RAIRO-Oper. Res. 55 (2021) S1747–S1765
https://doi.org/10.1051/ro/2020031

RAIRO Operations Research
www.rairo-ro.org

SOLVING A TIME-INDEXED FORMULATION FOR AN UNRELATED PARALLEL MACHINE SCHEDULING PROBLEM BY PREPROCESSING AND CUTTING PLANES

Lotte Berghman[1,*], Frits C.R. Spieksma[2] and Vincent T'kindt[3]

Abstract. We consider a time-indexed formulation for the unrelated parallel machine scheduling problem. We show that all polyhedral knowledge known from the single machine problem (in particular, valid inequalities) is applicable to this formulation. We present new facet-inducing valid inequalities and a preprocessing technique involving fixing variables based on reduced costs. We combine both techniques in a basic cutting-plane algorithm and test the performance of the resulting algorithm by running it on randomly generated instances.

Mathematics Subject Classification. 90B35.

Received June 14, 2019. Accepted March 18, 2020.

1. Introduction

Time-indexed formulations for single machine scheduling problems are well studied in the literature. Seminal works of Dyer and Wolsey [8] and Sousa and Wolsey [13], and further works by Crama and Spieksma [5], van den Akker *et al.* [17] and Berghman and Spieksma [3] have resulted in a large body of polyhedral results for time-indexed formulations. Generally speaking, the major advantage of a time-indexed formulation is the tight LP-bound, while the greatest disadvantage are the large number of variables, especially when processing times are large. One possible avenue to overcome, at least partially, this difficulty is using column generation, as was done in van den Akker *et al.* [19] and Bigras *et al.* [4]. An arc-time indexed formulation is an extended formulation that yields strictly better bounds than the time-indexed formulation at the cost of an even larger number of variables, one for each pair of jobs and each possible completion time (see, *e.g.*, [12, 14]).

As far as we are aware, all this polyhedral knowledge has not been applied to time-indexed formulations of scheduling problems with multiple machines, in particular unrelated parallel machine scheduling problems. This is confirmed by Unlu and Mason [16] who evaluate integer programming formulations for parallel machine scheduling and recommend to use a time-indexed formulation when job processing times are small. Moreover, they explicitly suggest to develop valid inequalities.

Keywords. Unrelated machine scheduling, time-indexed formulation, valid inequalities, cutting plane algorithm, variable fixing.

[1] TBS Business School, 20 BD Lascrosses BP 7010, 31068 Toulouse Cedex 7, France.

[2] Department of Mathematics and Computer Science, TU Eindhoven, 5600 MB Eindhoven, The Netherlands.

[3] Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée (EA 6300), ERL CNRS 7002 ROOT, 64 Avenue Jean Portalis, 37200 Tours, France.

*Corresponding author: l.berghman@tbs-education.fr

This paper deals with the time-indexed formulation of the unrelated parallel machine scheduling problem, where the processing cost of a job is an arbitrary function of its starting time. Notice that this allows to model many objective functions such as (weighted) sum of completion times or total lateness, tardiness or flow time and to incorporate features such as release times and precedence relations. Our goal is (1) to point out that all polyhedral knowledge existing for single-machine problems can be extended to multi-machine problems, (2) to describe a new class of facet-inducing inequalities for the time-indexed formulation for multiple machines, (3) to implement a preprocessing technique that uses variable fixing based on reduced costs, and (4) to show the computational performance of an algorithm that combines valid inequalities and variable fixing by testing this algorithm on randomly generated instances.

The remainder is organized as follows. The problem statement and a well-known time-indexed single machine scheduling formulation are presented in Section 2. We show in Section 3 that existing valid inequalities can be applied to our formulation. In Section 4, we present a new class of facet-inducing valid inequalities and Section 5 describes the preprocessing technique based on variable fixing. Section 6 presents our final algorithms and the outcome of running them on randomly generated instances, while Section 7 contains the conclusions.

2. Integer programming formulations

Consider the problem of scheduling n jobs on a single machine within a given timespan. The timespan $[0, T]$ is discretized into T time periods of length one. Period t refers to the time slot $[t-1, t]$; $t = 1, \ldots, T$. The processing time of job j equals p_j. The machine can handle at most one job at a time and preemption is not allowed. When job j starts in time period t, a known cost of c_{jt} is incurred. The problem is to find a schedule that minimizes total cost.

This problem can be modeled as follows: for each job j and for each time period $t = 1, \ldots, T$, we define

$$x_{jt} = \begin{cases} 1 & \text{if the processing of job } j \text{ starts in time periodt,} \\ 0 & \text{otherwise.} \end{cases}$$

The well-known time-indexed formulation for the single machine scheduling problem (as presented in [13,17]) is the following:

$$\min \sum_{j=1}^{n} \sum_{t=1}^{T} c_{jt} x_{jt} \tag{2.1}$$

subject to

$$\sum_{t=1}^{T} x_{jt} = 1 \qquad \forall j = 1, \ldots, n, \tag{2.2}$$

$$\sum_{j=1}^{n} \sum_{s=\max\{0; t-p_j+1\}}^{t} x_{js} \leq 1 \qquad \forall t = 1, \ldots, T, \tag{2.3}$$

$$x_{jt} \in \{0, 1\} \quad \forall j = 1, \ldots, n; \ \forall t = 1, \ldots, T. \tag{2.4}$$

The objective function (2.1) minimizes the total cost. Constraints (2.2) state that each job has to be scheduled exactly once and constraints (2.3) express that during each time period t, only one job can be executed; we refer to (2.3) as the *capacity constraints*. This formulation is often called pseudo-polynomial because the number of variables and the number of constraints depend on the length of the time horizon. Thus, indeed if processing times are large, the number of variables grows. However, notice that (1) the problem is already strongly NP-hard if $p_j = 2$ for all j [5] and (2) there exist applications where the cost of starting a job is "truly" arbitrary, see *e.g.*, the assignment of feeders to a component placement machine [6] or the assignment of ships to berths in container terminals [10], leading to an input of $O(nT)$ numbers.

When one wants to generalize this formulation to the identical parallel machine scheduling problem, the right-hand side of constraints (2.3) can be set to m, the number of machines. However, when the machines are not identical, *i.e.*, when a job's processing time depends on the machines, such a trick is no longer possible.

We now consider the problem of scheduling n jobs on m unrelated parallel machines within a given timespan. Again, each machine can handle at most one job at a time and preemption is not allowed. The processing time of a job now depends on the machine: the processing time of job i on machine k is denoted by p_{ik}. The processing cost of a job depends both on the machine and the time period in which the job is started: the processing cost of job i when executed at machine k and started at time period t is denoted by c_{ikt}. Again, we are interested in a feasible schedule minimizing total cost.

Unrelated parallel machine scheduling has received quite some attention in literature, especially the special case where one wants to minimize total weighted completion time. We will not review this literature, we simply mention Lenstra *et al.* [11] and Gairing *et al.* [9] and the references contained in those papers.

We will model this unrelated parallel machine scheduling problem by reducing to a single machine problem in the following way: by copying each job m times, to obtain nm tasks j. We define J as the set containing all tasks. This set can be partitioned in two different ways. First of all, we consider the subsets $J_i \subseteq J$ with $i = 1, \ldots, n$ containing all tasks related to job i. Secondly, we consider the subsets $J^k \subseteq J$ with $k = 1, \ldots, m$ containing all tasks related to machine k. Every subset $J_i \cap J^k$ consists of a single task j. The processing time of task $j = J_i \cap J^k$ equals $p_j = p_{ik}$. We denote by $c_{jt} = c_{ikt}$ the cost of starting task $j = J_i \cap J^k$ in time period t. Notice that specifying the task (index j), implies specifying the job and the machine, and *vice versa*.

For each task j and for each time period $t = 1, \ldots, T - p_j + 1$, we define the decision variables

$$x_{jt} = \begin{cases} 1 & \text{if task } j \text{ starts in time periodt,} \\ 0 & \text{otherwise.} \end{cases}$$

An IP-model for this machine scheduling problem is the following:

$$\min \sum_{j=1}^{nm} \sum_{t=1}^{T-p_j+1} c_{jt} x_{jt} \tag{2.5}$$

subject to

$$\sum_{j \in J_i} \sum_{t=1}^{T-p_j+1} x_{jt} = 1 \qquad \forall i = 1, \ldots, n, \tag{2.6}$$

$$\sum_{j \in J^k} \sum_{s=\max\{0;t-p_j+1\}}^{t} x_{js} \leq 1 \qquad \forall k = 1, \ldots, m; \quad \forall t = \min_{j \in J_k} p_j, \ldots, T, \tag{2.7}$$

$$x_{jt} \in \{0, 1\} \quad \forall j = 1, \ldots, nm; \quad \forall t = 1, \ldots, T - p_j + 1. \tag{2.8}$$

The objective function (2.5) minimizes the total cost. Constraints (2.6) state that out of the tasks related to job i, *i.e.*, J_i, exactly one task has to be scheduled. The capacity constraints are formulated using constraints (2.7): for each time period t, only one task out of the tasks related to machine k, *i.e.*, J^k, can be executed. We obtain the LP-relaxation of this formulation by replacing constraints (2.8) by the following one: $\forall j = 1, \ldots, nm; \forall t = 1, \ldots, T - p_j + 1 : 0 \leq x_{jt} \leq 1$. In the case of a single machine, *i.e.*, when $m = 1$, formulation (2.5) to (2.8) becomes (2.1) to (2.4).

Notice that there is a polytope for each m, n, T and p (where p represents a vector of processing times). With some abuse of notation, we denote by P_m the convex hull of feasible solutions of (2.6) to (2.8).

3. Known valid inequalities

In this section, we review the known valid inequalities for P_1. Notice that an inequality for P_1 can be extended to an inequality for P_m ($m > 1$) by setting all coefficients that correspond to variables that involve tasks not related to some specific machine k ($1 \leq k \leq m$) to 0. Then, it is not difficult to observe that in this way, any inequality valid for P_1 can be extended to an inequality valid for P_m. We record this observation formally.

Fact 3.1. Any inequality valid for P_r is valid for P_m, for each $r \leq m$.

Fact 3.1 motivates us to formulate the known inequalities in terms of P_m. To do so, we need the following notation. For each $j = 1, \ldots, nm$, we define $T(j)$ as the set of tasks that are related to the same machine as task j. Note that $T(j)$ does not include task j. Moreover, we define $p_j^* = \max_{l \in T(j)} p_l$; thus p_j^* is the largest processing time of the tasks in $T(j)$.

Sousa and Wolsey [13] give the following inequalities. For each time period $t = 1, \ldots, T$, for each task $j = 1, \ldots, nm$ and for each $\Delta \in \{2, \ldots, p_j^*\}$:

$$\sum_{s=t-p_j+1}^{t+\Delta-1} x_{js} + \sum_{l \in T(j)} \sum_{s=\max\{0, t-p_l+\Delta\}}^{t} x_{ls} \leq 1. \tag{3.1}$$

In inequality (3.1), task j is sometimes called the "special" task. These inequalities are known to be facet-defining for P_1 [13], and in fact they constitute all facet-defining inequalities for P_1 with integral coefficients and right-hand side 1 (see [18]).

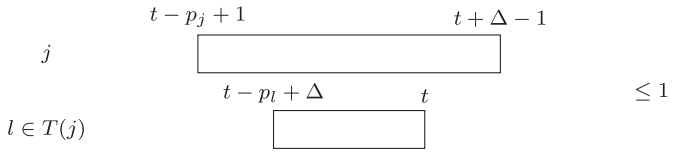
To give a pictorial description of this inequality, we will use a similar notation as van den Akker *et al.* [18]. The index-set of variables with nonzero coefficients in an inequality is denoted by V. The set of nonzero coefficients in an inequality associated with task j defines a set of time periods $V_j = \{t | (j,t) \in V\}$. Thus the union over all j of all V_j equals V. We define an interval $[a,b]$ as the set of periods $\{a, a+1, \ldots, b\}$. If $a > b$, then $[a,b] = \emptyset$. We shall represent inequalities by diagrams. A diagram contains a line for each task. The blocks on the line associated with task j indicate the time periods t for which x_{jt} occurs in the inequality.

Inequalities (3.1) of Sousa and Wolsey [13] use the following time periods:

$$\begin{array}{ll} \text{for task } j\text{:} & V_j = [t - p_j + 1, t + \Delta - 1]\,, \\ \text{for each task } l \in T(j)\text{:} & V_l = [t - p_l + \Delta, t]\,, \end{array}$$

where $\Delta \in \{2, \ldots, p_j^*\}$.

These inequalities can be represented by the following diagram.



Using this diagram, it is relatively easy to see that inequalities (3.1) are valid. Indeed, notice that if some task $l \in T(j)$ starts at some time in V_l, no other task from $T(j)$ can start in V_l (since both tasks would be active at time t). Also task j cannot start in V_j, since starting task j directly after the completion of task l is impossible: task l is active until $t + \Delta - 1$; starting task j before the beginning of task l is equally impossible, since even starting task j at $t - p_j + 1$ means that task j is active at time t. This implies validity of (3.1).

A lot more is known concerning the facial structure of P_1. Sousa and Wolsey [13] already give other classes of facet-defining inequalities, Berghman and Spieksma [3] generalize the inequalities (3.1) for the case with time-dependant processing times p_{jt}, Crama and Spieksma [5] find classes of facet-defining inequalities that apply to the case of equal processing times, and van den Akker *et al.* [18] present three classes of facet-defining inequalities that collectively constitute all facet-defining inequalities with integral coefficients that have right-hand side 2. We will not give an explicit description of these inequalities, however, let us emphasize here that

TABLE 1. The coefficients c_{jt} for the example instance.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	2	2	2	1	2	2	2	2	2	2	2	2	2
2	1	1	1	1	1	1	1	0	1	1	1	1	1
3	1	1	1	1	1	1	0	1	1	1	1	1	1
4	2	2	2	2	1	2	2	2	2	2	2	2	2
5	0	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1

any facet-defining inequalities for P_1 other than an inequality from Sousa and Wolsey [13] has right-hand side 2 or more. Moreover note that all these inequalities deal with a single machine. In the next section, we exhibit a class of valid inequalities that specifically focus on the presence of multiple machines. Indeed, this is the first description of a class of valid inequalities that contains variables corresponding to different machines.

4. A NEW CLASS OF VALID INEQUALITIES

In this section, we introduce a new class of valid inequalities that contains variables corresponding to different machines.

4.1. Example

We first specify an instance. Let $n = 3$, $m = 2$, $T = 14$, $J^1 = \{1, 2, 3\}$, $J^2 = \{4, 5, 6\}$, $J_1 = \{1, 4\}$, $J_2 = \{2, 5\}$, $J_3 = \{3, 6\}$, $p_1 = 4$, $p_2 = 3$, $p_3 = 5$, $p_4 = 1$, $p_5 = 5$ and $p_6 = 2$, Further, the c_{jt} coefficients are given in Table 1 where a row corresponds to a task, and a column corresponds to a time period.

When solving the LP-relaxation (2.5) to (2.8) of this instance, we find the fractional solution $x_{1,4} = x_{2,8} = x_{3,7} = x_{4,5} = x_{5,1} = x_{6,2} = \frac{1}{2}$. We claim that this solution is not cut off by any known facet-defining inequality. Indeed, observe that the sum of the variables corresponding to jobs on machine 1, *i.e.*, the variables corresponding to jobs $1, 2, 3$, sum up to $\frac{3}{2}$. Hence, this partial solution cannot be eliminated by any known facet-defining inequality other than an inequality from (3.1), since all available facet-defining inequalities other than (3.1) have right-hand side 2 or more. In addition, we leave to the reader to verify that inequalities (3.1) also do not cut away this particular solution. A similar argument holds for the jobs corresponding to machine 2.

4.2. A new class of valid inequalities

For each pair of jobs $\{i_1, i_2\} \in \{1, \ldots, n\}$, and for each pair of machines $\{k_1, k_2\} \in \{1, \ldots, m\}$, let $j = J_{i_1} \cap J^{k_1}$, $q = J_{i_2} \cap J^{k_1}$, $a = J_{i_1} \cap J^{k_2}$ and $b = J_{i_2} \cap J^{k_2}$. Define $p^*_{jq} = \max_{l \in T(j) \cap T(q)} p_l$. For each quadruple of such four tasks, for all time periods $t_1, t_2 = 1, \ldots, T$, for all $\Delta_1 \in \{2, \ldots, p^*_{jq}\}$ and for all $\Delta_2 \in \{1, \ldots, p^*_a\}$, we have the following inequalities:

$$\sum_{s=\max\{t_1-p_j+1;0\}}^{t_1+\Delta_1-1} x_{js} + \sum_{s=\max\{t_1-p_q+1;0\}}^{t_1+\Delta_1-1} x_{qs} + \sum_{l \in T(j)\setminus\{q\}} \sum_{s=\max\{t_1-p_l+\Delta_1;0\}}^{t_1} x_{ls} + \sum_{s=\max\{t_2-p_a+1;0\}}^{t_2+\Delta_2-1} x_{as} + \sum_{s=\max\{t_2-p_b+\Delta_2;0\}}^{t_2} x_{bs} \leq 2. \tag{4.1}$$

These inequalities can be represented by the diagram presented in Figure 1.

Remark that if we have $p^*_{jq} = 1$, we do not have any possible value for Δ_1 and we do not have a new valid inequality. We would obtain the sum of a constraint of type (2.7) for k_1 and t_1 and part of an inequality of type (3.1) for k_2, t_2 and Δ_2.

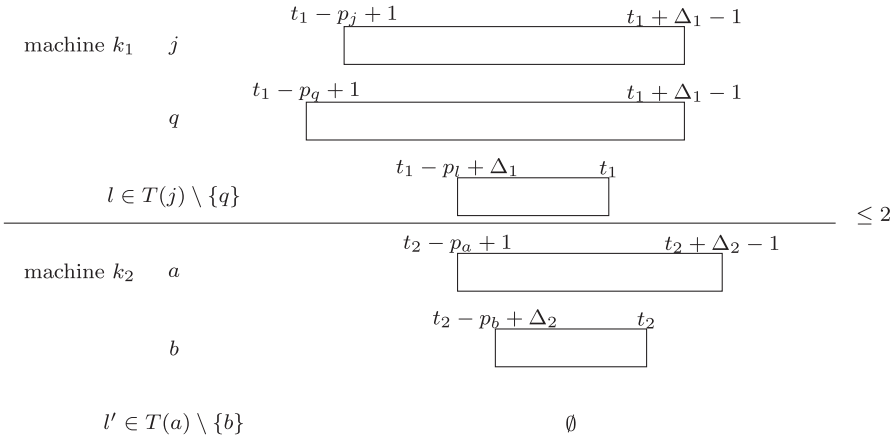


FIGURE 1. The diagram representing the valid inequalities (4.1).

Theorem 4.1. *Inequalities (4.1) are valid inequalities for P_m, for each $m \geq 2$.*

Proof. Observe that the two jobs i_1, i_2, and the two machines k_1, k_2, as well as t_1, t_2, Δ_1, Δ_2 are given. We use integer rounding as follows. Consider two inequalities from type (2.6), one for job i_1, and one for job i_2, each with weight $1 - \frac{1}{\Delta_1}\cdot$ Next, consider inequalities of type (2.7) for machine k_1, and for $t = t_1, \ldots, t_1 + \Delta_1 - 1$, each with weight $\frac{1}{\Delta_1}\cdot$ Finally, we consider an inequality of type (3.1) for machine k_2, period t_2, job i_1 and $\Delta_2 \in \{1, \ldots, p_a^*\}$ with weight $\frac{1}{\Delta_1}\cdot$ (Remark that this inequality becomes another one of type (2.7) if we choose $\Delta_2 = 1$.) If we apply integer rounding on both sides of the resulting inequality, we obtain the inequality (4.1). □

Notice that the Chvatal rank of these inequalities does not exceed 2; further, there are $\mathcal{O}\left(n^2 m^2 T^2 p_{\max}^2\right)$ inequalities in the new class. The inequalities cannot be strengthened, as witnessed by our next result.

Theorem 4.2. *Inequalities (4.1) are facet-defining inequalities for P_m, for each $m \geq 2$.*

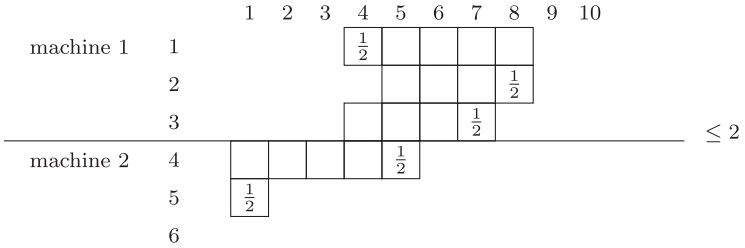
We give a proof of this result in the Appendix A.

4.3. Example continued

With $j = 1$, $q = 2$, $a = 4$, $b = 5$, $t_1 = 7$, $\Delta_1 = 2$, $t_2 = 1$ and $\Delta_2 = 5$, inequality (4.1) boils down to

$$x_{1,4} + x_{1,5} + x_{1,6} + x_{1,7} + x_{1,8} + x_{2,5} + x_{2,6} + x_{2,7} + x_{2,8} + x_{3,4} + x_{3,5} + x_{3,6} + x_{3,7} + x_{4,1} + x_{4,2} + x_{4,3} + x_{4,4} + x_{4,5} + x_{5,1} \leq 2.$$

It is displayed by the squared blocks in the figure below, and cuts off the fractional solution.



5. Preprocessing the IP formulation

Variable fixing is a preprocessing technique that is able to reduce the number of 0–1 variables in formulation (2.5) to (2.8), and it is expected that the reduced IP formulation will be solved faster than the initial formulation. In addition, variable fixing allows to find stronger LP-bounds. Variable fixing is a technique based on simple links between an IP formulation and its linear relaxation. More recent applications of this technique can be found in Baptiste *et al.* [1] and T'kindt *et al.* [15].

We write x^{LP} for the optimal values of the x-variables of the LP-relaxation of (2.5) to (2.8), and we write x^{IP} for the optimal values of the x-variables of the IP-formulation (2.5) to (2.8). Let z^*_{LP} (resp z^*_{IP}) be the value of the corresponding optima and let B^* be the associated basis.

For non-basic variables $x^{\mathrm{IP}}_{jt} \notin B^*$, it is well known that, applied to our model, we have:

$$z^*_{\mathrm{IP}} = z^*_{\mathrm{LP}} + \sum_{x^{\mathrm{IP}}_{jt} \notin B^*} r_{jt} x^{\mathrm{IP}}_{jt}, \tag{5.1}$$

with r_{jt} the reduced cost associated to variable x^{IP}_{jt}. Let UB be an upper bound to z^*_{IP}. Then, we can write:

$$\sum_{x^{\mathrm{IP}}_{jt} \notin B^*} r_{jt} x^{\mathrm{IP}}_{jt} \leq \mathrm{UB} - z^*_{\mathrm{LP}}. \tag{5.2}$$

From inequality (5.2) we can deduce the following fixing rule: $\forall x^{\mathrm{IP}}_{jt} \notin B^*$, if $r_{jt} > \mathrm{UB} - z^*_{\mathrm{LP}}$ then in any optimal solution of the IP formulation, $x^{\mathrm{IP}}_{jt} = 0$. By reasoning on the slack variables s_{jt} associated to the constraints $x_{jt} \leq 1$, we can deduce when $x^{\mathrm{IP}}_{jt} = 1$: if a non-basic slack variable s_{jt} has to be fixed to 0, then the associated variable x^{IP}_{jt} is fixed to 1. Remark that this technique is implemented at some node of the search tree in mathematical programming solvers like CPLEX, using the upper bound in the search tree, calculated by the solver as value for UB.

For basic variables $x^{\mathrm{IP}}_{jt} \in B^*$, we use penalties l_{jt} and u_{jt} computed by means of Driebeek's penalties (also called Beale and Small penalties, see [2] and [7] for more details). The meaning of these penalties is the following: l_{jt} (resp. u_{jt}) is a unitary lower estimate on the increase of z^*_{LP} if x^{IP}_{jt} is set to 0 (resp. 1). We have:

(1) $\forall x^{\mathrm{IP}}_{jt} \in B^*$, if $(l_{jt} x^{\mathrm{LP}}_{jt}) > (\mathrm{UB} - z^*_{\mathrm{LP}})$ then $x^{\mathrm{IP}}_{jt} = 1$,
(2) $\forall x^{\mathrm{IP}}_{jt} \in B^*$, if $(u_{jt}(1 - x^{\mathrm{LP}}_{jt})) > (\mathrm{UB} - z^*_{\mathrm{LP}})$ then $x^{\mathrm{IP}}_{jt} = 0$.

Remark that, to the best of our knowledge, this technique is not implemented in solvers as CPLEX.

The efficiency of these two variable fixing techniques is strongly influenced by the gap between UB and z^*_{LP}, and the value of the reduced costs. To strengthen these techniques we can use valid inequalities like the ones presented in Section 4. The choice of the included valid inequalities and of the upper bound UB are reported in Section 6.

The preprocessing algorithm works as described in Algorithm 1. Ideally, the preprocessing is done at each node of a branch-and-price tree, as the branching decision limits the set of feasible solutions and for that reason (1) one can find additional valid inequalities and (2) one can fix additional basic and non-basic variables. Unfortunately, commercial solvers like CPLEX do not allow to interact at each particular node.

Remark that we can reduce the m-machine problem to a single machine problem by concatenating the timespan of the m machines to obtain a large timespan spanning mT periods. The processing times now become dependent upon the particular period: for each of the first T periods, the processing time of job j equals p_{j1}, then p_{j2} for the next T periods, and so on. All results of this paper are also valid for the single machine case with arbitrary period-dependent processing times as the presented m-machine problem is a special case of this single machine problem.

Algorithm 1. Preprocessing algorithm;
inputs: a set of *valid inequalities* and an upper bound UB on z^*_{IP}.

Add the *valid inequalities* to the LP relaxation.
Solve the LP relaxation: let z^0_{LP} be the optimal solution value.
Fix basic and non-basic variables with UB.
Solve the LP relaxation: let z^1_{LP} be the optimal solution value.
while $(z^0_{\text{LP}} < z^1_{\text{LP}})$ **do**
  Fix basic and non-basic variables with UB.
  $z^0_{\text{LP}} = z^1_{\text{LP}}$.
  Solve the LP relaxation: let z^1_{LP} be the optimal solution value.
end while
return z^1_{LP} and the associated variable values x^1_{LP}

6. Computational evaluations

A series of computational evaluations have conducted in order to evaluate the impacts of the valid inequalities and the preprocessing algorithm on: (1) the linear relaxation LP of the IP formulation; (2) the solution of the IP formulation. We first provide details about the generation of our instances (Sect. 6.1), before discussing the obtained results on the LP (Sect. 6.3) and on the IP (Sect. 6.4). Notice that, as far as the solution of an IP or an LP fomulation is involved, we use the CPLEX solver.

6.1. Generation of the instances

The IP formulation of instances that are generated according to Sousa and Wolsey [13] and van den Akker *et al.* [18] are almost always easy to solve by CPLEX, especially as the number of machines increase. For this reason, we introduce another generation scheme which leads to harder instances for the parallel machine problem. The idea of such a scheme is to increase the number of resource conflicts when trying to minimize the objective function.

The number of jobs n is taken in the set $\{100, 150, 200\}$ and the number of machines m in the set $\{1, 2, 3, 5, 10\}$. The time horizon is defined as $|T| = 1.25 \times \frac{p_{\max}+1}{2} \times \frac{n}{m}$, with $p_{\max} = 20$, the maximal processing time value. The processing times on machines are uniformly distributed in $[1, p_{\max}]$. Remark that as a consequence, the number of variables is $13.125 \times n^2$ and the number of constraints is $14.125 \times n$. The difficulty of the instances comes from the generation of the processing costs c_{jt}: we have set up a complex generation scheme for the c_{jt}'s in order to create conflicts between jobs. The scheduling horizon $[0; T]$ is split into $\sqrt{n}$ equal-size intervals $[T_i; T_{i+1}]$ and $\sqrt{n}$ jobs take their minimum c_{jt} values in each interval. So, all these time intervals have the same number of conflicting jobs. The size of each interval, denoted by SizeInt, is equal to $\lceil \frac{T}{\sqrt{n}} \rceil$ except the last one which can be slightly smaller due to the rounding. Then, c_{jt} are drawn at random in the interval $[0; 10 * T - G(\mu_{jt}, \sigma)]$ with $G(\mu, \sigma)$ referring to a normal distribution of mean μ_{jt} and variance σ. We experimentally set $\mu_{jt} = \frac{9*T}{|t\%\text{SizeInt} - j\%\sqrt{n}|+1}$ and $\sigma = 1.3 * \text{SizeInt}$ with % referring to the modulo operator. The underlying idea is that $\sqrt{n}$ jobs will have similar costs c_{jt} in each time interval and they achieve their minimum value in the same time interval. So, minimizing the problem objective function implies scheduling $\sqrt{n}$ conflicting jobs on the m machines in each time interval. An illustration of the distribution of the $G(\mu_{jt}, \sigma)$'s distributions are given in Figure 2. For each combination of n and m, 20 instances were created, yielding 300 instances in total[4].

6.1.1. Testing environment

All algorithms are encoded in C using the Microsoft Visual Studio programming environment, and executed on a PC computer with an Intel Core i5 CPU 4 Core 2.6 GHz processor and 8 GB RAM, equipped with Windows 7. CPLEX version 12.2 is used to solve the IP and LP models, and is configured to use only 1 thread. When solving

[4]The instances can be found at https://hal.archives-ouvertes.fr/view/index/docid/1685398

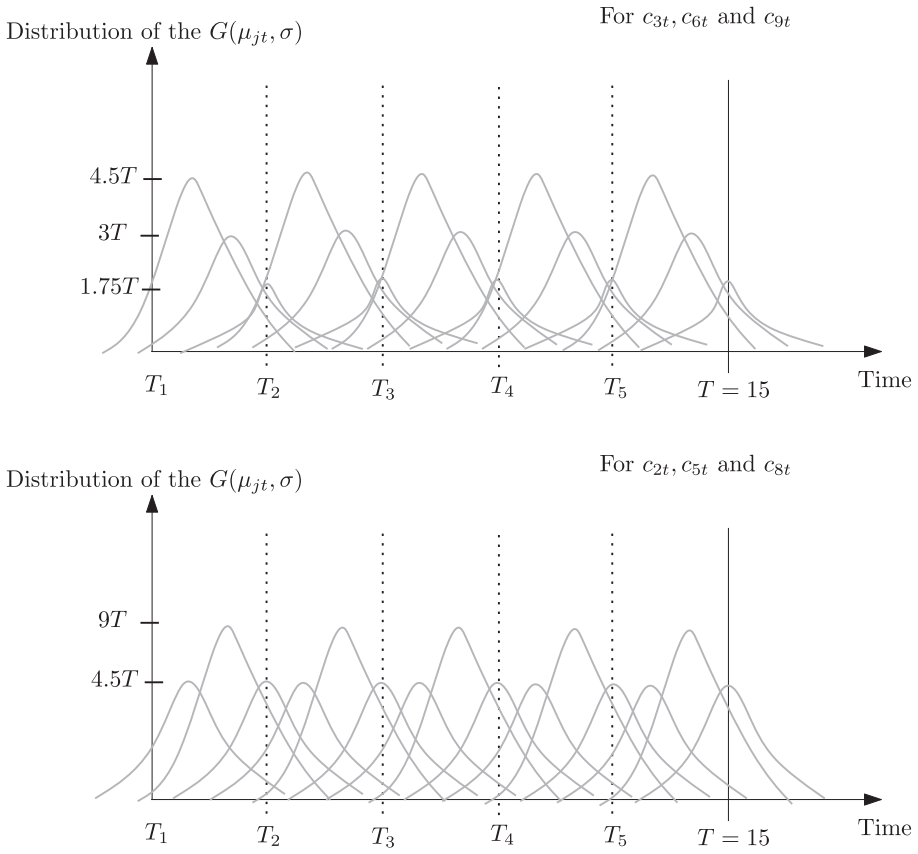


FIGURE 2. Example of the distribution of the processing costs with $n = 9$ and $T = 15$.

the IP model, a time limit of 3600 s is imposed: whenever that limit is reached before the end of the solution, then the solver is assumed to have failed to solve the corresponding instance.

6.2. Separation

Notice that the LP-solution $\mathbf{x}^{\mathrm{LP}}$ will be sparse; this fact is used in the separation. We describe a vector $\mathbf{x}$ by a list of those x_{jt} variables that have a positive value and call this list the *support* of $\mathbf{x}$. Thus, given some $\mathbf{x}$, we view the support of $\mathbf{x}$ as a set of all pairs of indices (j, t) such that $x_{jt} > 0$. We describe in the next fact some necessary conditions with respect to the support of a solution $\mathbf{x}$ that violates an inequality of type (4.1). This can be used to accelarate the separation.

Proposition 6.1. *If* $\mathbf{x}$ *violates an inequality of type (4.1) and satisfies inequalities (2.7) and (3.1), then there exists a violated inequality of type (4.1) defined by parameters* $\{i_1, i_2\} \in \{1, \dots, n\}$, $\{k_1, k_2\} \in \{1, \dots, m\}$ *(with* $j = J_{i_1} \cap J^{k_1}$, $q = J_{i_2} \cap J^{k_1}$, $a = J_{i_1} \cap J^{k_2}$ *and* $b = J_{i_2} \cap J^{k_2}$*),* $t_1, t_2 = 1, \dots, T$, $\Delta_1 \in \{2, \dots, p^*_{jq}\}$ *and* $\Delta_2 \in \{1, \dots, p^*_a\}$ *such that the support of* $\mathbf{x}$ *contains (i) and (ii) or (iii) and (iv)) and (v) and (vi) or (vii) and (viii), with*

(i) $x^{\mathrm{LP}}_{j,t_1-p_j+1}$ *or* $x^{\mathrm{LP}}_{q,t_1-p_q+1}$
(ii) $x^{\mathrm{LP}}_{j,t_1+\Delta_1-1}$ *or* $x^{\mathrm{LP}}_{q,t_1+\Delta_1-1}$
(iii) $x^{\mathrm{LP}}_{l,t_1-p_l+\Delta_1}$ *for some* $l \in T(j) \setminus \{q\}$

(iv) x^{LP}_{l,t_1} *for some* $l \in T(j) \setminus \{q\}$
(v) $x^{\mathrm{LP}}_{a,t_2-p_a+1}$
(vi) $x^{\mathrm{LP}}_{a,t_2+\Delta_2-1}$
(vii) $x^{\mathrm{LP}}_{b,t_2-p_b+\Delta_2}$
(viii) x^{LP}_{b,t_2}.

Proof. We argue by contradiction. Each time, we suppose that $\mathbf{x}$ violates a specific inequality (4.1), determined by parameters $i_1, i_2, k_1, k_2, t_1, t_2, \Delta_1, \Delta_2$ and that the support of $\mathbf{x}$ does not contain the considered variables. First, deal with (i). There are two cases: either $\Delta_1 = 2$ or $\Delta_1 > 2$. In the first case, since $x^{\mathrm{LP}}_{j,t_1-p_j+1} + x^{\mathrm{LP}}_{q,t_1-p_q+1} = 0$, it follows that each variable related to machine k_1 in the violated inequality is active at period t_1+1, and hence this violated inequality is implied by an inequality (2.7) with $t = t_1+1$. In the second case, we claim that the inequality of type (4.1) determined by parameters $i_1, i_2, k_1, k_2, t_1+1, t_2, \Delta_1-1, \Delta_2$ (called the "new" inequality) also corresponds to a violated inequality of type (4.1). This follows from the observation that the only variables that appear in the former inequality and not in the "new" one are $x^{\mathrm{LP}}_{j,t_1-p_j+1}$ and $x^{\mathrm{LP}}_{q,t_1-p_q+1}$ both equal to 0.

Consider now (ii). There are two cases: either $\Delta_1 = 2$ or $\Delta_1 > 2$. In the first case, since $x^{\mathrm{LP}}_{j,t_1+\Delta_1-1} + x^{\mathrm{LP}}_{q,t_1+\Delta_1-1} = 0$, it follows that each variable related to machine k_1 in the violated inequality is active at period t_1, and hence this violated inequality is implied by an inequality (2.7) with $t = t_1$. In the second case, we claim that the inequality of type (4.1) determined by parameters $i_1, i_2, k_1, k_2, t_1, t_2, \Delta_1 - 1, \Delta_2$ also corresponds to a violated inequality of type (4.1).

Considering (iii), we claim that the inequality of type (4.1) determined by parameters $i_1, i_2, k_1, k_2, t_1, t_2, \Delta_1 + 1, \Delta_2$ (called the "new" inequality) also corresponds to a violated inequality of type (4.1). And considering (iv), we claim that the inequality of type (4.1) determined by parameters $i_1, i_2, k_1, k_2, t_1 - 1, t_2, \Delta_1 + 1, \Delta_2$ also corresponds to a violated inequality of type (4.1).

Consider now (v). There are two cases: either $\Delta_2 = 1$ or $\Delta_2 > 1$. In the first case, we claim that the inequality of type (4.1) determined by parameters $i_2, i_1, k_1, k_2, t_1, t_2, \Delta_1, \Delta_2 = 2$ also corresponds to a violated inequality of type (4.1). In the second case, we claim that the inequality of type (4.1) determined by parameters $i_1, i_2, k_1, k_2, t_1, t_2 + 1, \Delta_1, \Delta_2 - 1$ also corresponds to a violated inequality of type (4.1).

Consider now (vi). There are two cases: either $\Delta_2 = 1$ or $\Delta_2 > 1$. In the first case, we claim that the inequality of type (4.1) determined by parameters $i_2, i_1, k_1, k_2, t_1, t_2-1, \Delta_1, \Delta_2 = 2$ also corresponds to a violated inequality of type (4.1). In the second case, we claim the same for parameters $i_1, i_2, k_1, k_2, t_1, t_2, \Delta_1, \Delta_2 - 1$.

Considering (vii), we claim that the inequality of type (4.1) determined by parameters $i_1, i_2, k_1, k_2, t_1, t_2, \Delta_1, \Delta_2 + 1$ also corresponds to a violated inequality of type (4.1). And finally, considering (viii), we claim that the inequality of type (4.1) determined by parameters $i_1, i_2, k_1, k_2, t_1, t_2 - 1, \Delta_1, \Delta_2 + 1$ also corresponds to a violated inequality of type (4.1). □

6.3. Improving the LP relaxation

To see the effect of the valid inequalities, we have implemented a basic cutting-plane algorithm. Its working is illustrated in Figure 3. As the separation of the inequalities of van den Akker *et al.* [18] takes too much computation time for the generation instances, we do not consider these inequalities in the computational experiments.

Table 2 provides statistics on this algorithm. Remark that preprocessing is not yet included in the algorithm here. We present the average total computation time to execute the cutting-plane algorithm as *time*, the average value of the LP solution as z^*_{LP} and the average value of the IP solution as z^*_{IP}. Recall that each row corresponds to 20 instances. Moreover, we provide statistics on the frequency with which optimal solutions are found. More precisely, we report the number of instances for which the optimal solution of the LP-relaxation is integral, n_{LP}, the number of instances for which the solution is integral after the addition of cuts (3.1), $n_{(3.1)}$, and after addition of cuts (4.1), $n_{(4.1)}$. For the instances whose LP-relaxation is fractional, the percentage of the gap $(z_{\mathrm{IP}} - z_{\mathrm{LP}})$

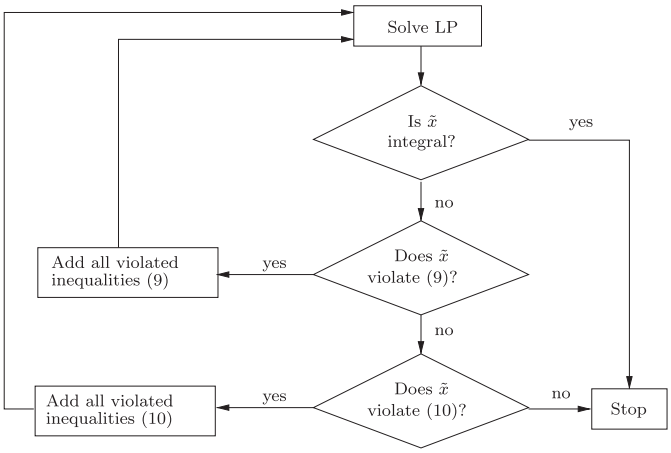


FIGURE 3. The basic cutting-plane algorithm.

that is closed after adding valid inequalities is displayed. The percentage is computed as $100 \times \frac{z(\mathbf{x}^{\mathrm{LP}}) - \mathbf{z}^*_{\mathrm{LP}}}{z^*_{\mathrm{IP}} - z^*_{\mathrm{LP}}}$ where $z(\mathbf{x}^{\mathrm{LP}})$ is the value found after adding the corresponding inequalities. When, in an extreme case, the LP solution and the IP solution have the same value, although the LP solution is fractional, we say that the gap is closed with 0% if the solution stays fractional after adding cuts and the gap is closed with 100% when the solution becomes integral. We also report the number of valid inequalities that were added ($\mathrm{VI}_{(3.1)}$ and $\mathrm{VI}_{(4.1)}$ respectively). z^{IP} is found by giving CPLEX one hour of computation time to find an IP solution.

We see that a small portion of the instances has an integral LP-solution, to be precise 5%. This percentage seems to grow mildly with the size of an instance (more concretely with the number of machines m). Adding inequalities (3.1) helps in producing integral solutions: another 4 out of the 300 instances become integral and inequalities (4.1) yield another 4 additional instances. We conclude that both classes have a contribution in closing (part of) the gap. Inequalities (3.1) are quite powerful, bridging on average 5.32% of the gap. Inequalities (4.1) are also quite effective, bridging an additional 5.28% of the gap although this is mostly true for instances with large m.

Next, we implement the same cutting-plane algorithm, but we now use variable fixing in each iteration. To obtain an upper bound on the optimal solution, we run CPLEX to find a feasible solution with a GAP smaller than 3%. If no such feasible solution is found within the time limit of one minute, we rerun CPLEX until one is found.

Table 3 provides statistics on the implementation. We see that adding preprocessing helps further in producing integral solutions and in closing the gap. Adding inequalities (3.1) makes that another 11 out of the 300 instances become integral and bridge on average 7.76% of the gap. Inequalities (4.1) yield another 7 integral instances and bridge on average 5.15% of the gap.

6.4. Exact solution of the problem

In this section we focus on the exact solution of the IP formulation. A first approach consists in directly solving the IP formulation by CPLEX with a time limit of one hour (referred to as $InitIP$). The second approach (referred to as Alg2) consists in applying the preprocessing technique inside a cutting-plane algorithm, as described in Algorithm 2, before solving by CPLEX the modified instance. The latter may have fixed variables x_{jt} and added valid inequalities. We implemented two variants to asses the impact of the tightness of the upper bound on the performance of the algorithm. In Alg2(0), we use the best objective value found by CPLEX within one hour of computation time as an upper bound for the proprocessing. The performance of this variant will

TABLE 2. Impact of the valid inequalities (3.1) and (4.1).

$(m \times n)$	Time	LP		LP + (3.1)			LP + (3.1), (4.1)			IP
		z^*_{LP}	n_{LP}	$GAP_{(3.1)}$	$n_{(3.1)}$	$VI_{(3.1)}$	$GAP_{(4.1)}$	$n_{(4.1)}$	$VI_{(4.1)}$	z^*_{IP}
1 × 100	2.35	774 598.42	0	5.52%	0	6.00	Not applicable			774 884.65
1 × 150	8.67	1 797 688.70	0	4.07%	0	6.65	Not applicable			1 798 358.50
1 × 200	19.61	3 287 610.99	0	2.55%	0	6.25	Not applicable			3 288 583.75
2 × 100	107.15	232 060.62	0	2.11%	0	1.20	2.60%	0	317.90	232 110.20
2 × 150	329.87	575 452.75	0	2.51%	0	3.60	2.67%	0	346.50	575 646.10
2 × 200	613.17	1 039 188.77	0	1.22%	0	2.80	1.44%	0	392.05	1 039 433.30
3 × 100	62.30	77 873.45	0	3.28%	0	0.20	3.58%	0	189.25	77 892.05
3 × 150	174.64	197 899.01	1	1.14%	1	0.37	1.85%	1	196.63	197 972.50
3 × 200	396.12	388 240.13	0	3.74%	0	1.80	3.95%	0	234.35	388 400.85
5 × 100	14.54	6907.19	1	0.00%	1	0.00	18.82%	2	300.16	6909.00
5 × 150	72.29	21 615.12	0	0.00%	0	0.00	2.94%	0	225.05	21 620.50
5 × 200	200.40	47 849.28	1	0.00%	1	0.00	1.18%	1	729.89	47 861.15
10 × 100	10.52	2804.70	7	25.13%	9	0.31	48.03%	11	29.00	2805.25
10 × 150	74.69	10 044.04	3	23.91%	5	0.53	31.45%	5	540.41	10 045.20
10 × 200	83.21	22 839.14	2	4.55%	2	0.11	12.45%	3	230.06	22 839.90

TABLE 3. Impact of preprocessing with the objective value for the heuristic procedure as an upper bound.

$(m \times n)$	Time	LP		LP + pre + (3.1)			LP + pre + (3.1), (4.1)			IP
		z_{LP}	n_{LP}	$GAP_{(3.1)}$	$n_{(3.1)}$	$VI_{(3.1)}$	$GAP_{(4.1)}$	$n_{(4.1)}$	$VI_{(4.1)}$	z_{IP}
1 × 100	68.98	774 598.42	0	5.65%	0	6.20	Not applicable			774 884.65
1 × 150	180.75	1 797 688.70	0	4.07%	0	6.65	Not applicable			1 798 358.50
1 × 200	290.88	3 287 610.99	0	2.55%	0	6.25	Not applicable			3 288 583.75
2 × 100	105.09	232 060.62	0	2.11%	0	1.20	2.82%	0	250.90	232 110.20
2 × 150	308.35	575 452.75	0	2.51%	0	3.55	2.67%	0	346.50	575 646.10
2 × 200	487.69	1 039 188.77	0	1.22%	0	2.75	1.44%	0	392.05	1 039 433.30
3 × 100	53.65	77 873.45	0	3.34%	0	0.20	3.64%	0	162.05	77 892.05
3 × 150	178.61	197 899.01	1	1.15%	1	0.37	1.85%	1	0.00	197 972.50
3 × 200	415.38	388 240.13	0	3.74%	0	1.80	3.95%	0	234.20	388 400.85
5 × 100	14.83	6907.19	1	7.02%	2	0.00	23.64%	5	216.32	6909.00
5 × 150	77.35	21 615.12	0	5.00%	1	0.00	7.94%	1	225.05	21 620.50
5 × 200	347.54	47 849.28	1	0.01%	1	0.00	1.18%	1	255.79	47 861.15
10 × 100	10.31	2804.70	7	38.46%	12	0.31	58.79%	14	15.62	2805.25
10 × 150	191.09	10 044.04	3	23.91%	5	0.53	33.66%	6	146.06	10 045.20
10 × 200	286.27	22 839.14	2	15.66%	4	0.11	24.37%	5	86.11	22 839.90

give us an idea about the effectiveness of the algorithm in case we are able to find a very tight upper bound within a small computation time, and can be seen as some best case scenario. In Alg2(1), this upper bound is obtained by the same heuristic procedure as described in Section 6.3. The processing time of the heuristic is here included in the total computation time of the algorithm. For experimentation, we only consider instances with one, two or three machines since the generated instances start to be easily solvable (in a few seconds) by CPLEX as the number of machines increases.

In Table 4, we present the number of instances (out of 20) that were solved to optimally in column # for all algorithms. We also mention the average objectif values. The average computation time and the average number

TABLE 4. Exact solution of the problem.

		InitIP				Alg2(0)					Alg2(1)			
$(m \times n)$	#	Objectif	Time	#Nodes	#	Objectif	Time	#Nodes	Fixed	#	Objectif	Time	#Nodes	Fixed
1 × 100	19	774 884.65	498.31	3076.05	19	774 883.30	436.80	3239.85	68.99	19	774 882.25	536.95	3056.85	53.13
1 × 150	14	1 798 358.50	2096.69	4610.45	12	1 798 342.65	2092.70	5518.70	56.02	11	1 798 341.50	2573.16	6450.25	24.59
1 × 200	0	3 288 583.75	3600.00	1667.75	0	3 288 592.70	3728.64	1918.75	42.07	0	3 288 574.25	3600.00	1726.45	17.88
2 × 100	20	232 110.20	35.48	350.10	20	232 110.20	168.52	378.40	96.11	20	232 110.20	194.86	291.20	95.86
2 × 150	19	575 646.10	1047.84	6242.25	19	575 647.40	1223.35	7363.95	83.38	18	575 646.10	1478.15	7040.20	27.04
2 × 200	12	1 039 433.30	2639.28	6632.80	14	1 039 435.20	2708.34	7819.15	83.65	10	1 039 453.10	2724.27	5906.60	29.88
3 × 100	20	77 892.05	14.36	62.00	20	77 892.05	97.36	48.15	99.03	20	77 892.05	54.62	48.15	99.03
3 × 150	20	197 972.50	96.81	1183.70	20	197 972.50	349.92	1271.15	97.47	20	197 972.50	199.58	702.80	94.27
3 × 200	20	388 400.85	571.00	2274.25	20	388 400.85	982.68	2568.65	94.48	20	388 401.15	883.00	2814.75	65.57

Algorithm 2. Exact solution with an initial preprocessing; input: an upper bound UB on z^*_{MIP}.

Run Algorithm 1 with UB and no valid inequalities: $\mathbf{x}^*_{\text{LP}}$ denotes the returned variable values.
if ($\mathbf{x}^*_{\text{LP}}$ is not integral) **then**
  Iterate=*true*.
end if
while (Iterate=*true*) **do**
  Starting with $\mathbf{x}^*_{\text{LP}}$, let VI be the set of violated inequalities (3.1).
  if ($|\text{VI}| \neq \emptyset$) **then**
    Run Algorithm 1 with UB and VI: $\mathbf{x}^*_{\text{LP}}$ denotes the returned variable values.
  else
    Iterate=*false*.
  end if
end while
Convert the LP with added valid inequalities and fixed variables into an IP model.
Solve the resulting IP: z^*_{IP} is the computed optimal solution value.
return z^*_{IP}.

of nodes that CPLEX explores, are displayed in columns *time* and *#nodes* for all algorithms. For Alg2(0) and Alg2(1) we also provide in column *fixed* the average percentage of variables that are fixed by the preprocessing.

Looking at the results of *InitIP*, we notice that the instances seem to be easier for CPLEX as the number of machines increase. Remark that the cutting-plane algorithm with preprocessing may penalize the solution by CPLEX: effort is spent in generating valid inequalities and fixing variables in Alg2(0) and Alg2(1) while *InitIP* is efficient. To illustrate that, look at the instances with 3 machines and 100 jobs. On average, these instances are solved by CPLEX in only 14.36 s, so we cannot expect a large gain in solution time. Indeed, generating valid inequalities and fixing variables lead to an increased overall CPU time even if the number of explored nodes is reduced and more than 99% of the variables are fixed by preprocessing.

In general, we can state that *InitIP* performs rather well and that the cutting-plane algorithm with preprocessing does not really outperform it. However, to solve difficult instances, it turns out that the the cutting-plane algorithm with preprocessing yields better results. The efficiency of the cutting-plane algorithm with preprocessing decreases as the upper bound becomes weaker. For that reason, Table 4 highlights an interesting possible future research: improving the quality of the heuristic algorithm to obtain better final solutions for the problem. Notice that we have not been able to do preprocessing at each node of the branch-and-cut algorithm of CPLEX due to a lack of interactions with the mathematical solver. This would also drastically improve the results of Alg2(1).

7. Conclusion

We modeled the unrelated parallel machine scheduling problem where the processing cost of each job is an arbitrary function of its starting time as a single machine scheduling problem using a time-indexed formulation. We have shown that valid inequalities from literature for single-machine problems can be applied to multi-machine problems. A new set of facet-inducing inequalities has been presented, and a cutting-plane algorithm has been proposed. We have also implemented a preprocessing technique within that algorithm to try to reduce the size of the instances to solve. Computational experiments have been conducted and show that the valid inequalities lead to strengthen the linear relaxation of the time-indexed formulation. These experiments also show that the proposed cutting-plane algorithm with preprocessing may help to solve the instances which are difficult to solve for the mathematical solver more efficiently.

As future research directions, it would be interesting to implement a faster separation algorithm for the inequalities of van den Akker *et al.* [18] to improve the exact solution of the problem by the cutting-plane algorithm with preprocessing. It would also be interesting to provide a very fast and good heuristic algorithm to get a good preprocessing. This might considerably improve the computational results for both the LP relaxation and the IP formulation. At last, the possibly most promising research line from an experimental point of view is certainly to integrate the preprocessing at each node of the branch-and-cut algorithm used to solve the time-indexed formulation.

Appendix A.

Theorem A.1. *Inequalities (4.1) are facet-defining inequalities for P_m, for each $m \geq 2$.*

Proof. Let us first establish the dimension of the corresponding polytope P_m.

Lemma A.2. $\dim(P_m) = n(mT + m - 1) - \sum_{j=1}^{nm} p_j$.

Proof. The total number of variables in (2.5)–(2.8) equals: $\sum_{j=1}^{nm}(T - p_j + 1) = n(mT + m) - \sum_{j=1}^{nm} p_j$. Since we have n linearly independent inequalities (2.6) it easily follows that $\dim(P_m) \leq n(mT + m) - \sum_{j=1}^{nm} p_j$.

We will now show that, in fact, equality holds. For ease of notation in the second part of this proof, we change the indices for p and x in the following way: $\forall j = J_i \cap J^k$, $p_j = p_{ik}$ and $x_{jt} = x_{i,k,t}$.

Consider some equality that is valid for all feasible solutions x:

$$\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1} \pi_{i,k,t} x_{i,k,t} = \pi_0. \tag{A.1}$$

We now identify a particular feasible solution, denoted by solution S, as follows. Let some job i $(1 \leq i \leq n)$ start on machine k $(1 \leq k \leq m)$ at time s $(1 \leq s \leq T - p_{ik} + 1)$; all other jobs start on machine ℓ, $\ell \neq k$, in a way that no job on machine ℓ overlaps the moments $\{s, s+1, \ldots, s + p_{i\ell} - 1\}$ (notice that this is always possible, if T is large enough, say $T \geq 2 \times \max_k \sum_i p_{ik}$).

Consider now a feasible solution that is identical to S except that job i starts on machine k at time t, $t \neq s$. Since equality (A.1) holds for all feasible solutions, it follows that

$$\pi_{i,k,s} = \pi_{i,k,t} \quad \forall i = 1, \ldots, n; \quad \forall k = 1, \ldots, m; \quad \forall s, t \in \{1, \ldots, T - p_{ik} + 1\}. \tag{A.2}$$

Consider now another feasible solution that is identical to S except that job i starts also on machine ℓ at time s. Again, since equality (A.1) holds for all feasible solutions, it follows that

$$\pi_{i,k,s} = \pi_{i,\ell,s} \quad \forall i = 1, \ldots, n; \quad \forall k, \ell \in \{1, \ldots, m\}; \quad \forall s = 1, \ldots, T - p_{ik} + 1. \tag{A.3}$$

Using (A.2) and (A.3), we conclude that there exist multipliers π_i such that

$$\pi_i = \pi_{i,k,t} \quad \forall i = 1, \ldots, n; \quad \forall k = 1, \ldots, m; \quad \forall t = 1, \ldots, T - p_{ik} + 1.$$

Thus, we can rewrite equality (A.1) as follows:

$$\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1}\pi_{i,k,t}x_{i,k,t}=\sum_{i=1}^{n}\pi_i\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1}x_{i,k,t}=\pi_0,$$

thereby showing that equality (A.1) is a linear combination of equalities (2.6). This proves the lemma. □

We now proceed to prove that inequalities (4.1) are facet-defining. In order to facilitate the description of the proof, we define the following intervals of time-units. Given jobs i_1, i_2, machines k_1, k_2, time-units t_1, t_2, and parameters Δ_1, Δ_2, we define:

$$\begin{aligned}
A &= [t_1 - p_{i_1,k_1} + 1, \ldots, t_1 + \Delta_1 - 1],\\
B &= [t_1 - p_{i_2,k_1} + 1, \ldots, t_1 + \Delta_1 - 1],\\
C_i &= [t_1 - p_{i,k_1} + \Delta_1, \ldots, t_1]\,(i \neq i_1, i \neq i_2),\\
D &= [t_2 - p_{i_1,k_2} + 1, \ldots, t_2 + \Delta_2 - 1],\\
E &= [t_2 - p_{i_2,k_2} + \Delta_2, \ldots, t_2].
\end{aligned}$$

This allows us to rewrite inequality (4.1) as follows:

$$\sum_{s\in A}x_{i_1,k_1,s}+\sum_{s\in B}x_{i_2,k_1,s}+\sum_{i:i\neq i_1,i\neq i_2}\sum_{s\in C_i}x_{i,k_1,s}+\sum_{s\in D}x_{i_1,k_2,s}+\sum_{s\in E}x_{i_2,k_2,s}\leq 2. \tag{A.4}$$

Let $F = \{x \in P_m\colon x \text{ satisfies (A.4) with equality}\}$, and consider some equality that is valid for all $x \in F$:

$$\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1}\pi_{i,k,t}x_{i,k,t}=\pi_0.$$

We will show that there exist multipliers ρ_i $(1 \leq i \leq n)$, and α such that:

$$\begin{aligned}
\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1}\pi_{i,k,t}x_{i,k,t} = \sum_{i=1}^{n}\rho_i\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1}x_{i,k,t} + \alpha\Bigg(&\sum_{s\in A}x_{i_1,k_1,s}+\sum_{s\in B}x_{i_2,k_1,s}\\
&+\sum_{i:i\neq i_1,i\neq i_2}\sum_{s\in C_i}x_{i,k_1,s}+\sum_{s\in D}x_{i_1,k_2,s}+\sum_{s\in E}x_{i_2,k_2,s}\Bigg)=\pi_0.
\end{aligned}$$

This shows that any equality valid for all feasible solutions, is a linear combination of the equalities (2.6) and the equality corresponding to (A.4). In our proof we will denote by F the set of feasible solutions. We will schedule jobs for which the starting time and the affected machine is not important for the proof "in some feasible way on some machine k". Our assumption that T is large enough guarantees that there is indeed always some way to place those jobs on that machine.

Consider now a solution called S_1, where job i_1 starts on machine k_1 at time $t_1 - p_{i_1,k_1} + 1$, where job i_2 starts on machine k_1 at time $t_1 + 1$, where some job i, $(i \neq i_1, i \neq i_2)$ starts on some machine k $(k \neq k_1)$ at time s, with $1 \leq s \leq T - p_{ik} + 1$, and where all other jobs start, in some feasible way, on machine k_1. We modify solution S_1 by starting job i on machine k at time t, $t \neq s$. Clearly, both solution S_1, and the modified solution are in F. It follows that

$$\pi_{i,k,s}=\pi_{i,k,t}\quad \forall i\in\{1,\ldots,n\}\setminus\{i_1,i_2\};\quad \forall k=1,\ldots,m;\quad k\neq k_1;\quad \forall s,t\in\{1,\ldots,T-p_{ik}+1\}. \tag{A.5}$$

Also, we can modify S_1 by shifting job i to machine ℓ, $\ell \neq k, \ell \neq k_1$ at time s, to arrive at:

$$\pi_{i,k,s}=\pi_{i,\ell,s}\quad \forall i\in\{1,\ldots,n\}\setminus\{i_1,i_2\};\quad \forall k,\ell\in\{1,\ldots,m\}\setminus\{k_1\};\quad \forall s=1,\ldots,T-p_{ik}+1. \tag{A.6}$$

Together, equalities (A.5) and (A.6) imply:

$$\pi_{i,k,t} = \rho_i \quad \forall i \in \{1,\dots,n\} \setminus \{i_1, i_2\}; \quad \forall k = 1,\dots,m; \quad k \neq k_1; \quad \forall t = 1,\dots,T - p_{ik} + 1. \tag{A.7}$$

We will now proceed to argue that (A.7) is in fact also valid for machine k_1 when $t \notin C_i$, $1 \leq i \leq n$, *i.e.*, we will show that:

$$\pi_{i,k_1,t} = \rho_i \quad \forall i \in \{1,\dots,n\} \setminus \{i_1, i_2\}; \quad \forall t \notin C_i. \tag{A.8}$$

Consider now solution S_2, where job i_1 starts on machine k_1 at time $t_1 - p_{i_1,k_1} + 1$, with $p_{i,k_1} \leq t_1 \leq T - p_{i,k_1}$ for all i, where job i_2 starts on machine k_2 at time $t_2 - p_{i_2,k_2} + \Delta_2$, where some job i starts on machine k_1 at time s, $s \geq t_1 + 1$, and where all other jobs are placed in some feasible way on machine k_2 leaving free the time units $[s,\dots,s + p_{i,k_2} - 1]$. We modify solution S_2 by starting job i on machine k_1 at time $t = 1$. Clearly, both solution S_2, and the modified solution are in F. It follows that:

$$\pi_{i,k_1,s} = \pi_{i,k_1,1} \quad \forall i \in \{1,\dots,n\} \setminus \{i_1, i_2\}; \quad \forall s = t_1 + 1,\dots,T - p_{i,k_1} + 1. \tag{A.9}$$

We can also modify S_2 by shifting job i to some machine k ($k \neq k_1$) at time s, implying (using (A.7)):

$$\pi_{i,k_1,s} = \pi_{i,k,s} = \rho_i \quad \forall i \in \{1,\dots,n\} \setminus \{i_1, i_2\}; \quad \forall k = 1,\dots,m; \quad k \neq k_1; \quad \forall s = 1,\dots,T - p_{i,k_1} + 1. \tag{A.10}$$

Consider now solution S_3, where job i_1 starts on machine k_1 at time $t_1 + \Delta_1 - 1$, where job i_2 starts on machine k_2 at time $t_2 - p_{i_2,k_2} + \Delta_2$, where some job i starts on machine k_1 at time s, $s \leq t_1 - p_{i,k_1} + \Delta_1 - 1$, and where all other jobs are placed in some feasible way on machine k_2. We modify solution S_3 by starting job i on machine k_1 at time $t = 1$. Clearly, both solution S_3, and the modified solution are in F. It follows that:

$$\pi_{i,k_1,s} = \pi_{i,k_1,1} \quad \forall i \in \{1,\dots,n\} \setminus \{i_1, i_2\}; \quad \forall s = 1,\dots,t_1 - p_{i,k_1} + \Delta_1 - 1. \tag{A.11}$$

Together, the conditions (A.9) to (A.11) imply (A.8).

Consider now solution S_4 where job i_1 starts on machine k_2 at time $t_2 - p_{i_1,k_2} + 1$, where job i, $i \neq i_1, i \neq i_2$ starts on machine k_1 at time $t_1 - p_{i,k_1} + \Delta_1$, where job i_2 starts on some machine k, $k \neq k_1, k \neq k_2$ at some time s, and where all other jobs are placed in a feasible way on machine k_1. We modify solution S_4 by starting job i_2 on machine k at time t ($t \neq s$). Clearly, both solution S_4, and the modified solution are in F. We get:

$$\pi_{i_2,k,s} = \pi_{i_2,k,t} \quad \forall k \in \{1,\dots,m\} \setminus \{k_1, k_2\}; \quad \forall s,t \in \{1,\dots,T - p_{i_2,k} + 1\}. \tag{A.12}$$

We can also modify S_4 by shifting job i_2 to some other machine ℓ ($\ell \neq k_1, \ell \neq k_2$) at time s, implying:

$$\pi_{i_2,k,s} = \pi_{i_2,\ell,s} \quad \forall k,\ell \in \{1,\dots,m\} \setminus \{k_1, k_2\}; \quad \forall s = 1,\dots,T - p_{i_2,k} + 1. \tag{A.13}$$

Together, the conditions (A.12) and (A.13) imply:

$$\pi_{i_2,k,t} = \rho_{i_2} \quad \forall k \in \{1,\dots,m\} \setminus \{k_1, k_2\}; \quad \forall t = 1,\dots,T - p_{i_2,k} + 1. \tag{A.14}$$

A similar construction can be used to infer:

$$\pi_{i_1,k,t} = \rho_{i_1} \quad \forall k \in \{1,\dots,m\} \setminus \{k_1, k_2\}; \quad \forall t = 1,\dots,T - p_{i_1,k} + 1. \tag{A.15}$$

Consider solution S_5 where job i_1 starts on machine k_1 at time s, $s \notin A$, where job i_2 starts on machine k_2 at time t_2, where job i starts on k_1 at time $t_1 - p_{i,k_1} + \Delta_1$, and where all other jobs are placed in a feasible way on machine k_2. We modify solution S_5 by starting job i_1 on machine k_1 at time t ($t \notin A$). Clearly, both solution S_5, and the modified solution are in F. Using constructions like these, we get:

$$\pi_{i_1,k_1,s} = \pi_{i_1,k_1,t} \quad \forall s,t \notin A. \tag{A.16}$$

We can also modify S_5 by shifting job i_1 to some other machine k ($k \neq k_1$) at time s, implying:

$$\pi_{i_1,k_1,s} = \pi_{i_1,k,s} \quad \forall k = 1,\ldots,m; \quad k \neq k_1; \quad \forall s = 1,\ldots,T - p_{i_1,k_1} + 1. \tag{A.17}$$

Together, the conditions (A.16), and (A.17) imply:

$$\pi_{i_1,k_1,t} = \rho_{i_1} \quad \forall t \notin A. \tag{A.18}$$

Similar constructions can be used to infer:

$$\pi_{i_1,k_2,t} = \rho_{i_1} \quad \forall t \notin D, \tag{A.19}$$

$$\pi_{i_2,k_1,t} = \rho_{i_2} \quad \forall t \notin B, \tag{A.20}$$

and

$$\pi_{i_2,k_2,t} = \rho_{i_2} \quad \forall t \notin E. \tag{A.21}$$

Consider solution S_6 where job i_1 starts on machine k_1 at time s, $s \in A$, where job i_2 starts on machine k_2 at time $t_2 - p_{i_2,k_2} + \Delta_2$, and where all other jobs are placed in a feasible way on machine k_2. We modify solution S_6 by starting job i_1 on machine k_1 at time t ($t \neq s, t \in A$). Clearly, both solution S_6, and the modified solution are in F. We get:

$$\pi_{i_1,k_1,s} = \pi_{i_1,k_1,t} \quad \forall s,t \in A. \tag{A.22}$$

In a similar fashion, we can derive:

$$\pi_{i_1,k_2,s} = \pi_{i_1,k_2,t} \quad \forall s,t \in D. \tag{A.23}$$

Moreover, consider solution S_7 where job i_1 starts on machine k_1 at time $t_1 - p_{i_1,k_1} + 1$, where job i_2 starts on machine k_1 at time $t_1 + \Delta_1 - 1$, and where all other jobs are placed in a feasible way on machine k_2, leaving free the time units $[t_2 - p_{i_1,k_2} + 1,\ldots,t_2]$. We modify solution S_7 by starting job i_1 on machine k_2 at time $t_2 - p_{i_1,k_2} + 1$. Clearly, both solution S_7, and the modified solution are in F. We get:

$$\pi_{i_1,k_1,t_1-p_{i_1,k_1}+1} = \pi_{i_1,k_2,t_2-p_{i_1,k_2}+1}. \tag{A.24}$$

Using conditions (A.22) to (A.24), we find:

$$\pi_{i_1,k_1,t} = \pi_{i_1,k_2,s} \equiv \pi^{in}_{i_1} \quad \forall t \in A; \quad \forall s \in D. \tag{A.25}$$

In a similar fashion, we can derive:

$$\pi_{i_2,k_1,t} = \pi_{i_2,k_2,s} \equiv \pi^{in}_{i_2} \quad \forall t \in B; \quad \forall s \in E, \tag{A.26}$$

and:

$$\pi_{i,k_1,t} = \pi^{in}_{i} \quad \forall i \in \{1,\ldots,n\} \setminus \{i_1,i_2\}; \quad \forall t \in C_i. \tag{A.27}$$

Consider solution S_8 where job i_1 starts on machine k_2 at time $t_2 - p_{i_1,k_2} + 1$, where job i_3 ($i_3 \neq i_2$) starts on machine k_1 at time t_1 (notice that $t_1 \in C_{i_3}$), where job i_4 ($i_4 \neq i_2, i_4 \neq i_3$) starts on machine k_1 at time s with $s \geq t_1 + p_{i_3,k_1} + p_{i_4,k_1}$, and where all other jobs are somewhere on machine k_2. We modify solution S_8 by interchanging jobs i_3 and i_4, ie, by starting job i_3 on machine k_1 at time s, and job i_4 on machine k_1 at time t_1. Clearly, both solution S_8, and the modified solution are in F. Then, we find that:

$$\pi_{i_3,k_1,t_1} + \pi_{i_4,k_1,s} = \pi_{i_3,k_1,s} + \pi_{i_4,k_1,t_1}.$$

Using (A.8), this is equivalent to:

$$\pi_{i_3,k_1,t_1} - \rho_{i_3} = \pi_{i_4,k_1,t_1} - \rho_{i_4} \equiv \alpha,$$

or, by extending the construction for any time $t \in C_i$

$$\pi_{i,k_1,t} - \rho_i = \alpha \quad \forall i \in \{1,\ldots,n\} \setminus \{i_1, i_2\}; \quad \forall t \in C_i. \tag{A.28}$$

Consider a solution S_9 where job i_1 starts on machine k_1 at time t_1, where job i_2 starts on machine k_2 at time t_2 (notice that $t_2 \in E$), where some job i ($i \neq i_1, i \neq i_2$) starts on machine k_1 at time s with $s \geq t_1 + p_{i_1,k_1} + p_{i,k_1}$, and where all other jobs are somewhere on machine k_2. We modify solution S_9 by interchanging jobs i_1 and i, *i.e.*, by starting job i on machine k_1 at time t_1, and job i_1 on machine k_1 at time s. Clearly, both solution S_9, and the modified solution are in F. Then, we find that:

$$\pi_{i_1,k_1,t_1} + \pi_{i,k_1,s} = \pi_{i_1,k_1,s} + \pi_{i,k_1,t_1}.$$

Using (A.8) and (A.18), this is equivalent to:

$$\pi_{i_1,k_1,t_1} - \rho_{i_1} = \pi_{i,k_1,t_1} - \rho_i = \alpha,$$

or, in fact:

$$\pi_{i_1,k_1,t} - \rho_{i_1} = \pi_{i_1,k_2,s} - \rho_{i_1} = \alpha \quad \forall t \in A; \quad \forall s \in D. \tag{A.29}$$

A similar construction allows us to conclude:

$$\pi_{i_2,k_1,t} - \rho_{i_2} = \pi_{i_2,k_2,s} - \rho_{i_2} = \alpha \quad \forall t \in B; \quad \forall s \in E. \tag{A.30}$$

We are now finally able to derive the result:

$$\begin{aligned}
\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1} \pi_{i,k,t}x_{i,k,t} = {} & \sum_{i:i\neq i_1,i\neq i_2}\sum_{k\neq k_1}\left(\sum_{t=1}^{T-p_{ik}+1}\pi_{i,k,t}x_{i,k,t}\right) + \sum_{t\notin C_i}\pi_{i,k_1,t}x_{i,k_1,t} + \sum_{t\in C_i}\pi_{i,k_1,t}x_{i,k_1,t} \\
& + \sum_{k\neq k_1,k\neq k_2}\left(\sum_{t=1}^{T-p_{i_1,k}+1}\pi_{i_1,k,t}x_{i_1,k,t} + \sum_{t=1}^{T-p_{i_2,k}+1}\pi_{i_2,k,t}x_{i_2,k,t}\right) \times \sum_{t\in A}\pi_{i_1,k_1,t}x_{i_1,k_1,t} \\
& + \sum_{t\in D}\pi_{i_1,k_2,t}x_{i_1,k_2,t} + \sum_{t\notin A}\pi_{i_1,k_1,t}x_{i_1,k_1,t} + \sum_{t\notin D}\pi_{i_1,k_2,t}x_{i_1,k_2,t} \\
& + \sum_{t\in B}\pi_{i_2,k_1,t}x_{i_2,k_1,t} + \sum_{t\in E}\pi_{i_2,k_2,t}x_{i_2,k_2,t} \\
& + \sum_{t\notin B}\pi_{i_2,k_1,t}x_{i_2,k_1,t} + \sum_{t\notin E}\pi_{i_2,k_2,t}x_{i_2,k_2,t}.
\end{aligned}$$

By plugging in the values we found for the $\pi_{i,k,t}$ coefficients derived in the conditions respectively (A.7), (A.8), (A.28), (A.15), (A.14), (A.29), (A.18), (A.19), (A.30), (A.20) and (A.21), we find:

$$\begin{aligned}
\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1} \pi_{i,k,t}x_{i,k,t} = {} & \sum_{i=1}^{n}\rho_i\sum_{k=1}^{m}\sum_{t=1}^{T-p_{ik}+1} x_{i,k,t} + \alpha\Bigg(\sum_{s\in A}x_{i_1,k_1,s} + \sum_{s\in B}x_{i_2,k_1,s} \\
& + \sum_{i:i\neq i_1,i\neq i_2}\sum_{s\in C_i}x_{i,k_1,s} + \sum_{s\in D}x_{i_1,k_2,s} + \sum_{s\in E}x_{i_2,k_2,s}\Bigg) = \pi_0,
\end{aligned}$$

thereby proving our result. □

Acknowledgements. This work is supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office, and by FWO grant G.0729.13. Frits Spieksma is supported by NETWORKS (Grant No. 024.002.003).

References

[1] P. Baptiste, F. Della Croce, A. Grosso and V. T'Kindt, Sequencing a single machine with due dates and deadlines: an ILP-based approach to solve very large instances. *J. Scheduling* **13** (2010) 39–47.

[2] E. Beale and R. Small, Mixed integer programming by a branch-and-bound technique. *Proc. Third IFIP Cong.* **2** (1965) 450–451.

[3] L. Berghman and F.C.R. Spieksma, Valid inequalities for a time-indexed formulation. *Oper. Res. Lett.* **43** (2015) 268–272.

[4] L. Bigras, M. Gamache and G. Savard, Time-indexed formulations and the total weighted tardiness problem. *INFORMS J. Comput.* **20** (2008) 133–142.

[5] Y. Crama and F.C.R. Spieksma, Scheduling jobs of equal length: complexity, facets and computational results. *Math. Prog.* **72** (1996) 207–227.

[6] Y. Crama, A.W.J. Kolen, A.G. Oerlemans and F.C.R. Spieksma, Throughput rate optimization in the automated assembly of printed circuit boards. *Ann. Oper. Res.* **26** (1990) 455–480.

[7] N.J. Driebeek, An algorithm for the solution of mixed integer programming problems. *Manage. Sci.* **12** (1966) 576–587.

[8] M.E. Dyer and L.A. Wolsey, Formulating the single machine sequencing problem with release dates as a mixed integer problem. *Disc. Appl. Math.* **26** (1990) 255–270.

[9] M. Gairing, B. Monien and A. Woclaw, A faster combinatorial approximation algorithm for scheduling unrelated parallel machines. *Theor. Comput. Sci.* **380** (2007) 87–99.

[10] P. Hansen, C. Oğuz and N. Mladenović, Variable neighborhood search for minimum cost berth allocation. *Eur. J. Oper. Res.* **191** (2008) 636–649.

[11] J.K. Lenstra, D.B. Shmoys and E. Tardos, Approximation algorithms for scheduling unrelated parallel machines. *Math. Prog.* **46** (1990) 259–271.

[12] F. Sourd, New exact algorithms for one-machine earliness-tardiness scheduling. *INFORMS J. Comput.* **21** (2009) 167–175.

[13] J.P. Sousa and L.A. Wolsey, A time indexed formulation of non-preemptive single machine scheduling problems. *Math. Prog.* **54** (1992) 353–367.

[14] S. Tanaka, S. Fujikuma and M. Araki, An exact algorithm for single-machine scheduling without machine idle time. *J. Scheduling* **12** (2009) 575–593.

[15] V. T'kindt, F. Della Croce and J.L. Bouquard, Enumeration of Pareto optima for a flowshop scheduling problem with two criteria. *INFORMS J. Comput.* **19** (2007) 64–72.

[16] Y. Unlu and S.J. Mason, Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Comput. Ind. Eng.* **58** (2010) 785–800.

[17] J.M. van den Akker, J.A. Hoogeveen and S.L. van de Velde, Parallel machine scheduling by column generation. *Oper. Res.* **47** (1999) 862–872.

[18] J.M. van den Akker, C.P.M. Van Hoesel and M.W.P. Savelsbergh, A polyhedral approach to single-machine scheduling problems. *Math. Prog.* **85** (1999) 541–572.

[19] J.M. van den Akker, C.A.J. Hurkens and M.W.P. Savelsbergh, Time-indexed formulations for machine scheduling problems: column generation. *INFORMS J. Comput.* **12** (2000) 111–124.