Integer Programming and Constraint Programming in Solving a Multi-Machine Assignment Scheduling Problem with Deadlines and Release Dates

Ruslan Sadykov, Laurence Wolsey

November 1, 2003

Abstract

We consider both branch-and-cut and column generation approaches for the problem of finding a minimum cost assignment of jobs with release dates and deadlines to unrelated parallel machines. Results are presented for several variants both with and without Constraint Programming. Among the variants, the most effective strategy is to combine a tight and compact, but approximate, Mixed Integer Programming formulation with a global constraint testing single machine feasibility. All the algorithms have been implemented in the Mosel modelling and optimization language.

1 Introduction

The problem considered here is a standard (and academic) scheduling problem in which jobs with costs must be assigned optimally to unrelated machines while satisfying release dates and deadlines. The objective is to find a minimum cost assignment of the jobs to the machines such that all release dates and deadlines are met. This particular problem has already been studied by Jain and Grossman [8] and by Bockmayr and Pisaruk [3] from a combined Integer Programming (IP)/Constraint Programming (CP) viewpoint. A similar problem, but without release dates and with machine-independent

1

assignment costs, has been tackled by Chen and Powell [5] using column generation.

In this paper we pursue the exploration of algorithms to solve such problems. Our starting point was three simple observations

i) for a problem with suitable structure, the weakness of infeasibility or "no good" cuts, which only cut off one or a small set of feasible solutions without giving any structural information, suggests that a column generation approach with optimization (rather that feasibility) subproblems should provide much better information, and potentially stronger lower bounds

ii) the single item subproblem is in practice a difficult scheduling problem $1|r_j|\sum_j w_j U_j$, namely the problem of minimizing the weighted sum of late jobs

iii) the power of recent modelling languages such as MOSEL makes it relatively easy to test quickly a wide range of hybrid approaches.

It turns out that there is a considerable range of possible IP and IP/CP algorithms for tackling the above problem, which may in turn be suggestive of ways to tackle other related or more difficult problems.

We now outline the contents of this paper. In section 2 we describe seven algorithmic variants based on a MIP solver and a CP global constraint testing feasibility of a single machine job assignment. The first two are pure MIP formulations, the next two can be classified as MIP branch-and-cut approaches with CP, the next two as column generation or branch-and-price algorithms, and the final one as branch-and-price and CP (where each subproblem is solved by branch-and-cut with CP). In section 3 we discuss how to tighten the MIP formulation of the multi-machine problem, and the corresponding single machine subproblem. In section 4 we present implementation details and computational results with the seven algorithms, and we terminate with a discussion of further directions of research.

The tentative conclusions suggested by the computational results are:

i) when using an IP algorithm, it is important to develop as tight an IP/MIP formulation as possible;

ii) when using an IP/CP algorithm in which the IP optimizes over a relaxation (superset) of the set of feasible schedules and CP tests feasibility, it is important both to tighten the IP formulation and also to control its size: a weak formulation results in a large number of feasibility tests, an overly large formulation may lead to long LP solution times, and in both cases overall solution time may increase

iii) using either an IP/CP algorithm with a tightened IP, or a column generation algorithm with a combined IP/CP algorithm for the subproblem, it is possible to solve instances with up to 50 jobs and 9 machines.

2 Various MIP and IP/CP Algorithms

Here we describe the multi-machine assignment scheduling problem (MMASP). A set $N = \{1, \ldots, n\}$ of jobs have to be processed on a set $K = \{1, \ldots, k\}$ of machines. Any job can be processed on any machine and each machine can only process one job at a time. Processing of a job $j \in N$ can only begin after its release date r_j and must be completed at the latest by its deadline d_j. The processing cost and the processing time of job $j \in N$ on machine $m \in K$ are c_j^m and p_j^m respectively. The objective is to minimize the total cost of processing of all the jobs. Using the standard scheduling notation, and writing d_j for the deadlines, MMASP can be written as $R \mid r_j, d_j \mid \sum c_{ij}$. We assume throughout that $p_j^m \leq d_j - r_j$ for all $m \in K$ and $j \in N$.

The problem MMASP can be modelled as a MIP. The binary variable x_j^m is equal to one when job j is assigned to machine m. Using these assignment variables, we obtain the first problem representation (IP).

IP:

$$\min \quad \sum_{m \in K} \sum_{j \in N} c_j^m x_j^m \tag{1}$$

$$s.t. \quad \sum_{m \in K} x_j^m = 1 \quad \forall j \in N \tag{2}$$

$$x^m \in X^m \quad \forall m \in K \tag{3}$$

where $x^m \in X^m$ if and only if the corresponding set of jobs $J^m = \{j \in N : x_j^m = 1\}$ is feasible on machine m (can be processed without violation of release dates and deadlines).

A standard way to represent the set $X^1 \times \ldots \times X^k$ as a MIP is to introduce additional binary variables y_{ij}, which indicate whether job i precedes job j when both the jobs are on the same machine. The detailed MIP formulations will be presented in Section 3. Below we denote the assignment constraints (2) by $Ax = b$, and the scheduling constraints describing (3) by $Bx + Cy \leq g$.

In the next two subsections we describe the seven algorithms without specifying in detail the actual MIP formulation $Bx + Cy \leq g$ used to represent the set $X^1 \times \ldots \times X^k$.

2.1   Branch and Cut Algorithms

Algorithm 1. MIP

Here we take a standard MIP formulation and directly apply a commercial MIP system using linear programming based branch-and-cut.

MIP:

$$
\begin{aligned}
\min \quad & cx \\
s.t. \quad & Ax = b \\
& Bx + Cy \leq g \\
& x \in \{0,1\}^{k \times n}, y \in \{0,1\}^{n \times n}.
\end{aligned}
$$

Here additional variables y are necessary to provide a correct IP formulation, but such formulation are known to provide weak bounds.

Algorithm 2. MIP^+

Here we suppose that it is possible to tighten the formulation in the x-space by adding a set of constraints $Fx \leq f$ such that a significant percentage of the solutions $(x^1, \ldots, x^k)$ of

$$
Ax = b, \quad Fx \leq f, \quad x \in \{0,1\}^{k \times n}
$$

have the property that $x^m \in X^m$ for all $m \in K$ (i.e. correspond to feasible schedules). We solve the tightened formulation (TMIP) by a standard MIP solver.

TMIP:

$$\min \quad cx$$
$$\text{s.t.} \quad Ax = b$$
$$Fx \leq f$$
$$Bx + Cy \leq g$$
$$x \in \{0,1\}^{k \times n}, y \in \{0,1\}^{n \times n}$$

Ways to tighten the formulation are discussed in Section 3.

Algorithm 3. IP/CP

Now we consider the first hybrid formulation (IPCP).

IPCP:

$$\min \quad cx$$
$$\text{s.t.} \quad Ax = b$$
$$disjunctive(m, N_m) \ \forall m \in K$$
$$x \in \{0,1\}^{k \times n}$$

where $N_m = \{j \in N : x_j^m = 1\}$.

Here $disjunctive(m, N_m)$ denotes a global constraint based on Constraint Programming, which tests whether or not a given set of jobs N_m can be carried out on a single machine m so as to satisfy release dates and deadlines. Such constraints are sometimes based on Carlier's algorithm [4] which solves the 1-machine scheduling problem of minimizing the maximum lateness, and thus proves that a set of jobs is feasible by showing that the maximum lateness is zero.

In this hybrid approach, the IP $\min\{cx : Ax = b, x \in \{0,1\}^{k \times n}\}$ is fed to the MIP solver. If x^* is the linear programming solution at a node of the branch-and-cut tree, suppose that there exists a set J^m of jobs with $\sum_{j \in J^m} x_j^{m*} > |J^m| - 1$ for one or more machines m. The set (J^m, p^m, r, d) is sent to the *disjunctive* global constraint. If the set J^m cannot be processed, the "no-good" or infeasibility cut

$$\sum_{j \in J^m} x_j^m \leq |J^m| - 1. \tag{4}$$

is added to the IP as a globally valid cut, and the tree search is continued.

This is the hybrid approach adopted by Jain and Grossman [8] and by Bockmayr and Pisaruk [3]. The former tested a very simple version in which the optimal IP solution was tested for feasibility and the IP was rerun from scratch whenever more cuts were added. The latter tested a more sophisticated version in which even fractional LP solutions are rounded so as to obtain a machine assignment to be checked for feasibility. Note that when the LP solution at a node is integer, the global constraint can be called for each machine $m \in K$ and job set $J^m = \{j \in N : x_j^m = 1\}$.

Algorithm 4. IP^+/CP

This variant is the same as Algorithm 3, except that the improved formulation (TIPCP) is used.

TIPCP:

$$
\begin{aligned}
\min \quad & cx \\
s.t. \quad & Ax = b \\
& Fx \leq f \\
& disjunctive(m, N_m) \ \forall \ m \in K \\
& x \in \{0,1\}^{k \times n}
\end{aligned}
$$

Here again the motivation for using a tightened formulation is that the number of infeasible assignments generated and thus the number of infeasibility cuts needed will decrease.

2.2 Column Generation (Branch-and-Price) Algorithms

Given the structure of MMASP, it is very natural to think of a column generation approach. The use of Dantzig-Wolfe decomposition to solve scheduling problems is not new. This approach was used by Chen and Powell [5] in tackling problems similar to MMASP, and an example of constraint programming based column generation can be found in [9].

Here we give a very brief description of the Dantzig-Wolfe decomposition algorithm for MMASP. We decompose MMASP into a master problem and K subproblems. To obtain the master problem we rewrite the initial model (IP) in a different way. Let T^m denote the set of all feasible partial schedules

on machine m, $m \in K$. Let h_t^m be the total cost of a partial schedule t, $t \in T^m$. For each job j, $j \in N$, let $a_{jt}^m = 1$ if the partial schedule t, $t \in T^m$, covers the job j, and 0 otherwise.

Define the variables: $\lambda_t^m = 1$ if a solution of the master problem includes partial schedule t, and $\lambda_t^m = 0$ otherwise. The master problem (MP) is then as follows:

MP:

$$\max \quad \sum_{m \in K} \sum_{t \in T^m} -h_t^m \lambda_t^m \tag{5}$$

$$s.t. \quad \sum_{m \in K} \sum_{t \in T^m} a_{jt}^m \lambda_t^m = 1 \quad \forall j \in N \tag{6}$$

$$\sum_{t \in T^m} \lambda_t^m \leq 1 \quad \forall m \in K \tag{7}$$

$$\lambda_t^m \in \{0,1\} \quad \forall m \in K, \forall t \in T^m \tag{8}$$

The constraints (6) specify that each job lies in precisely one partial schedule and the constraints (7) guarantee that each machine is occupied by at most one partial schedule.

The standard approach to avoid solving (MP) with this very large number of unknown columns is to iterate between solution of the linear programming restriction (RMP) of the master problem involving only a subset of the columns and solution of subproblems (SP^m) for $m \in K$. The subproblem (SP^m) is used to generate an additional column in T^m with positive reduced cost in (RMP), or to prove that no such column exists. If we denote the optimal dual variables of (RMP) by $(\pi, \mu) \in \mathcal{R}^{n+k}$, the subproblem can be written as:

SP^m:

$$\zeta^m = \quad \max \quad \sum_{j \in N} (-c_j^m - \pi_j) x_j^m - \mu_m \tag{9}$$

$$s.t. \quad (x_1^m, \ldots, x_k^m) \in X^m. \tag{10}$$

If $\zeta^m > 0$, the corresponding schedule is added to (RMP). The linear programming relaxation of MP has been solved when $\zeta^m = 0$ for all $m \in K$.

Finally as the linear programming solution of MP may not be integer, it is necessary to embed the solution of the linear programming relaxation

of MP within a branch-and-bound tree, giving a so-called *branch-and-price* algorithm. Further details of our very straight-forward implementation are given in Section 4.

Now we consider the *subproblem* and how it is solved. This is crucial to all variants of the branch-and-price algorithm because the subproblem has to be solved many times.

Subproblem (SP^m) involves finding a subset of jobs that can be scheduled on machine m satisfying the release and deadlines, such that the sum of the modified costs $(-c_j^m - \pi_j)$ of jobs in this schedule is maximized. This can be modelled as a single machine problem with release dates and deadlines with the criteria of minimizing the weighted number of late jobs, denoted $1 \mid r_j \mid \sum w_j U_j$ problem, with $w_j = -(c_j^m - \pi_j)$. This problem is known to be NP-hard in the strong sense even when $w_j = 1$ for all $j \in N$ [7].

For such a branch-and-price algorithm, several questions arise:

- Are there effective algorithms to solve the subproblem rapidly?

- If so, does the Linear Programming relaxation of (MP) provide a tight lower bound on the optimal value, in contrast to the weak bound obtained using a Branch-and-Cut approach?

- Does the LP relaxation of (MP) require very many iterations to converge?

- Does the set of active columns at the optimal solution of the LP relaxation of (MP) provide a good set of schedules for the construction of a good multi-machine assignment?

With these questions in mind, and especially the issue of the speed of solution of the subproblem, we now present three branch-and-price variants. Note that we did not dispose of a global constraint able to solve (SP^m), though such a constraint has been developed by Baptiste et al. [1].

Algorithm 5. CG-MIP

This is the standard column generation IP approach in which the subproblems are solved as MIPs using the single machine version of the basic (MIP) model from the Algorithm 1.

Algorithm 6. CG-MIP$^+$

Here subproblems are again solved as MIPs using the single machine version of the tightened (TMIP) model from the Algorithm 2.

Algorithm 7. CG-MIP$^+$/CP

Here we apply the combined MIP/CP approach to the subproblem using the strengthened IP formulation (TIPCP) adapted for the single machine case, combined with the *disjunctive* global constraint to test feasibility at the (integer) nodes of the branch-and-cut tree.

3 MIP Formulations

3.1 Multi-machine case

Here we present the basic formulation of MMASP used by Jain and Grossman [8] and by Bockmayr and Pisaruk [3]. Their two formulations are essentially the same.

Remember that there are *assignment* variables:
$x_j^m = 1$ if job j is assigned to machine m, and $x_j^m = 0$ otherwise.
In addition there are *sequencing* variables:
$y_{ij} = 1$ if jobs i and j are assigned to the same machine and job i precedes job j, with $y_{ij} = 0$ otherwise.
s_j and e_j denote the *start* and *end* times of job $j \in N$.
The following formulation is standard:

$$\min \quad \sum_{m \in K} \sum_{j \in N} c_j^m x_j^m \tag{11}$$

$$s.t. \quad \sum_{m \in K} x_j^m = 1 \quad \forall j \in N \tag{12}$$

$$\sum_{j \in N} p_j^m x_j^m \leq \max_{j \in N} d_j - \min_{j \in N} r_j \quad \forall m \in K \tag{13}$$

$$s_j + \sum_{m \in K} p_j^m x_j^m - e_j = 0 \quad \forall j \in N \tag{14}$$

$$e_i - s_j + U y_{ij} \leq U \quad \forall i, j \in N, i \neq j \tag{15}$$

$$r_j \leq s_j \quad \forall j \in N \tag{16}$$

$$d_j \geq e_j, \quad \forall j \in N \tag{17}$$

$$y_{ij} + y_{ji} \leq 1 \quad \forall i, j \in N, i < j \tag{18}$$

$$x_i^m + x_j^m - y_{ij} - y_{ji} \leq 1 \quad \forall i, j \in N, i < j, \forall m \in K \tag{19}$$

$$x_i^l + x_j^m + y_{ij} + y_{ji} \leq 2 \quad \forall l, m \in K, l \neq m, \forall i, j \in N, i < j \quad (20)$$

$$x_j^m \in \{0, 1\} \quad \forall m \in K, \forall j \in N \quad (21)$$

$$y_{ij} \in \{0, 1\}, \quad \forall i, j \in N, i \neq j. \quad (22)$$

Let X^{MM} denote the feasible region (12)-(22). Here the equalities (12) enforce the assignment of each job to exactly one machine. The inequalities (13) are valid inequalities stating that the total processing time of all the jobs that are assigned to machine m should be less than the difference between the latest deadline and the earliest release date. The constraints (14) relate the start and completion times. The sequencing constraints (15) ensure that processing of job j begins after processing of job i if $y_{ij} = 1$. Here U is a big value and can be, for example, instantiated as the difference between the maximum deadline and the minimum release date. Constraints (16) and (17) enforce the release dates and deadlines. The constraints (18) insure that only one of two jobs i, j is processed before the other. The constraints (19) and (20) relate x and y variables: the first group ensures that if jobs i and j are assigned to machine m, then one must be processed before the other; the second group guarantee that the sequencing variables y_{ij} and y_{ji} are both zero if jobs i and j are assigned to different machines.

Now we consider ways to tighten this formulation in the space of the x variables. A first basic inequality states the obvious fact that the sum of the processing times of all jobs that must be processed within the interval $[r_i, d_j]$ cannot exceed the length of the interval.

Proposition 1 *Consider a pair of jobs $i, j \in N$ with $r_i < d_j$ and let $S_{ij} = \{l \in N : r_l \geq r_i \text{ and } d_l \leq d_j\}$. The inequality*

$$\sum_{l \in S_{ij}} p_l^m x_l^m \leq d_j - r_i \quad (23)$$

is valid for X^{MM} for all $m \in K$.

Now we consider two ways to strengthen inequality (23). First suppose that for some set $S \subseteq N$, $\mid S \mid \geq 2$, $\arg\min_{j \in S} r_j = \arg\max_{j \in S} d_j = j_S$. Let $\delta_S = \min_{j \in S \setminus \{j_S\}} r_j - r_{j_S} \geq 0$ and $\epsilon_S = d_{j_S} - \max_{j \in S \setminus \{j_S\}} d_j \geq 0$.

Proposition 2 *If $\min(\delta_S, \epsilon_S) > 0$ and*

$$p_{j_S}^m \leq d_{j_S} - r_{j_S} - (\delta_S + \epsilon_S) + \max\{\delta_S, \epsilon_S\}, \quad (24)$$

then the inequality

$$\sum_{l \in S} p_l^m x_l^m \le d_{j_S} - r_{j_S} - (\delta_S + \epsilon_S) + (\max\{\delta_S, \epsilon_S\}) x_{j_S}^m \tag{25}$$

is valid for X^{MM} for all $m \in K$.

Proof. Consider a point (x, y) with $x_{j_S}^m = 0$. Then the inequality (25) reduces to

$$\sum_{l \in S \setminus \{j_S\}} p_l^m x_l^m \le d_{j_S} - r_{j_S} - (\delta_S + \epsilon_S),$$

or to

$$\sum_{l \in S \setminus \{j_S\}} p_l^m x_l^m \le \max_{j \in S \setminus \{j_S\}} d_j - \min_{j \in S \setminus \{j_S\}} r_j,$$

which is valid by Proposition 1.

Now consider a point (x, y) with $x_{j_S}^m = 1$. If $\sum_{l \in S \setminus \{j_S\}} x_l^m = 0$ then (25) is satisfied because of condition (24). Otherwise there is some other job l completed on machine m in the interval $[r_i, d_j]$. As j_S must come before or after l, all the jobs in S must be scheduled inside segment $[r_{j_S}, d_{j_S} - \epsilon_S]$ or $[r_{j_S} + \delta_S, d_{j_S}]$. Thus

$$\sum_{l \in S} p_l^m x_l^m \le \max\{d_{j_S} - \epsilon_S - r_{j_S}, d_{j_S} - r_{j_S} - \delta_S\}$$

$$= d_{j_S} - r_{j_S} - \delta_S - \epsilon_S + \max\{\delta_S, \epsilon_S\} x_{j_S}^m.$$

$\square$

Note that at most n^3 constraints of this type have to be considered per machine - for each potential job j_S, there are at most $O(n^2)$ pairs of r_i and d_j such that $r_{j_S} < r_i \le r_j$ and $d_{j_S} > d_j \ge d_i$.

Finally we consider an alternative, second way to strengthen the inequality (23).

Consider again an interval $[r_i, d_j]$, and for each job l, let $\alpha_{li} = (r_i - r_l)^+$ and $\beta_{jl} = (d_l - d_j)^+$.

Proposition 3 *The inequality*

$$\sum_{l \in N} \min[d_j - r_i, (p_l^m - \max\{\alpha_{li}, \beta_{jl}\})^+] x_l^m \le d_j - r_i \tag{26}$$

is valid for X^{MM} for all $m \in K$.

11

Proof. For each job l, we show that at least the amount $\min[d_j - r_i, (p_l^m - \max\{\alpha_l^{ij}, \beta_l^{ij}\})^+]$ of its total processing time p_l must lie inside the interval $[r_i, d_j]$.

If $r_l \geq r_i$ or $\alpha_{li} = 0$, the maximum amount that can be processed after d_j is $(d_l - d_j)^+ = \max\{\alpha_{li}, \beta_{jl}\}$. Hence the minimum time within the interval is $(p_l^m - \max\{\alpha_{li}, \beta_{jl}\})^+$ which is less than or equal to $d_j - r_i$. The case when $d_l \leq d_j$ or $\beta_{jl} = 0$ is similar.

Otherwise the interval $[r_i, d_j]$ lies strictly within the interval $[r_l, d_l]$, or equivalently $\alpha_{li} > 0$ and $\beta_{jl} > 0$. Suppose that $\beta_{jl} \geq \alpha_{li}$. Then if $p_l^m - \beta_{jl} < d_j - r_i$, the maximum amount of processing time of l that can lie outside the interval is β_{jl}. Thus the minimum amount within the interval is $p_l^m - \beta_{jl} = p_l^m - \max\{\alpha_{li}, \beta_{jl}\}$. Alternatively if $p_l^m - \beta_{jl} \geq d_l - r_i$, job l occupies the whole interval and uses $d_j - r_i$ of the processing time.

$\qquad\square$

3.2 Single-machine case

For solving subproblems in the algorithms 5-7 we need single-machine variants of the problem formulations.

In the algorithm CG-MIP, we use the following formulation to solve the subproblem SP^m (9)-(10). All the variables are the same as in the multimachine formulation (11)-(22), except that the superscript on the variables x has been removed. Also the assignment constraint has gone as a job does not have to be processed, and (18) is replaced by (33).

$$\max \quad \sum_{j \in N} (-c_j^m - \pi_j) x_j - \mu_m \tag{27}$$

$$s.t. \quad \sum_{j \in N} p_j^m x_j \leq \max_{j \in N} d_j - \min_{j \in N} r_j \tag{28}$$

$$s_j + p_j^m x_j - e_j = 0 \quad \forall j \in N \tag{29}$$

$$e_j - s_j + U y_{ij} \leq U \quad \forall i, j \in N, i \neq j \tag{30}$$

$$r_j \leq s_j \quad \forall j \in N \tag{31}$$

$$d_j \geq e_j, \quad \forall j \in N \tag{32}$$

$$y_{ij} + y_{ji} - x_j \leq 0 \quad \forall i, j \in N, i \neq j \tag{33}$$

$$x_i + x_j - y_{ij} - y_{ji} \leq 1 \quad \forall i, j \in N, i < j \tag{34}$$

$$x_j \in \{0, 1\} \quad \forall j \in N \tag{35}$$

$$y_{ij} \in \{0, 1\} \quad \forall i, j \in N, i \neq j. \tag{36}$$

To tighten the formulation in the algorithms CG-MIP$^+$ and CG-MIP$^+$/CP, it is possible to use the valid inequalities (26) with the machine superscript on the processing times removed.

4 Implementation and Computational Results

4.1 Details of the Implementations

Here we describe briefly the details concerning our implementation of Algorithms 1-7.

Starting with the pure MIP algorithms, for Algorithm 1 we use the MIP formulation (11)-(22) described in Section 3.1 with $U = \max_j d_j - \max_i r_i$, and for Algorithm 2 we have tightened with the constraints (26) for all pairs $i, j \in N$ with $r_i < d_j$ and all $m \in K$.

For the MIP/CP algorithms, we used the assignment constraints (12),(21) in Algorithm 3, and we again tightened with the constraints (26) in Algorithm 4. Preliminary computational tests showed that constraints (26) were more effective than constraints (25), and that the run times increased when both sets of constraints were added. In Algorithm 3 the *disjunctive* constraint was called at all nodes of the tree as in the study of Bockmayr and Pisaruk, whereas in Algorithm 4 initial testing led us to only call *disjunctive* at nodes in which the linear programming solution was integer.

We now describe the main steps of our implementation of the branch-and-price algorithm used in Algorithms 5-7.

We start the branch-and-price algorithm by solving the linear programming relaxation of the Master Problem (MP) by Column Generation. If the solution is fractional, we branch on a variable with a fractional value in this solution. For 0-1 problems, it is standard to branch on the variables $x \in \{0, 1\}^{k \times n}$ of the initial (IP). The relation of the x variables of the (IP) formulation and λ variables of the (MP) formulation is the following:

$$x_j^m = \sum_{t \in T^m} a_{jt}^m \lambda_t^m. \tag{37}$$

When we branch, we set variable x_j^m to 1 or 0. This means that all partial schedules or columns for machine m either must contain or else must not contain job j. This restriction is easily added into the subproblems. For

the *branching variable selection*, we use a standard most fractional variable strategy, choosing to branch on the variable with value closest to 0.5.

Suppose that the variable x_j^m is chosen for branching at some node of the Branch-and-Price enumeration tree. Then two descendant nodes are created, one with the additional constraint $x_j^m = 0$ and the other with $x_j^m = 1$. The (RMP) at the successor nodes is then initialized by taking those columns of the parent node that satisfy the additional constraint.

After generating and solving (RMP) by linear programming at each node, we then solve (RMP) with its present set of columns as an IP. Typically this restricted IP can be solved fairly rapidly, and provides a good feasible solution, without going deep the search tree. In addition if the optimal value of this restricted IP is the integer round-up of the optimal value of (RMP), the node can be immediately pruned.

Finally the *node selection* strategy used is best bound.

Turning now to the solution of the subproblems (SP^m), we use the MIP formulation (27)-(36) in Algorithm 5, we add the constraints (26) in Algorithm 6. For Algorithm 7 we take the valid inequalities (26) plus the binary restrictions on the x variables (35) as the tight relaxation of (SP^m). The *disjunctive* global constraint is called at each node in which the LP solution is integer, and if necessary the "no-good" cut (4) is added as in Algorithms 3 and 4.

4.2   Numerical Results

Here we present numerical results for all seven algorithms, described in the section 2. All experiments have been carried out on a PC with a Pentium 4 2GHz processor and 512 Mb RAM. The algorithms were implemented in the MOSEL system [6] version 1.3.2, using XPress-MP version 14.21 as the MIP solver, and CHIP version 5.4.3 as the CP solver.

The first 9 instances of the MMASP problem are taken from the Bockmayr and Pisaruk paper [3]. The names of these instances are of the form "$m - n[\gamma]$", where m is the number of machines, n is the number of jobs and γ denotes a character to distinguish instances of the same size.

Test	Obj	Best LB	Time	Init. LB	Obj	Best LB	Time	XLP
			MIP				MIP$^+$	
3-12a	101	*	64.985	98.000	101	*	6.828	99.632
3-12b	104	*	61.125	99.177	104	*	0.532	104.000
5-15a	115	*	388.563	113.000	115	*	16.562	114.352
5-15b	129	*	277.265	121.000	129	*	3.188	128.887
5-20a	159[1]	156.000	> 1 hour	156.000	158	*	83.594	157.647
5-20b	143[1]	135.560	> 1 hour	135.134	139	*	1066.500	137.442
6-24	238[1]	224.000	> 1 hour	223.683	227	*	2405.500	226.322
7-30	-[2]	205.608	> 1 hour	205.468	-[2]	211.418	> 1 hour	210.812
8-34	-[2]	242.629	> 1 hour	242.582	-[2]	249.875	> 1 hour	249.858
Test			MIP/CP				MIP$^+$/CP	
3-12a	101	*	0.282	97.869	101	*	0.407	99.883
3-12b	104	*	2.344	98.417	104	*	0.328	104.000
5-15a	115	*	0.438	112.650	115	*	0.563	114.470
5-15b	129	*	4.313	121.103	129	*	0.563	129.000
5-20a	158	*	6.110	154.910	158	*	1.313	157.672
5-20b	139	*	287.703	134.289	139	*	2.703	137.548
6-24	227[1]	225.130	> 1 hour	222.688	227	*	3.672	226.286
7-30	215[1]	207.650	> 1 hour	203.629	213	*	16.656	211.037
8-34	253[1]	243.583	> 1 hour	241.613	252	*	3916.420 [3]	250.096

Table 1: MIP and MIP/CP algorithms: lower bounds and solution time

Test	Obj	Best LB	Time	Obj	Time	Obj	Time	LP
		CG-MIP			CG-MIP$^+$		CG-MIP$^+$/CP	
3-12a	101	*	36.938	101	8.297	101	2.079	100.500
3-12b	104	*	51.656	104	17.531	104	2.500	104.000
5-15a	115	*	32.922	115	20.172	115	3.813	114.500
5-15b	129	*	52.703	129	17.109	129	4.437	129.000
5-20a	158	*	245.485	158	61.781	158	12.141	157.818
5-20b	139	*	475.485	139	83.437	139	16.219	138.500
6-24	232[1]	226.545	> 1 hour	227	794.500	227	36.718	226.545
7-30	-[4]	-	> 1 hour	213	444.015	213	43.093	212.200
8-34	-[4]	-	> 1 hour	252	2857.219	252	561.219	251.333

Table 2: Column generation algorithms : lower bounds and solving time

---

[1] The best found solution after 1 hour (optimality is not proven)

[2] No solution was found after 1 hour

[3] This instance is solved in just over an hour

[4] The linear programming Master Problem is not solved within 1 hour

15

Results for these instances with Algorithms 1-4 are shown in Table 1, and with Algorithms 5-7 in Table 2. In Table 1 the first column indicates the instance. Then there are four columns per algorithm: the first contains the value of the best feasible solution found, the second the best lower bound (with a $*$ if upper and lower bounds are equal), the third contains the time in seconds to prove optimality (or to the cutoff time of 1 hour), and the fourth column gives the value XLP of the initial linear programming solution LP after the addition of system cuts.

As expected, the scheduling constraints $Cx + Dy \leq d$ do not improve the LP bound at all, and thus it turns that the LP values (which are not reported in Table 1) satisfy the following

$$LP_{MIP/CP} = LP_{MIP} \leq LP_{MIP^+}$$

and that

$$LP_{MIP/CP} \leq LP_{MIP^+/CP} = LP_{MIP^+}.$$

Note that the XLP values behave similarly as they are typically less than 1 more than the LP values.

The best earlier results that we know of are those of Bockmayr and Pisaruk [3]. Algorithm 3 is essentially the algorithm that they proposed. Our results for the formulation MIP/CP resemble theirs in that they managed to solve the first six instances, but could not solve the last three within one hour.

In the column generation results in Table 2, the first column gives the instance, and the last column the LP bound obtained from the Master Problem LP relaxation. Between the two, there are three columns for Algorithm 5 (CG-MIP) giving the value of the best solution found, the value of the best lower bound, and the run time, and for Algorithms 6 and 7 there are two columns with the best solution value and the run time. In Algorithm 7 initial testing indicated that it was best to add only a subset of the constraints (26) in the tightened subproblem. Specifically the constraint is only added for pairs $[r_i, d_j]$ such that $i \neq j, d_i \leq d_j$ and $r_i \leq r_j$.

We observe that the LP bound from column generation (Table 2) is always better than the LP bounds from the direct MIP formulations (Table 1). As expected these LP bounds are very tight. In addition, solving the restricted Master at each node produces very good integer solutions quickly, as seen by the very small number of tree nodes. Algorithms 4 and 7 with both the

strengthened MIP formulation and the CP feasibility test clearly dominate the others.

To further compare these two Algorithms, we then generated some larger instances. For each instance with m machines and $n = \eta m$ jobs first some parameters are defined randomly: $bmc_m \in [6, 12]$ (base machine cost), $bmt_m \in [bmc_m - 2, bmc_m + 2]$ (base machine time), $m \in K$, $bjc_j \in [6, 12]$ (base job cost), $bjt_j \in [bjc_j - 2, bjc_j + 2]$ (base job time), $j \in N$. Costs and processing times of jobs are distributed uniformly in following intervals: $c_j^m \in [round(bmc_m/2 + bjc_j/2) - 3, round(bmc_m/2 + bjc_j/2) + 3]$, $p_j^m \in [round(bmt_m/2 + bjt_j/2) - 3, round(bmt_m/2 + bjt_j/2) + 3]$, $m \in K$, $j \in N$. Then release dates and deadlines are generated with uniform distribution in the following segments: $r_j \in [0, 10]$, $d_j' \in [\beta - 10, \beta + 10]$, $d_j = \max\{d_j', r_j + \max_{m \in K}\{p_j^m\}\}$, $j \in N$, where $\beta = \sum_{m \in K, l \in N}(p_l^m)\frac{\theta}{m^2}$. θ here is the "freedom" parameter, the less is θ the tighter are deadlines.

We generated 5 instances for each triple of parameters: (m, η, θ), where $m \in \{7, 8, 9\}$, $\eta \in \{3, 4, 5, 6\}$, $\theta \in \{0.5, 0.6, 0.8, 1\}$. In the table 3 we present results only for those instances, which have at least one feasible solution (all the jobs can be scheduled inside their time windows) and for which the solution by the MIP$^+$/CP algorithm took more than 200 seconds. Names of all the instances from the second group have the form "$m - n - \theta - \kappa$", where κ is a number used to distinguish between instances with the same triple of parameters.

In Table 3 the first column indicates the instance, and then for Algorithms 4 and 7 we have five columns giving "Obj" the value of the best feasible solution found, "Time" the time till optimality was proved, "Nodes" the number of nodes in the enumeration tree, Cuts" the number of nogood cuts added during the algorithm, and finally "LP bound" the value of the linear program at the top node. Details are given for the 9 instances appearing in Tables 1 and 2, as well as the 27 newly generated instances. For both the branch-and-cut and branch-and-price algorithms we used the subset of the cuts (26) described above.

Test	Branch&Cut MIP⁺/CP					Branch&Price CG-MIP⁺/CP					
	Obj	Time	Nodes	Cuts	LP bound	Obj	Time	Nodes	Iter	Cuts	LP bound
3-12a	101	**0.407**	61	0	99.883	101	2.079	1	26	62	100.500
3-12b	104	**0.328**	1	0	104.000	104	2.500	1	22	29	104.000
5-15a	115	**0.563**	62	0	114.470	115	3.813	1	24	84	114.500
5-15b	129	**0.563**	1	0	129.000	129	4.437	2	29	153	129.000
5-20a	158	**1.313**	77	0	157.672	158	12.141	1	41	106	157.818
5-20b	139	**2.703**	392	8	137.548	139	16.219	1	36	718	138.500
6-24	227	**3.672**	416	0	226.286	227	36.718	2	44	1278	226.545
7-30	213	**16.656**	516	10	211.037	213	43.093	1	43	167	212.200
8-34	252	3916.420	194655	0	250.096	252	**561.219**	12	80	538	251.333
7-28-0.6-2	220	285.016	21299	4850	208.336	220	**86.469**	3	48	1910	219.222
7-35-0.6-1	270	**311.578**	8291	33	261.502	270	952.484	19	171	5437	266.333
7-35-0.6-2	236	**248.406**	10540	114	231.919	236	261.703	6	71	965	234.440
7-35-0.6-5	306	1169.344	40095	72	295.895	306	**726.438**	9	104	4028	302.106
7-42-0.6-4	312	**239.203**	4481	18	306.281	312	637.734	1	76	2819	312.000
8-32-0.6-3	278	213.704	6375	69	268.321	278	**154.391**	3	50	894	276.286
8-32-0.6-4	277	**230.250**	6166	2	268.236	277	299.985	13	85	420	273.375
8-32-0.6-5	243	508.750	22034	62	236.002	243	**381.953**	15	97	1584	240.412
8-40-0.6-1	282	735.547	16461	433	278.516	282	**406.281**	1	55	5066	282.000
8-40-0.6-2	344⁵ *342.848*	> 1 hour	79584	632	332.438	344	**588.625**	3	66	3295	343.400
8-40-0.6-3	288⁵ *286.595*	> 1 hour	76953	687	280.769	288	**1252.953**	15	131	5325	286.312
8-40-0.8-1	271	**469.000**	10148	64	268.390	271	493.422	7	92	918	270.389
8-48-0.6-1	383⁵ *359.812*	> 1 hour	35824	25	356.257	**364⁵** *362.806*	> 1 hour	20	194	8467	361.158
8-48-0.6-4	391	**1659.157**	19303	3	385.953	391	2541.422	10	157	918	389.857
8-48-0.6-5	441	**229.407**	2358	4	432.457	441	1338.265	3	99	1695	438.667
8-48-0.8-3	322	**336.875**	5076	0	320.440	322	1875.000	10	139	149	321.449
9-36-0.6-4	312	1387.156	19160	23	299.304	312	**922.907**	24	147	2498	306.097
9-36-0.6-5	248	444.031	13301	373	242.785	248	**318.453**	6	67	814	247.000

Test	Branch&Cut MIP$^+$/CP					Branch&Price CG-MIP$^+$/CP					
	Obj	Time	Nodes	Cuts	LP bound	Obj	Time	Nodes	Iter	Cuts	LP bound
9-36-0.8-1	228	**228.484**	6945	193	224.029	228	405.562	12	96	798	226.625
9-45-0.6-2	348[5]	> 1 hour	32162	44	328.094	**339**	**1854.875**	9	103	8445	336.710
	335.620										
9-54-0.5-3	-[6]	> 1 hour	21764	5	480.672	**504**	**3372.219**	8	128	3112	496.353
	495.448										
9-54-0.6-1	**435**	**2052.703**	17367	0	430.268	444[5]	> 1 hour	10	131	124	432.876
						434.183					
9-54-0.6-2	452	**1825.203**	12656	8	446.931	452	1850.985	1	84	1149	452.000
9-54-0.6-3	-[6]	> 1 hour	20502	0	411.672	**425**[5]	> 1 hour	9	118	4970	420.123
	≈ 417					*422.667*					
9-54-0.6-4	-[6]	> 1 hour	18387	0	435.313	-[6]	> 1 hour	7	121	2591	440.840
	440.908					*441.759*					
9-54-0.6-5	**437**	**2078.844**	15746	10	431.898	439[5]	> 1 hour	8	125	3533	435.857
						436.364					
9-54-0.8-4	**405**	**2154.313**	24926	6	403.084	406[5]	> 1 hour	10	148	157	403.684
						404.168					

Table 3: The algorithms MIP$^+$/CP and CG-MIP$^+$/CP : further comparison

---

[5]The best found solution after 1 hour (optimality is not proven), the best known lower bound see just below
[6]No solution was found after 1 hour, the best known lower bound see just below

For the 27 new instances with 7-9 machines and 28-54 jobs, we observe that all but three are solved by at least one of the two algorithms within 1 hour. 7 were not solved within 1 hour by the Branch-and-Cut algorithm, and 6 not solved by Branch-and-Price. However somewhat surprisingly, 13 are solved faster by Algorithm 4 and 11 by Algorithm 7. For this test set it appears that for the larger instances the Branch&Price algorithm CG-MIP$^+$/CP is better than the Branch-and-Cut algorithm MIP$^+$/CP when the ratio $\frac{n}{m}$ is lower. It should however be emphasized that both implementations are completely written in Mosel, and neither has been optimized in any way.

A further test was carried out using all the inequalities (26). The Branch-and-Price algorithm (Algorithm 7) was always worse. On the other hand using Branch-and-Cut (Algorithm 4), only 18 instances were unsolved after 200 seconds. Again 7 were unsolved after 1 hour. In this case the linear programming times are always increased and the number of nodes always decreased whether the instance is solved or not.

5 Conclusions and future work

In this paper we have presented seven different algorithms for solving MMASP. Two of them appear to dominate. The first, the algorithm MIP$^+$/CP has two features distinguishing it from the algorithm of Bockmayr and Pisaruk: a tighter IP formulation is used, and the constraint generation feasibility check is only performed at nodes having integral solutions of the LP relaxation.

The second algorithm, the algorithm CG-MIP$^+$/CP successfully exploits the structure of the problem and generates very good lower bounds. This fact along with the possibility of generating feasible solutions at each node of the Branch-and-Price search tree allows one to find an optimal solutions quickly. However a drawback of this approach time is the time required to solve the subproblems. Here it would be interesting to try the algorithms by Baptiste et al. [1] and Peridy et al. [10] for the $1 \mid r_j \mid \sum w_j U_j$ problem. The former is a Constraint Programming algorithm, and the latter is based on Lagrangian relaxation and restricted shortest paths.

In the problems instances we have tackled, the single machine subproblems involve selecting an average of at most 6 jobs from 40 or 50 jobs. As this average number of jobs increases, the subproblems become harder, and the number of no-good cuts required in Algorithms 4 and 7 increases significantly. This suggest at least two problems that need to be tackled if we wish

to make further progress. We need very efficient algorithms for even hard 50 job instances of the single machine subproblem $1|r_j|\sum_j w_j U_j$, and we need to find ways to strengthen the no-good cuts so that less cuts need to be added.

Finally it would be interesting to try to apply a similar approach to solve other parallel machine scheduling problems with similar structure and/or with additional constraints.

References

[1] Baptiste, Ph., A. Jouglet, C. Le Pape, W. Nuijten. 2000. *A Constraint-Based Approach to Minimize the Weighted Number of Late Jobs on Parallel Machines.* Research Report UTC, 2000/288.

[2] Baptiste, Ph., C. Le Pape, W. Nuijten. 2001. *Constraint-based scheduling: applying constraint programming to scheduling problems.* Kluwer Academic Publishers.

[3] Bockmayr, A., N. Pisaruk. 2003. MIP/CP cooperation in MOSEL. LISCOS project report.

[4] Carlier, J., 1982. The one machine sequencing problem. European J. of Operational Research **11**, 42-47.

[5] Chen, Z-L., W.B. Powell. 1999. Solving parallel machine scheduling problems by column generation. INFORMS Journal on Computing, **11** 78-94.

[6] Colombani, Y., T. Heipcke. 2002. Mosel: an extensible environment for modeling and programming solutions. *4th International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatirial Optimization Problems, CP-AI-OR'02*, Le Croisic, France, 277-290.

[7] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness, Freeman.

[8] Jain, V., I.E. Grossman. 2001. Algorithms for hybrid MILP/CLP models for a class of optimization problems. INFORMS Journal on Computing, **13** 258-276.

[9] Juncker, U., S. Karish, N. Kohl, B. Vaaben, T. Fahle, M. Sellman. 1999. A framework for constraint programming based column generation. *Fifth International Conference on the Principles and Pactice of Constraint Programming (CP'99)*, Alexandria, VA.

[10] Peridy, L., E. Pinson and D. Rivreau, 2003. Using short-term memory to minimize the weighted number of late jobs on a single machine, European J. of Operational Research **148**, 591-603.

[11] van Hentenryck, P., 2002. Constraint and integer programming in OPL. Departnment, Brown University.