Minimizing the maximum lateness on a single machine with raw material constraints by branch-and-cut

Péter Györgyi[a], Tamás Kis[a,*]

[a]*Institute for Computer Science and Control, H1111 Budapest, Kende str. 13–17, Hungary*

Abstract

Machine scheduling with raw material constraints has a great practical potential, as it is solved by ad-hoc methods in practice in several manufacturing and logistic environments. In this paper we propose an exact method for solving this problem with the maximum lateness objective based on mathematical programming, our main contribution being a set of new cutting planes that can be used to accelerate a MIP solver. We report on computational results on a wide set of instances.

Keywords: Scheduling, single machine, non-renewable resources, branch-and-cut, integer programming

1. Introduction

Counting with raw materials (or non-renewable resources more generally) in the course of planning and scheduling of manufacturing processes is inevitable in order to obtain feasible production plans and schedules (see e.g., Stadtler & Kilger (2008)). The following case occurs frequently in practice and constitutes the main motivation of this paper. We have to schedule the production of some parts on a production line over the next week, and we have an initial stock and expect some additional shipments from the suppliers over the week. Our goal is to minimize the maximum of the late deliveries, or in other words, the lateness. Since the parts may require common raw materials for their production, it is not obvious how to allocate the supplies to the parts to produce. The arising optimization problem is precisely the topic of this paper.

More formally, we focus on scheduling a single machine subject to raw material constraints. That is, in addition to the machine, there are some raw materials with an initial stock and some additional replenishments over time with a-priori known dates and quantities. Jobs may require various quantities from these resources, and a job can be started only if the required amount is on

*Corresponding author

Email addresses: gyorgyi.peter@sztaki.mta.hu (Péter Györgyi), kis.tamas@sztaki.mta.hu (Tamás Kis)

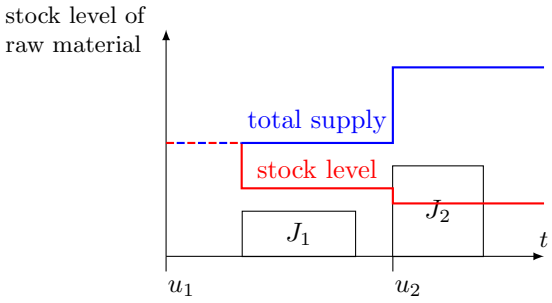

Figure 1: Illustration of the problem. The height of a job indicates the amount of the required raw material.

stock. Upon starting a job, the stock level of all the resources are decreased by the quantities needed by the job. Each job has a due-date and the objective is to minimize the maximum lateness. As an illustration, consider Figure 1 in which a schedule of two jobs is shown on a single machine, and notice that job J_2 must wait until the replenishment of the raw-material, because the first scheduled job decreases the stock level below its requirement.

The above model has been first studied by Carlier (1984), and by Slowinski (1984). In particular, Carlier has shown that minimizing the maximum job completion time (makespan) is NP-hard in the strong sense in general. This implies that our problem is NP-hard in the strong sense as well. Over the years, a number of papers appeared dealing with some variants and proposing either complexity results (Toker et al. (1991), Xie (1997), Gafarov et al. (2011)), or approximation algorithms (Grigoriev et al. (2005), Györgyi & Kis (2015a), Györgyi & Kis (2015b), Györgyi & Kis (2017)). However, there are only sporadic computational results on this problem. Grigoriev et al. (2005) have provided some test results for one of their approximation algorithms. Belkaid et al. (2012) propose lower bounds and heuristics for minimizing the makespan in a parallel machine environment with non-renewable resource constraints.

To our best knowledge, no exact method has been described for our problem in the literature. However, for a related problem, where some of the jobs produce, while other jobs consume some non-renewable resources (and there are no replenishments from external sources) Briskorn et al. (2013) propose an exact method for minimizing the total weighted completion time of the jobs. In the more general project scheduling setting, Neumann & Schwindt (2003) study the makespan minimization problem with inventory constraints, and describe a branch-and-bound method for solving it.

Single machine scheduling with the maximum lateness objective is polynomially solvable by ordering the jobs in earliest due-date order, see Jackson (1955). In spite of the existence of a polynomial time algorithm, we are not aware of any linear programming based method of polynomial time complexity, in which the convex hull of feasible solutions is determined by a linear system such that the

coefficients of the variables are determined by polynomial functions of the problem data, i.e., passing from one input to the other with the same number of jobs only means that we change some data in the LP formulation by plugging the problem data into multivariate polynomials, but it is not allowed to insert new constraints, or do some sorting and then fill in the coefficients of the variables and the right-hand-sides in the LP. In fact, Blazewicz et al. (1991) propose a MIP formulation for $1||L_{\max}$ using positional variables. Moreover, a number of alternative formulations are compared and evaluated for single machine scheduling problems with various objective function by Keha et al. (2009). In contrast, for single machine scheduling with the sum of (weighted) job completion times objective, $1||\sum w_j C_j$, an LP formulation is developed by Queyranne (1993) in which the coefficients of the constraints are linear functions of the problem data, while the right-hand-sides are determined by quadratic polynomials of the data. Although the number of inequalities is exponential in the number of jobs, but they can be separated efficiently, so the LP can be solved in polynomial time. We will adapt the inequalities of Queyranne in Section 3.2 to our MIP model.
Main results and structure of the paper. Firstly, we will elaborate upon the modeling of the problem by a mixed-integer linear program (MIP). Since we have to compute the maximum lateness objective, choosing the right MIP model is a non-trivial issue (Section 2). Second, we will devise new inequalities valid for the feasible solutions of the MIP formulation, and also two which may cut off feasible solutions, but they keep at least one optimal solution (Section 3.2). The new inequalities will be used in a branch-and-cut method to strengthen the LP-relaxation of the MIP formulation (Section 3.1). We will also sketch a heuristic method for getting an initial feasible solution as well as an upper bound on the optimum value in Section 3.3. We emphasize that in most papers mentioned above, mathematical programs are used only for modeling the problem, while the methods devised are based on some other representations. In contrast, our branch-and-cut method uses the MIP model as the representation of the problem, and we do not use the solver as a black-box, instead, we generate cutting planes in the course of the solution process in order to speed up the optimization algorithm. Thirdly, we summarize our computational results on a large set of benchmark instances. The goal of the experiments is to determine the limitation of the method, and also to assess the benefit of using cutting planes to strengthen the MIP formulation (Section 4). Finally, we conclude the paper in Section 5.

2. Problem formulation

In this section first we define our problem more formally, then describe our MIP formulation in several steps.

In our scheduling problem there is a single machine, a set of n jobs $\mathcal{J}$, and a set of ρ non-renewable resources $\mathcal{R}$. Each job j has a processing time $p_j > 0$, a due-date $d_j \geq 0$, and resource requirements $a_{ij} \geq 0$ for $i \in \mathcal{R}$. The non-renewable resources are supplied at dates $0 = u_1 < u_2 < \cdots < u_q$, and the

amount supplied from resource $i \in \mathcal{R}$ at date u_ℓ is $\tilde{b}_{i,\ell} \geq 0$. All problem data are non-negative and integer.

A *schedule* specifies the starting time S_j of each job $j \in \mathcal{J}$; it is feasible if (i) the jobs are not preempted, (ii) no pair jobs overlap in time, i.e., $S_{j_1} + p_{j_1} \leq S_{j_2}$ or $S_{j_2} + p_{j_2} \leq S_{j_1}$ for each pair of distinct jobs j_1 and j_2, and (iii) for each resource $i \in \mathcal{R}$, and for each time point t, the total supply until time t is not less than the total consumption of those jobs starting not later than t, i.e., if $u_\ell \leq t$ is the last supply date before t, then $\sum_{j \in J: S_j \leq t} a_{ij} \leq \sum_{k=1}^{\ell} \tilde{b}_{ik}$ for each resource $i \in \mathcal{R}$. We aim at finding a feasible schedule S minimizing the maximum lateness $L_{\max}(S) := \max_{j \in \mathcal{J}} C_j(S) - d_j$, where $C_j(S) = S_j + p_j$ is the completion time of job j in schedule S.

We may assume that for each $i \in \mathcal{R}$, the total demand does not exceed the total supply, i.e., $\sum_{j \in \mathcal{J}} a_{ij} \leq \sum_{\ell=1}^{q} \tilde{b}_{i\ell}$, otherwise no feasible solution exists. The cumulative supply of resource i up to supply date u_ℓ is $b_{i\ell} := \sum_{k=1}^{\ell} \tilde{b}_{ik}$.

Keha et al. (2009) describe 4 distinct MIP formulations for single machine scheduling with the maximum lateness objective ($1||L_{\max}$). None of these formulations take non-renewable resources into account, but any of them could be further developed to model our problem. We have ruled out the time-indexed formulation, since in that model the number of variables linearly depends on the magnitude of the job processing times, and should we extended that model by non-renewable resource constraints, also on the magnitude of the supply dates. After some preliminary tests (we extended each model of Keha et al. by modeling the non-renewable resource constraints), we have chosen the model with *completion time variables*, and we describe it in detail subsequently.

We use three main types of variables in our formulation. Variable C_j denotes the completion time of job $j \in \mathcal{J}$ and for each ordered pair of jobs $j_1, j_2 \in \mathcal{J}$ with $j_1 < j_2$, the binary variable ord_{j_1,j_2} has value 1 if and only if j_1 precedes j_2 in the schedule. Finally, there are $q \cdot |\mathcal{J}|$ binary decision variables $z_{j\ell}$, $j \in \mathcal{J}, \ell = 1, \ldots, q$ to assign jobs to supplies, i.e., $z_{j\ell} = 1$ if and only if job j is started before $u_{\ell+1}$ ($u_{q+1} = \infty$). Then $z_{j\ell} \geq z_{j,\ell-1}$ must hold and if $z_{j\ell} - z_{j,\ell-1} = 1$ then job

j must start after u_ℓ. The MIP formulation is

$$\min L_{\max} \tag{1}$$

s.t.

$$C_j \geq p_j, \quad j \in \mathcal{J} \tag{2}$$

$$C_{j_1} + p_{j_2} \leq C_{j_2} + M \cdot (1 - ord_{j_1,j_2}), \quad j_1, j_2 \in \mathcal{J},\ j_1 < j_2 \tag{3}$$

$$C_{j_2} + p_{j_1} \leq C_{j_1} + M \cdot ord_{j_1,j_2}, \quad j_1, j_2 \in \mathcal{J},\ j_1 < j_2 \tag{4}$$

$$L_{\max} \geq C_j - d_j, \quad j \in \mathcal{J} \tag{5}$$

$$C_j - p_j \geq \sum_{\ell=2}^{q} u_\ell \cdot (z_{j,\ell} - z_{j,\ell-1}), \quad j \in \mathcal{J} \tag{6}$$

$$\sum_{j \in \mathcal{J}} a_{ij} z_{j\ell} \leq b_{i\ell}, \quad \ell = 1, \ldots, q-1,\ i \in \mathcal{R} \tag{7}$$

$$z_{j,\ell-1} \leq z_{j,\ell}, \quad j \in \mathcal{J}, \ell = 2, \ldots, q \tag{8}$$

$$z_{j,q} = 1, \quad j \in \mathcal{J} \tag{9}$$

$$ord_{j_1,j_2} \in \{0,1\}, \quad j_1, j_2 \in \mathcal{J},\ j_1 < j_2 \tag{10}$$

$$z_{j\ell} \in \{0,1\}, \quad j \in \mathcal{J},\ \ell = 1, \ldots, q. \tag{11}$$

The objective is to minimize $L_{\max}$. Constraints (2)-(4) ensure that the jobs do not overlap in time. Inequalities (5) express that $L_{\max}$ is at least $\max_{j \in \mathcal{J}} C_j - d_j$. By (6) for each job j, the starting time $C_j - p_j$ is at least the u_ℓ provided that $z_{j\ell} - z_{j,\ell-1} = 1$. The resource constraints are encoded by (7), since if $z_{j\ell} = 1$ then job j is started before $u_{\ell+1}$, hence, its resource consumption must be satisfied from the cumulative supply $b_{i\ell}$. The rest of the constraints order the $z_{j\ell}$, set $z_{jq} = 1$, since every job is processed before $u_{q+1} = \infty$, and ensure integrality of the variables.

3. Exact solution by branch-and-cut

In this section first we sketch how branch-and-cut, a general algorithmic approach for integer programming, can be applied to our problem, and then we describe a number of valid inequalities (cuts) that can be used to strengthen the LP relaxation of our MIP formulation.

3.1. Overview

One of the most successful methods for solving integer programming problems is branch-and-cut, which is linear-programming based branch-and-bound augmented with the generation of cutting planes in search-tree nodes, see e.g., Crowder et al. (1983); Balas et al. (1993). The main idea is as follows. Consider

a mixed-integer linear program over n variables and with m linear constraints:

$$\begin{aligned}
&\min \sum_{j=1}^{n} c_j x_j \\
&\text{s.t.} \\
&\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, \ldots, m \\
&x_j \in \mathbb{Z}, \quad \text{for } j \in I \\
&0 \leq x_j \leq u_j, \quad \text{for } j \in \{1, \ldots, n\}
\end{aligned}$$

where the c_j, a_{ij}, b_i and u_j are rational numbers, and $I \subseteq \{1, \ldots, n\}$ is the subset of integer variables, i.e., the constraints $x_j \in \mathbb{Z}$ for $j \in I$ prescribe that these variables must take integral values in any feasible solution. Its *LP relaxation* is obtained by dropping the integrality constraints. Let P denote the set of feasible solutions of the LP relaxation, and P_I the closed convex hull of $P \cap \{x \in \mathbb{R} \mid x_j \in \mathbb{Z}, \text{ for } j \in I\}$. It is known that both P and P_I are convex polyhedra, and $P_I \subseteq P$ (Schrijver, 1998). If $P_I \neq P$, then the LP relaxation can be strengthened by inequalities valid for P_I, but cutting off some fragment of P. That is, if $\bar{x}$ is the current optimal solution of the LP relaxation, but $\bar{x} \notin P_I$, then P_I must have a facet separating $\bar{x}$ from P_I (Schrijver, 1998). In fact, if we manage to generate an inequality $\alpha x \geq \beta$ such that (i) it is valid for P_I, i.e., $\alpha x' \geq \beta$ for all $x' \in P_I$, and (ii) it is violated by $\bar{x}$, i.e., $\alpha \bar{x} < \beta$, then we can add this inequality to the LP relaxation to strengthen it. This idea can be used in a linear-programming based branch-and-bound algorithm, both in the root node, and in the search-tree nodes as well. Clearly, adding such an inequality may speed up the search because it may help to increase the lower bound on the optimum, and also to find integer feasible solutions. Further on, if $\alpha x \geq \beta$ is not valid for P_I, but there is a nonempty subset V of *optimal* solutions which satisfy this inequality, then we can still add $\alpha x \geq \beta$ to the LP relaxation, as long as we do it consistently, i.e., we never add an inequality cutting off some solution in V. Then, we are guaranteed that the branch-and-cut procedure, when it terminates, finds an optimal solution, provided the problem is feasible, and has a finite optimum. In our setting, the arising mixed-integer programs always have a finite optimum.

3.2. Cutting planes

Note the form of the resource constraints (7). These are knapsack constraints, there are $(q-1)|\mathcal{R}|$ of them. MIP solvers can take advantage of such constraints by generating *cover cuts* to strengthen the MIP formulation, see Balas (1975).

Since the linear ordering variables express a total ordering of the jobs through the constraints (3)-(4), in any feasible solution they satisfy the 3-dicycle inequalities of Grötschel et al. (1985) (adapted to our convention that ord_{jk} is defined

only if $j < k$):

$$\left.\begin{array}{l} ord_{j_1,j_2} + ord_{j_2,j_3} - ord_{j_1,j_3} \le 1, \\ -ord_{j_1,j_2} - ord_{j_2,j_3} + ord_{j_1,j_3} \le 0 \end{array}\right\} \quad j_1, j_2, j_3 \in \mathcal{J},\ j_1 < j_2 < j_3 \qquad \text{(C1)}$$

Next we define four new classes of cuts to strengthen the LP relaxation. First, we have adapted the cutting planes of Queyranne (1993) to our problem:

$$\sum_{j\in S} p_j(d_{\max}(S) + L_{\max} - C_j + p_j) \ge \frac{1}{2}\left(\sum_{j\in S} p_j^2 + \left(\sum_{j\in S} p_j\right)^2\right), \quad S \subseteq \mathcal{J}, \text{ (C2)}$$

where $d_{\max}(S)$ is the maximum d_j among those jobs in S. To justify these inequalities, first notice that Queyranne gave a complete linear description of the polytope with vertex set consisting of all the possible completion times of a set of jobs $\mathcal{J}$ scheduled consecutively without idle times. The description consists of the inequalities

$$\sum_{j\in S} p_j C_j \ge \frac{1}{2}\left(\sum_{j\in S} p_j^2 + \left(\sum_{j\in S} p_j\right)^2\right), \quad S \subseteq \mathcal{J}. \qquad \text{(Q)}$$

Now, consider any subset S of the jobs and suppose that we reverse time, and schedule them backward from $d_{\max}(S)+L_{\max}$. Since $d_{\max}(S)+L_{\max}$ is an upper bound on the completion time C_j of any job $j \in S$ in any feasible solution of the MIP formulation, the completion time of j in the backward schedule is $d_{\max}(S) + L_{\max} - (C_j - p_j)$, that is, $d_{\max}(S) + L_{\max}$ minus the starting time of the job. Plugging this into (Q) yields (C2).

The second class consists of cuts derived using a pair of jobs and their impact on $L_{\max}$:

$$L_{\max} \ge C_{j_1} - (p_{j_2} - d_{j_2} + d_{j_1}) \cdot (1 - ord_{j_1,j_2}) + p_{j_2} - d_{j_2}, \quad j_1, j_2 \in \mathcal{J}, j_1 < j_2. \text{ (C3)}$$

To see validity, first suppose that $ord_{j_1,j_2} = 0$ in a feasible solution of MIP. Then the right-hand-side of (C3) equals $C_{j_1} - d_{j_1}$, and by (5), $L_{\max} \ge C_{j_1} - d_{j_1}$, so the MIP solution satisfies (C3). On the other hand, if $ord_{j_1,j_2} = 1$, then the right-hand-side equals $C_{j_1} + p_{j_2} - d_{j_2}$. Since $C_{j_1} + p_{j_2} \le C_{j_2}$ by (3), and $L_{\max} \ge C_{j_2} - d_{j_2}$ by (5), any feasible solution of MIP with $ord_{j_1,j_2} = 1$ must satisfy (C3).

The cuts in the third and fourth class may cut off feasible solutions, but they leave at least one optimal solution. That is, the cuts in the third class cut off solutions in which two jobs both starting in some interval $[u_\ell, u_{\ell+1}]$ are not in earliest due-date (EDD) order:

$$ord_{j_1,j_2} \ge z_{j_1,\ell+1} - z_{j_1,\ell} + z_{j_2,\ell+1} - z_{j_2,\ell} - 1, \quad j_1, j_2 \in \mathcal{J},\ d_{j_1} < d_{j_2}, \qquad \text{(C4)}$$

where we assume that $d_{j_1} < d_{j_2}$ implies $j_1 < j_2$.

Cuts in the fourth class cut off solutions in which the total processing time of those jobs starting in some interval $[u_\ell, u_{\ell+1}]$ is more than $u_{\ell+1} - u_\ell + p_{\max}$, since there always exists an optimal solution respecting this bound:

$$\sum_{j \in \mathcal{J}} (z_{j,\ell} - z_{j,\ell-1}) \cdot p_j \leq u_{\ell+1} - u_\ell + p_{max}, \quad \ell = 1, \ldots, q. \tag{C5}$$

3.3. Initial upper bound

We use an EDD based algorithm to obtain an initial upper bound. In this algorithm we consider the jobs one-by-one in EDD order and schedule the next job j at the earliest moment t where the following conditions hold: (i) the machine is idle in $[t, t + p_j]$ and (ii) the resulting partial schedule does not violate any of the resource constraints, i.e., we have enough resource for all the scheduled jobs. Note that $[t, t + p_j]$ is not necessarily the earliest idle interval where the actual amount of raw-materials in the stock is sufficient to schedule j, since we may use these materials later for other jobs.

4. Computational results

4.1. Test instances

We have generated instances for the following (n, q) pairs (job numbers and supply dates): $(30, 3)$, $(30, 10)$, $(50, 5)$, $(50, 10)$, $(100, 10)$ and $(100, 20)$. For each case, we examined instances with 1, 3 and 10 resources. In every instance the jobs have randomly generated processing times between 1 and 50, resource requirements between 0 and 50 and due-dates between 0 and $\sum_{j \in \mathcal{J}} p_j - 1$. Each quantity was generated independently to the others. The upper bound on the due-dates implies that the maximum lateness is positive for every feasible schedule, since there must be at least one job that completes not earlier than $\sum_{j \in \mathcal{J}} p_j$. We have equidistant supply dates between 0 and $\sum_{j \in \mathcal{J}} p_j$ and for each resource i we divide $\sum_{j \in \mathcal{J}} a_{ij}$ units from resource i among the supply dates randomly.

4.2. Implementation

We have implemented our method in the Mosel language of FICO Xpress (2016) (version 7.9) and we run all experiments on workstation with Intel Xeon X5650 @ 2.67GHz CPU, 4GB RAM, and Debian 6 Linux operating system. All experiments used only a single cpu thread. We have run experiments with and without our cutting planes, and also by enabling or disabling the built-in cuts of the solver. However, we disabled presolve in order to have a more clear evaluation of the cutting planes. Since it was clear after a few runs that an initial upper bound can greatly improve the results, in all experiments (either using our cuts, or not) we bounded the objective function value by the upper bound obtained by our heuristic method. More precisely, we have set the bound to that computed by the heuristic minus 1, since the optimum value is always integral, and in this way we ensure that during the search a better upper bound is sought,

and also, if the optimum value matches the initial upper bound, optimality can be proved faster. For 30 and 50 job instances, the run time limit was set to 600 seconds, and for 100 job instances to 1200 seconds. In the next paragraphs we sketch our algorithm, whose schematic view is depicted in Figure 2.

First, the algorithm determines an initial upper bound as it is described in Section 3.3. Then it formulates a MIP (Section 2) augmented with an upper bound on the objective function value using the output of the heuristic. In addition, all the inequalities (C5) are included into it, since preliminary experiments showed that we get better results if they are included in the initial formulation. Note that there are only q such inequality thus this modification increases the size of the problem only marginally (the original MIP contains $O(n^2 + q\rho)$ inequalities). However, we cannot do the same with the other cutting planes, since their number is considerably greater. After that, the algorithm proceeds with the branch-and-cut procedure.

Branch-and-cut, as explained in Section 3.1, is a tree-search procedure augmented with the separation of cutting planes in some search-tree nodes. In our implementation, in the root and in every fourth node (of depth at most 100) the algorithm tries to separate violated inequalities from some of the classes proposed in Section 3.2. Note that *separation* means that in each class, cuts are sought which are violated by the optimal solution of the LP in the search-tree node. Violated cuts, if any, are added to the LP of the node, and non-tight cuts inherited from ancestor search-tree nodes are deleted. Finally, reoptimization takes place. The separation procedures are called from a so-called *callback function*, which, as its name suggests, is called from whithin the MIP solver after solving the LP in each search-tree node. The separation of our cutting planes is quite easy, except the separation of (C2), where we have adapted the method for (Q) as described in Queyranne (1993). Sometimes, the separation of some classes of cutting planes is not beneficial, hence we have examined several settings and we summarize our experience on different types of problem instances.

Computational results are summarized in the following sections and the used cutting plane classes will be mentioned there.

4.3. Results with 30 jobs

We have summarized the 30 job results in Table 1 (3 supply dates) and in Table 2 (10 supply dates), where the rows depict the results in case of four settings: without generating any cutting planes ('no cuts'), enabling the built-in cuts, but not using our cuts ('Xpress'), disabling the built-in cuts, but using cuts (C3) and (C4) ('our'), and using both the built-in and our cuts ('Xpress+our'). These tables depict the number of the optimally solved instances (out of 10) and the average computation time (in seconds) of the different methods on those instances which were solved optimally by all the methods.

The first rows in Table 1 and Table 2 show that most of the instance were solved optimally within a few seconds even if we did not generate any cutting planes. However, the cuts are useful even in this case. In case of 3 supply dates we have solved each of the instances with our cutting planes and significantly

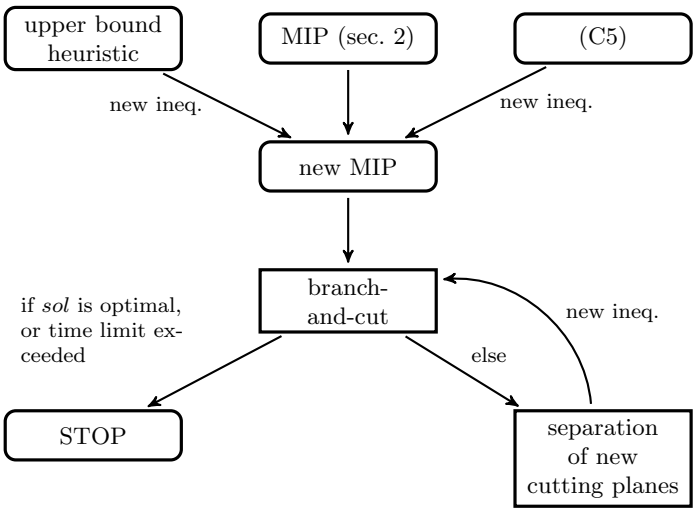

Figure 2: Schematic description of our method

Table 1: Results with 30 jobs and 3 supply dates.

30 jobs	1 resource		3 resources		10 resources	
3 supply dates	# opt	time (s)	# opt	time (s)	# opt	time (s)
no cuts	8	1.90	9	2.11	7	4.54
Xpress	9	1.42	9	3.69	7	2.74
our[a]	10	1.64	10	0.63	10	1.18
our[a] + Xpress	10	1.64	10	1.20	10	1.18

[a] cuts (C3) + (C4)

decreased the average running time. If there are 10 supply dates the results depend on the number of the resources: in case of 1 resource generating our cutting planes requires some time, but do not improve the results, in case of 3 resources generating the built-in cuts requires a lot of time, while our cutting planes greatly decrease it and solve the instance that remained unsolved in the previous settings. If there are 10 resources then we can observe only smaller differences among the results of the different methods.

Table 2: Results with 30 jobs and 10 supply dates.

30 jobs	1 resource		3 resources		10 resources	
10 supply dates	# opt	time (s)	# opt	time (s)	# opt	time (s)
no cuts	9	5.25	9	12.25	9	2.06
Xpress	9	6.34	9	36.75	10	1.87
our[a]	9	13.07	10	6.06	10	1.78
our[a] + Xpress	9	13.28	10	5.06	10	2.09

[a] cuts (C3) + (C4)

Table 3: Results with 50 jobs and 5 supply dates.

50 jobs	1 resource		3 resources		10 resources	
5 supply dates	# opt	gap	# opt	gap	# opt	gap
no cuts	7	1.057	4	1.040	6	1.038
Xpress	7	1.073	5	1.035	9	1.026
our[a]	6	1.052	6	1.025	8	1.011
our[a] + Xpress	7	1.054	8	1.024	7	1.014

[a] cuts (C3) + (C4)

Table 4: Results with 50 jobs and 10 supply dates.

50 jobs	1 resource		3 resources		10 resources	
10 supply dates	# opt	gap	# opt	gap	# opt	gap
no cuts	6	1.43	8	1.035	5	1.049
Xpress	7	1.77	7	1.007	5	1.039
our[a]	5	1.20	7	1.021	7	1.029
our[a] + Xpress	7	1.22	6	1.028	7	1.018

[a] cuts (C2) + (C3) + (C4)

4.4. Results with 50 jobs

If we have 50 jobs instead of 30, the number of the instances that each setting solves optimally significantly decreases (the results are obtained from 10-10 instances again), thus we characterize the results by the average integrality gap instead of the average required time by the optimally solved instances. The results can be seen in Table 3 (in case of 5 supply dates) and in Table 4 (10 supply dates).

In case of 5 supply dates and 1 resource the usage of the Xpress cuts increases the integrality gap, while our cuts slightly decrease it (second column of Table 3). If we have 3 or 10 resources, then our cuts significantly decrease the gaps and increase the number of the optimally solved instances (last four columns of Table 3), while enabling the Xpress cuts causes slighter improvement.

If the number of the supply dates is 10, then we can obtain somewhat different observations, see Table 4. Again, our cuts significantly decrease the integrality gap in every case, however the Xpress cuts produce varied results: the usage of these cuts considerably worsen the results in case of 1 resource, greatly improve them in case of 3 resources and moderately improve them in case of 10 resources.

4.5. Results with 100 jobs

These instances are much harder, thus we have increased the time limit to 1200 seconds to manifest the differences among the different settings. If there are 10 supply dates and one or three resources then our cutting planes clearly improve the results, while in case of 20 supply dates or 10 resources there are only minor differences among the settings, that is, the cutting planes do not help too much. Note that the Xpress cuts are very useful in case of 3 resources like in case of 50 jobs and 10 supply dates.

Table 5: Results with 100 jobs and 10 supply dates.

100 jobs	1 resource		3 resources		10 resources	
10 supply dates	# opt	gap	# opt	gap	# opt	gap
no cuts	2	1.242	3	1.088	2	1.091
Xpress	2	1.212	4	1.043	2	1.094
our[a]	2	1.188	2	1.058	2	1.091
our[b] + Xpress	3	1.175	4	1.053	2	1.091

[a] cuts (C2) + (C3) + (C4)
[b] cuts (C4)

Table 6: Results with 100 jobs and 20 supply dates.

100 jobs	1 resource		3 resources		10 resources	
20 supply dates	# opt	gap	# opt	gap	# opt	gap
no cuts	2	1.72	1	1.26	1	1.12
Xpress	3	1.69	2	1.21	1	1.11
our[a]	2	1.70	2	1.22	1	1.11
our[a] + Xpress	2	1.80	2	1.24	1	1.12

[a] cuts (C4)

4.6. Further observations

We have examined several other settings and we have some further observations about our cutting planes. The set of cutting planes (C1) never helped, while (C4) was almost always useful, see Figure 3. This figure depicts the average gaps on the 50 job instances with and without generating cutting planes (C4). We can see that generating these cutting planes almost always greatly decreases the gaps and increases it (slightly) only in case of 10 supply dates and 10 resources.

It is surprising that the instances with only one resource proved to be much harder than the instances with more resources. Usually, the difference between gaps reached in case of 3 and 10 resources is less than the difference between that for 1 and 3 resources.

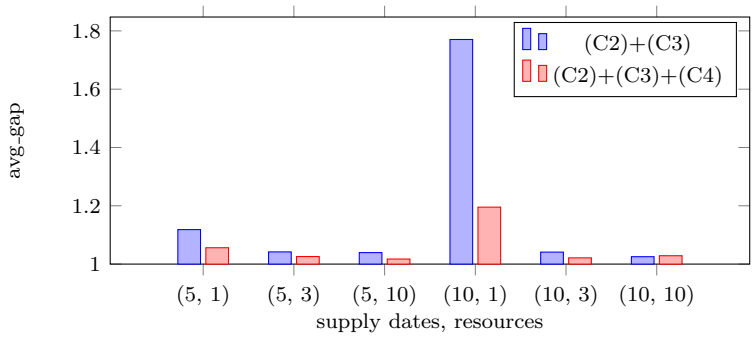

Figure 3: Average gaps in case of 50 jobs. Each bar represents the average of 10 instances.

5. Conclusions

We have implemented a branch-and-cut algorithm for the single-machine scheduling problem with resource consuming jobs and the maximum lateness objective. The results show that the proposed cutting planes mainly decrease the integrality gap, or in case of 30 job instances, decrease the computation time. This statement remains valid even if we enable the built-in cuts of Xpress. Note that in several cases, we have achieved the best results without the built-in cuts of the solver.

As future work we will consider the problem with total weighted completion time, which is apparently an even harder problem.

Acknowledgments

This work has been supported by the National Research, Development and Innovation Office – NKFIH, grant no. K112881, and by the GINOP-2.3.2-15-2016-00002 grant of the Ministry of National Economy of Hungary.

Balas, E. (1975). Facets of the knapsack polytope. *Mathematical programming*, *8*, 146–164.

Balas, E., Ceria, S., & Cornuéjols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical programming*, *58*, 295–324.

Belkaid, F., Maliki, F., Boudahri, F., & Sari, Z. (2012). A branch and bound algorithm to minimize makespan on identical parallel machines with consumable resources. In *Advances in Mechanical and Electronic Engineering* (pp. 217–221). Springer. doi:10.1007/978-3-642-31507-7-36.

Blazewicz, J., Dror, M., & Weglarz, J. (1991). Mathematical programming formulations for machine scheduling: a survey. *European Journal of Operational Research*, *51*, 283–300.

Briskorn, D., Jaehn, F., & Pesch, E. (2013). Exact algorithms for inventory constrained scheduling on a single machine. *Journal of Scheduling*, *16*, 105–115. doi:10.1007/s10951-011-0261-x.

Carlier, J. (1984). *Problèmes d'ordonnancements à contraintes de ressources: algorithmes et complexité. Thèse d'état*. Université Paris 6.

Crowder, H., Johnson, E. L., & Padberg, M. (1983). Solving large-scale zero-one linear programming problems. *Operations Research*, *31*, 803–834.

FICO Xpress (2016). Optimization suite. www.fico.com/xpress.

Gafarov, E. R., Lazarev, A. A., & Werner, F. (2011). Single machine scheduling problems with financial resource constraints: Some complexity results and properties. *Mathematical Social Sciences*, *62*, 7–13. doi:10.1016/j.mathsocsci.2011.04.004.

Grigoriev, A., Holthuijsen, M., & van de Klundert, J. (2005). Basic scheduling problems with raw material constraints. *Naval Research of Logistics*, *52*, 527–553. doi:10.1002/nav.20095.

Grötschel, M., Jünger, M., & Reinelt, G. (1985). Facets of the linear ordering polytope. *Mathematical Programming*, *33*, 43–60.

Györgyi, P., & Kis, T. (2015a). Reductions between scheduling problems with non-renewable resources and knapsack problems. *Theoretical Computer Science*, *565*, 63–76. doi:10.1016/j.tcs.2014.11.007.

Györgyi, P., & Kis, T. (2015b). Approximability of scheduling problems with resource consuming jobs. *Annals of Operations Research*, *235*, 319–336. doi:10.1007/s10479-015-1993-3.

Györgyi, P., & Kis, T. (2017). Approximation schemes for parallel machine scheduling with non-renewable resources. *European Journal of Operational Research*, *258*, 113–123.

Jackson, J. R. (1955). Scheduling a production line to minimize maximum tardiness. *Research Report 43, Management Science Res. Project, UCLA*, .

Keha, A. B., Khowala, K., & Fowler, J. W. (2009). Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering*, *56*, 357–367.

Neumann, K., & Schwindt, C. (2003). Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, *56*, 513–533. doi:10.1007/s001860200251.

Queyranne, M. (1993). Structure of a simple scheduling polyhedron. *Mathematical Programming*, *58*, 263–285.

Schrijver, A. (1998). *Theory of linear and integer programming*. John Wiley & Sons.

Slowinski, R. (1984). Preemptive scheduling of independent jobs on parallel machines subject to financial constraints. *European Journal of Operational Research*, *15*, 366–373. doi:10.1016/0377-2217(84)90105-X.

Stadtler, H., & Kilger, C. (2008). *Supply Chain Management and Advanced Planning. Concepts, Models, Software, and Case Studies*. (4th ed.). Springer.

Toker, A., Kondakci, S., & Erkip, N. (1991). Scheduling under a non-renewable resource constraint. *Journal of the Operational Research Society*, *42*, 811–814. doi:10.2307/2583664.

Xie, J. (1997). Polynomial algorithms for single machine scheduling problems with financial constraints. *Operations Research Letters*, *21*, 39–42. doi:10.1016/S0167-6377(97)00007-2.