 all 4 rows are ready and the corresponding four entries marked in the fourth quarter of Table 4 are included in set D^4. The next four iterations are similarly reflected in Table 4 (we omitted gain the 0-matrix of iteration 9). The resultant complete optimal schedule with makespan $\tau^8 = 2 + 1 + 4 + 1 + 3 + 2 + 7 + 1 = 21$ coincides with that of Figure 2e.

□

7. Concluding remarks

We studied preemptive non-split scheduling problem on unrelated machines, where a job part assigned to any machine is to be processed continuously on that machine without any further preemption. This setting is worth of study from both, theoretical and practical points of view. On the one hand, we showed that the problem is NP-hard, contrary to the traditional well-studied split version where a job part, assigned to a machine can be preempted any number of times on that machine. In practice, such redundant interruptions would imply additional setup costs which are typically undesirable. We showed that, in contrary to the traditional setting, an optimal schedule for our no-split problem not necessarily respects an optimal LP-distribution. Hence, alternative solution methods, ones which are not based on linear programming, are required. On the other hand, we extended the earlier proposed method for the construction of an optimal schedule respecting an optimal LP-distribution with simultaneously released jobs. Our extended method takes into account job release times and constructs a schedule with the minimum makespan respecting an LP-distribution for non-simultaneously released jobs.

Optimal distributions to our linear programs, including LP4($C_{\max}$), do not fully capture all required features for the construction of an optimal schedule to problem $R|r_j; pmtn|C_{\max}$, since an optimal schedule not necessarily respects an optimal LP-distribution to these linear programs. An intelligent linear program would "correctly" estimate the starting time of the first scheduled job on each machine, but this kind of estimation looks unrealistic without actually carrying out the scheduling of assigned job parts. Some optimal distributions may suit better than others, but any optimal schedule may respect a non-optimal distribution. In particular, a schedule with the minimum makespan respecting some non-optimal distribution created by our procedure may be globally optimal. Moreover, no optimal schedule respecting an optimal distribution may exist. The following challenging questions motivate further relevant research in this direction. Can an LP-solver be adopted to generate optimal distributions with some specific favorable properties, which kind of additional restrictions are to be imposed to obtain an optimal LP-distribution with desired favorable properties, which are these favorable properties that a distribution, respected by an optimal schedule, should possesses?

As it is well-known, the problem $R|pmtn|C_{\max}$ with simultaneously released jobs has a favorable property that the makespan of an optimal schedule respecting an optimal LP-distribution to linear program LP2($C_{\max}$) equals to the corresponding $C_{\max}$. As we showed, similar property does not hold for problem $R|r_j; pmtn|C_{\max}$ with non-simultaneously released jobs. In fact, the proposed here procedure for the construction of a schedule with the minimum makespan respecting an LP-distribution may create a schedule with the makespan larger than the corresponding $C_{\max}$, whereas it can be applied to any, in general, non-optimal LP-distribution. In particular, we could obtain a globally optimal schedule by applying our schedule construction procedure to a non-optimal LP-distribution. Such distribution may alternatively be obtained using, instead of linear programming, some other method. A study of such alternative methods can be a subject for further research.

Dedicated to the memory of my friend and colleague Badri Mamporia.

Conflict of interest

The author declares no conflict of interests.

References

1. T. Gonzalez, S. Sahni, Preemptive scheduling of uniform processor systems, *J. Assoc. Comput. Mach.*, **25** (1978), 92–101. https://doi.org/10.1145/322047.322055
2. J. Labetoulle, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, Preemptive scheduling of uniform machines subject to release dates, In: *Progress in combinatorial optimization*, New York: Academic Press, 245–261, 1984. https://doi.org/10.1016/B978-0-12-566780-7.50020-9
3. E. L. Lawler, J. Labetoulle, On preemptive scheduling of unrelated parallel processors by linear programming, *J. Assoc. Comput. Mach.*, **25** (1978), 612–619. https://doi.org/10.1145/322092.322101
4. C. N. Potts, Analysis of a linear programming heuristic for scheduling unrelated parallel machines, *Discrete Appl. Math.*, **10** (1985), 155–164. https://doi.org/10.1016/0166-218X(85)90009-5
5. J. K. Lenstra, D. B. Shmoys, E. Tardos, Approximation algorithms for scheduling unrelated parallel machines, *Math. Program.*, **46** (1990), 259–271. https://doi.org/10.1007/BF01585745
6. E. Shchepin, N. Vakhania, An optimal rounding gives a better approximation for scheduling unrelated machines, *Oper. Res. Lett.*, **33** (2005), 127–133. https://doi.org/10.1016/j.orl.2004.05.004
7. T. Gonzalez, S. Sahni, Open shop scheduling to minimize finish time, *J. Assoc. Comput. Mach.*, **23** (1976), 665–679. https://doi.org/10.1145/321978.321985
8. T. Gonzalez, E. L. Lawler, S. Sahni, Optimal preemptive scheduling of two unrelated processors, *ORSA J. Comput.*, **2** (1990), 209–301. https://doi.org/10.1287/ijoc.2.3.219
9. E. Shchepin, N. Vakhania, On the geometry, preemptions and complexity of multiprocessor and open shop scheduling, *Ann. Oper. Res.*, **159** (2008), 183–213. https://doi.org/10.1007/s10479-007-0266-1
10. M. Foumani, K. Smith-Miles, The impact of various carbon reduction policies on green flowshop scheduling, *Appl. Energ.*, **249** (2019), 300–315. https://doi.org/10.1016/j.apenergy.2019.04.155
11. G. L. Gong, Q. W. Deng, X. R. Gong, D. Huang, A non-dominated ensemble fitness ranking algorithm for multi-objective flexible job-shop scheduling problem considering worker flexibility and green factors, *Knowl. Based Syst.*, **231** (2021), 107430. https://doi.org/10.1016/j.knosys.2021.107430