A hybrid solution strategy for production planning

João Pedro Pedroso

Technical Report Series: DCC-2007-9

A hybrid solution strategy for production planning

João Pedro Pedroso
Dep. Ciência Computadores, Fac. Ciências, Universidade do Porto
Rua do Campo Alegre 1021-1055, 4169-007 Porto, Portugal
jpp@fc.up.pt

May 2007

Abstract

Production planning with lot-sizing and scheduling in industries where the cleaning times depend on the sequence of the products can be modelled as a mixed integer problem. However, this problem is very hard, and even small instances cannot be solved exactly in a reasonable time. We propose a division of the whole production planning problem into its two subproblems, and to solve each them independently: first the lot-sizing, then schedule the operations that were determined for each machine, for each period. In this strategy, the lot-sizing part does not take into consideration the changeover times. After scheduling the operations, if the cleaning times make a solution infeasible, a constraint cutting this solution must be added. The lot-sizing problem must be resolved, and its solution rescheduled, until obtaining feasibility. We propose an algorithm for integrating lot-sizing and scheduling, and analyse the results obtained.

1 Introduction

The problem that we deal with in this paper concerns industries with a number of machines, where several products can be manufactured in each machine. The objective is to minimise costs, which include setup costs, inventory, and backlog costs; besides, as machine cleaning depends on the sequence of production, changeover costs must also be considered.

Each machine has a limit on its operating time, which is used for setup and production. The time required for cleaning and setting up a machine depends on its previous state. These changeover times make the scheduling an important decision for maximising machine usage.

We present a MIP model for the whole problem, which includes lot-sizing and scheduling (for related models see, for instance, [1, 7]). This is a small bucket model, i.e., it allows one setup per period, and computes changeover costs and times when the setups is successive productions concern different items. It is not practical, though; scheduling is rather difficult to solve as a mixed-integer problem. Even for small instances of this complete model, the best solution found by state-of-the-art MIP solvers in a reasonable time has large gaps with respect to the lower bounds.

If the lot-sizing part is formulated as a big bucket model, i.e., it allows more than one setup per period as long as the machine capacities are respected, it does not include scheduling decisions anymore. On the other hand, its solution is not nearly as difficult, and can be determined by MIP solvers in a much shorter time.

For including both the lot-sizing and the scheduling decisions in the same solution process, an alternative is to use lot-sizing and scheduling in separated modules.

If a big-bucket lot-sizing model determines the setups that should be done in each machine for each period, determining the optimal order of the productions for a given period in this machine corresponds to minimising the changeover costs and/or times. This problem is equivalent to determining a shortest Hamiltonian path, in an asymmetric graph; even though this is also an NP-complete problem, there is a vast set of algorithms for tackling it, exactly or approximately, even for large size instances. Notice that this scheduling strategy does not discretise time, and hence cannot be precisely described as a MIP.

After solving the subproblem of scheduling the operations, one must check if the makespan of every machine, for all periods, is less than the available working time: as the MIP for the lot-sizing problem did not take into consideration the changeover times, the scheduling solution may exceed the available production time. If this occurs, additional constraints must incorporated into the MIP, for preventing the same solution (i.e., the same setups and the same quantities) of being obtained again on the concerned machines.

The lot-sizing problem must then be resolved, and its solution rescheduled; this is repeated until obtaining feasibility. We analyse the shape of the constraints, and how to iteratively use the lot-sizing and the scheduling modules. The aim is to obtain a feasible, good quality solution in a reasonable time.

2 A MIP model for lot-sizing

2.1 Variables

The models for production planning allow the managers to determine what quantity should be produced in each period, and in which machine. For a given item, the quantity produced and the demand determine the quantity that remains in inventory, or the quantity that could not be supplied and is backlogged.

Therefore, the decision variables concern the manufacture or not of a product in each period, as well as the amount to produce. Let y be the setup, binary variables: y^i_{pt} is 1 if item i is manufactured in the machine/production centre p during period t, and 0 otherwise. The continuous variable x^i_{pt} is the corresponding manufactured amount. The quantity that is held in inventory at the end of the period t is h^i_t. If the demand for this item could not be satisfied in this period, there is a quantity backlogged, represented by g^i_t. All these continuous variables must be non-negative.

2.2 The objective

The costs that are to be taken into account are setup, inventory, and backlog costs.

Let T be the number of periods and $\mathcal{T} = \{1, \ldots, T\}$. Let $\mathcal{I}$ be the set of products and $\mathcal{P}$ be the set of production centres, or machines. Let furthermore $\mathcal{P}^i$ be the subset of machines that are compatible with the production of i. The setup costs are then determined by:

$$\sum_{i \in \mathcal{I}} \sum_{p \in \mathcal{P}^i} \sum_{t \in \mathcal{T}} f^i_p \, y^i_{pt},$$

where f^i_p is the cost of setting up machine p for producing i. If h^i_t is the amount of product i that is kept in inventory at the end of period t, the holding costs can be determined by

$$\sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} r^i \, h^i_t,$$

where r^i is the unit inventory cost for product i. Analogously, if g^i_t is the amount of product i that failed to meet demand at the end of period t, the backlog costs can be determined by

$$\sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} b^i \, g^i_t,$$

where b^i is the unit backlog cost for product i. The lot-sizing objective can thus be written as

$$\text{minimise}\, z = \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \left[r^i \, h^i_t + b^i \, g^i_t + \sum_{p \in \mathcal{P}} f^i_p \, y^i_{pt} \right]. \tag{1}$$

2.3 Constraints

If the demand of a product i in period t is D^i_t, the flow conservation constraints can be written as

$$h^i_{t-1} - g^i_{t-1} + \sum_{p \in \mathcal{P}^i} x^i_{pt} = D^i_t + h^i_t - g^i_t \quad \forall\, i \in \mathcal{I}, \quad t \in \mathcal{T}. \tag{2}$$

The initial inventory and backlog for each product i should be assigned to h^i_0 and g^i_0, respectively (and possibly equivalent assignments might be made for h^i_T and g^i_T).

There is a limit on the time that each machine is available on each period; this implies that

$$\sum_{i \in \mathcal{I}: p \in \mathcal{P}^i} \left(\frac{x^i_{pt}}{\gamma^i_p} + \tau^i_{pt}\, y^i_{pt} \right) \le A_{pt} \quad \forall\, p \in \mathcal{P}, \quad t \in \mathcal{T}. \tag{3}$$

In this equation, γ^i_p is the rate of production of product i on machine p per time unit, τ^i_{pt} is the setup time required if there is production of i on machine p during period t, and A_{pt} is the number of time units available for production on machine p during period t.

Manufacturing of a given product can only occur on machines which have been setup for that product:

$$x^i_{pt} \le \gamma^i_p\, A_{pt}\, y^i_{pt} \quad \forall\, i \in \mathcal{I}, \quad p \in \mathcal{P}^i, \quad t \in \mathcal{T}. \tag{4}$$

This completes a MIP model for the big-bucket lot-sizing problem, which we will call *Model 1*.

3 Modeling changeovers in a MIP

If changeovers are to be taken into account on the MIP, the production periods should be subdivided, in order to allow a finer control on the scheduling details. Let us call the number of subperiods into which each period is divided S, and let $\mathcal{S} = \{1, \ldots, S\}$. The total number of subperiods is then $S\,T$; in the remainder of this section, we will use t' as an index for the subperiods:

$$t' = S\,(t-1) + s, \quad \forall t \in \mathcal{T},\ s \in \mathcal{S}.$$

In a machine p, a changeover from product i into product j occurs in period t' when $y^i_{p,t'-1} = 1$ and $y^j_{pt'} = 1$. In this case, the changeover variable, $w^{ij}_{pt'}$ must be set to one; this is assured by:

$$w^{ij}_{pt'} \ge y^i_{p,t'-1} + y^j_{pt'} - 1 \quad \forall\, i, j \in \mathcal{I}, \quad p \in \mathcal{P}^i, \quad t' \in \mathcal{T} \times \mathcal{S}. \tag{5}$$

As this is a small-bucket model, at most one setup can occur in each subperiod:

$$\sum_{i \in \mathcal{I}} y^i_{pt'} \le 1 \quad \forall\, p \in \mathcal{P}, \quad t' \in \mathcal{T} \times \mathcal{S}. \tag{6}$$

The constraints of section 2.3 must be adapted to include all the productions on a subperiod. If we denote the cost of a changeover from i to j by e^{ij}_p and the corresponding time by δ^{ij}_p, the complete model can be summarised as the following MIP, which we will call *Model 2*:

$$\text{minimise}\, z = \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \left[r^i\, h^i_t + b^i\, g^i_t + \sum_{p \in \mathcal{P}} \sum_{s \in \mathcal{S}} \left(f^i_p\, y^i_{pt'} + \sum_{j \neq i \in \mathcal{I}} e^{ij}_p\, w^{ij}_{pt'} \right) \right]$$

subject to:

$$h^i_{t-1} - g^i_{t-1} + \sum_{p \in \mathcal{P}^i} \sum_{s \in \mathcal{S}} x^i_{pt'} = D^i_t + h^i_t - g^i_t, \quad \forall\, i \in \mathcal{I}, \quad t \in \mathcal{T}$$

$$\sum_{i \in \mathcal{I}: p \in \mathcal{P}^i} \sum_{s \in \mathcal{S}} \left(\frac{x^i_{pt'}}{\gamma^i_p} + \tau^i_{pt}\, y^i_{pt'} + \sum_{j \neq i \in \mathcal{I}} \delta^{ij}_p\, w^{ij}_{pt'} \right) \le A_{pt}, \quad \forall\, p \in \mathcal{P}, \quad t \in \mathcal{T}$$

$$x^i_{pt'} \leq \gamma^i_p \, A_{pt} \, y^i_{pt'} \quad \forall \, i \in \mathcal{I}, \quad p \in \mathcal{P}^i, \quad t \in \mathcal{T}, \quad s \in \mathcal{S}$$

$$h^i_t, \; g^i_t \in \mathbb{R}^+, \quad \forall \, i \in \mathcal{I}, \quad t \in \mathcal{T}$$

$$x^i_{pt} \in \mathbb{R}^+, \; y^i_{pt} \in \{0,1\}, \quad \forall \, i \in \mathcal{I}, \; p \in \mathcal{P}, \quad t \in \mathcal{T}. \tag{7}$$

As will be seen in the computational results (Section 6), this problem is very hard, and no useful results can be obtained on a reasonable time, even for small instances.

4 Scheduling as a separate subproblem

As a heuristic for obtaining feasible solutions in a short time, we propose to keep lot sizing and scheduling as separate procedures, and integrate them for producing a solution to the complete problem. In this context, we will use *Model 1* for determining the quantities of each item to be produced in each machine, for each of the periods. After this, there remains to be determined the order of these operations inside each machine. For this scheduling problem, the first goal is to reduce the machine makespan up to the point where the schedule is feasible; after this, a second goal is to minimise costs, subject to keeping makespan feasibility. For the sake of simplicity, let us assume that the only goal is to minimise makespan.

Let us reserve the symbol 0 for no production. We denote by n_{pt} the last item produced in machine p on period $t-1$ (for $t=0$, this is a datum). If the machine was not occupied at the end of $t-1$, then $n_{pt}=0$. For a machine p, period t, the graph consists of nodes $N_{pt} = \{n^0_{pt}\} \cup \{i \in \mathcal{I} : \bar{y}^i_{pt} = 1\}$, where $\bar{y}$ is the solution of *Model 1*.

This scheduling subproblem is a variant of the of the shortest Hamiltonian path, in a possibly asymmetric graph. The path must start from the last item produced in that period, n^0_{pt}, and may finish on any node. The distances from node i to j are the changeover times δ^{ij}_p if $i \neq 0$, 0 otherwise. We denote by $\bar{\delta}_{pt}$ the length of the optimal path (i.e., the minimum makespan), given the current solution $\bar{y}$.

After the minimal path is found, the corresponding minimal makespan is determined by adding, to each node i of the path, the duration of the operation in the machine, $\bar{x}^i_{pt}/\gamma^i_p + \tau^i_{pt}$.

There is a number of algorithms that can be used for this scheduling problem. As the number of nodes is typically small, the exact solution is a possibility. In our implementation we used a greedy, nearest neighbour construction followed by local search based on arc-exchange [2]. For real-world problems, the scheduling part is likely to be a more complex problem, and more elaborate strategies, as those proposed on [4, 5], might be necessary for its solution.

4.1 Additional constraints

After solving the subproblem of scheduling the operations, one must check if the makespan is less than the available working time, for every machine and period. Indeed, as the MIP for the lot-sizing problem did not take into consideration the changeover times, the scheduling solution may exceed the available production time. If this occurs, additional constraints must incorporated into the MIP, for preventing the same solution (i.e., the same setups and the same quantities) of being obtained again on the concerned machines.

One possibility for preventing an infeasible scheduling solution on the lot-sizing problem is to tentatively reduce the available time on the machines whose makespan was too large (see e.g. [3, 6]), for example by the amount that was spent on the changeovers as determined by the scheduling process. However, this leads to suboptimal solutions, as different combinations of products could potentially be produced on that machine with shorter changeover times.

Another possibility is to reduce the available time on those machines, but only for the setups of the current solution. Practically, this is done by creating a new variable a_{pt} in *Model 1*, representing the available time on machine p, period t. The following constraints are added initially to *Model 1*:

$$a_{pt} \leq A_{pt}, \quad \forall \, p \in \mathcal{P}, \quad t \in \mathcal{T},$$

and A in Equation 3 is replaced by a.

Let us now consider the constraints that cut infeasible schedules from the lot-sizing problem. As an illustration, let us suppose that a machine p had setups $\bar{y}^i_{pt} = 1$ and $\bar{y}^j_{pt} = 1$, for a given period t, and let $\bar{\delta}_{pt}$ be the changeover times on the optimal schedule for this machine. The following constraint cuts this solution from the feasible region:

$$a_{pt} \leq (A_{pt} - \bar{\delta}_{pt})\,(3 - y^i_{pt} - y^j_{pt})$$

More generally, if $\mathrm{card}(N_{pt})$ denotes the cardinality of the set of operations in machine p (including the node representing the previous setup), the constraint can be written as:

$$a_{pt} \leq (A_{pt} - \bar{\delta}_{pt}) \left[\mathrm{card}(N_{pt}) - \sum_{i \in N_{pt} \setminus \{n_{pt}\}} y^i_{pt} \right] \tag{8}$$

Lemma 1 *If there are no changeovers on the machine p at the begin of period t, and the sum of the changeovers on the optimal scheduling solution for t is $\bar{\delta}_{pt} \leq A_{pm}/2$, the constraints of Equation 8 are valid inequalities.*

Proof: as long as the schedule is optimal, any quantities produced for this set of items in this machine must spend at least $\bar{\delta}_{pt}$ time for changeovers. If one of the products of the set is not manufactured, the right-hand side of Equation 8 is $2\ (A_{pt} - \bar{\delta}_{pt})$; this is larger than A_{pt} for all $\bar{\delta}_{pt} \leq A_{pm}/2$. Therefore, on this condition the constraint does cut feasible solutions.

Lemma 2 *If there are changeovers on the machine at the begin of the period, the constraints of Equation 8 are* not *valid inequalities.*

Proof: a feasible solution with a different setup for the machine at the previous period, such that the first setup is smaller, is cut by the constraint of Equation 8.

As we have seen, whenever the scheduling solution is not feasible, a constraint must be added to *Model 1*, and the the lot-sizing problem must then be resolved. Then, its solution must be rescheduled; this is repeated until obtaining feasibility, i.e., until all the machines have a schedule that does not exceed the maximum makespan.

5 Iterative solution procedure

The global solution procedure iteratively solves the lot-sizing and the scheduling models, until reaching a feasible solution, as depicted in Algorithm 1. The algorithm starts by solving the *Model 1* lot-sizing, initially with no additional constraints. Given the solution of this problem, it determines the set of operations that should be produced in each of the machines. With this, the optimal schedule for each machine is determined. If the schedules are feasible (i.e., they do not exceed the allowed production time) for all machines, then the current solution is feasible, and it is returned. Otherwise, a cutting constraint is generated using Equation 8 and included in the next lot-sizing instance.

The most intricate aspect of this algorithm concerns its step 8: removing redundant constraints. If this step is not included, after a certain number of iterations too many invalid inequalities are added, resulting in a solution that is generally far from the optimal. Thus, removal of invalid inequalities is essential for the efficacy of the algorithm. The method used for this is the following:

- After scheduling a machine, verify if its makespan is less that the maximum allowed.
- If not, then no constraints are removed for this machine and period.
- Otherwise, check if there were constraints added for this machine and period. For each of them, obtain the set of items being produced and recalculate the time necessary for changeovers.
- If this time is zero, then remove the constraint; if it is smaller than the existing cut, then update its value; otherwise, leave the constraint as it is.

Algorithm 1: Main solution procedure: alternate lot-sizing and scheduling until reaching a feasible solution.

```
MAIN()
(1)  repeat
(2)      solve the Model 1 lot-sizing MIP
(3)      generate a scheduling instance based on the MIP solution
(4)      solve the scheduling problem
(5)      if scheduling instance is not feasible
(6)          generate constraints that exclude previous MIP solution
(7)          incorporate constraints in the MIP
(8)          possibly update/remove invalid constraints
(9)  until the scheduling solution is feasible
(10) return solution of lot-sizing and scheduling
```

Processing the constraint pool this way leads to improved solutions. There is a problem, however: for some instances, there can be cycles adding and removing constraints in successive iterations, and the algorithm would not stop. For preventing this, we keep, for each constraint, a list of the values that it had assumed in the past. If it has been changed too many times, then only changes that increase the size of the cut are accepted (for ensuring convergence). The number of times that a constraint (not incrementing the cut size) can be updated is a parameter of the algorithm. In our experiments, cycles were only observed for relatively large instances.

5.1 An example

For illustrating the solution process for the different models and algorithms, we selected a toy instance of the series `p4i6`, presented in Section 6. In this instance there are two production periods, and two machines, each able to produce two of three items. The capacity of production is 200 units in each period, but changeover times must be deducted.

5.1.1 Small-bucket models

We firstly show the solution using the small bucket *Model 2*, with a division of each period into two subperiods ($S = 2$). The solution is:

Machine	Period 1		Period 2	
	$s = 1$	$s = 2$	$s = 1$	$s = 2$
1	x_1=100	x_1=100	x_1=80	x_2=70
2	x_2=100	x_3=80	x_3=70	x_3=100

On this solution, inventory variables are 0 except for $h_1^1 = 75$. Backlogs are 0 except for $g_1^2 = 25, g_1^3 = 45, g_2^1 = 50$. The total cost is $z = 1201.875$, of which the setup cost is 0.8, the changeover cost is 1.0, the inventory cost is 0.075, and the backlog cost 1200.

We now show the solution using the same model, with a division of each period into ten subperiods ($S = 10$). The solution is now:

Machine	Period 1									
	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$	$s = 8$	$s = 9$	$s = 10$
1	x_1=20	x_1=20	x_1=20	x_1=20	x_1=20	x_1=20	—	x_2=20	x_2=20	x_2=20
2	x_3=20	x_3=20	x_3=20	x_3=20	x_3=20	x_3=20	—	x_2=20	x_2=20	x_2=20

Machine	Period 2									
	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$	$s = 6$	$s = 7$	$s = 8$	$s = 9$	$s = 10$
1	x_2=20	x_2=20	x_2=20	—	x_1=20	x_1=20	x_1=20	x_1=20	x_1=20	x_1=20
2	x_2=20	x_2=20	x_2=20	—	x_3=20	x_3=20	x_3=20	x_3=20	x_3=20	x_3=20

On this solution, all the inventory variables are all 0. Backlogs are 0 except for $g_1^1 = 5, g_1^2 = 5, g_1^3 = 5, g_2^1 = 10, g_2^2 = 10, g_2^2 = 10$. The total cost is $z = 453.6$, of which the setup cost is 2.6, the changeover and inventory costs are 0, and the backlog cost is 450.

As the main costs in these instances are backlog costs, the flexibility given by a finer division of the periods leads to a better usage of the machines, and hence a superior solution. Unfortunately, this subdivision is very costly in terms of the hardness of the problem, and cannot be afforded but for very small instances. As we will see, if scheduling is a separate procedure this flexibility is even greater.

5.1.2 Iterative lot-sizing and scheduling

Let us now see the solution obtained using the iterative procedure presented in Algorithm 1.

Iteration 1
The first solution of the lot-sizing *Model 1* and scheduling is:

Machine	Period 1		Period 2	
1	x_1=125	x_2=50	x_1=125	x_2=50
2	x_3=125	x_2=75	x_2=75	x_3=125

This solution is not feasible: the makespan on machine 2, period 1 exceeds the available time $A_{21} = 1$. The changeover this for this machine is $\bar{\delta} = 0.1$, and hence the following cut is added:

$$a_{21} \leq (1 - 0.1)\,(3 - y_{21}^2 - y_{21}^3)$$

Similarly, for machine 2 and period 2:

$$a_{22} \leq (1 - 0.1)\,(3 - y_{22}^2 - y_{22}^3)$$

Iteration 2
On iteration, 2 the lot-sizing and scheduling solution is:

Machine	Period 1		Period 2	
1	x_1=125	x_2=70	x_2=75	x_1=125
2	x_3=125	x_2=55	x_2=50	x_3=125

In this solution, it is machine 1, in periods 1 and 2, that is exceeding the available time. The constraints added are:

$$a_{11} \leq (1 - 0.1)\,(3 - y_{11}^1 - y_{11}^2)$$

$$a_{12} \leq (1 - 0.1)\,(3 - y_{12}^1 - y_{12}^2)$$

Iteration 3
The next lot-sizing and scheduling solution is:

Machine	Period 1		Period 2	
1	x_1=110	x_2=70	x_2=40	x_1=140
2	x_3=125	x_2=55	x_2=50	x_3=125

This solution is feasible; makespans are 1 for both the machines, in both periods. There is no inventory, and the non-zero backlogs are $g_1^1 = 15, g_2^2 = 30$. The total cost is 453.6, including a backlog cost of 450, a fixed, setup cost of 0.8, and changeover costs of 2.8.

This solution is equivalent to the obtained by *Model 2* with 10 periods, but its schedule is more flexible, as the time at which changeovers occur are not discretised.

	Instance prefixes	
Parameter	`p2i3`	`p4i6`
D_t^i	125	50
$\gamma_p^i \times S$	200	200
f_p^i	0.1	0.1
r^i	0.001	0.001
b^i	10.	10.
τ_{pt}^i	0	0
A_{pt}	1.	1.
δ_p^{ij}	0.1	0.1
e_p^{ij}	0.5	0.5

Table 1: Data for the two series of instances used for benchmarking (S is the number of subperiods in *Model 2*, or 1 for instances of *Model 1*).

6 Results

6.1 Computational Environment

The computer environment used in this experiment is the following: a machine with an AMD Athlon(tm) XP 2800+ at 2 GHz, with 512 KB of cache and 1GB of RAM, with the Linux Debian operating system.

The MIP solver used is the commercial *Xpress-MP Optimizer, Release 17.10.04*.

The scheduling algorithm and the iterative solution procedure of Algorithm 1 were implemented in the Python language. Profiling showed that the CPU time spent on the Python part is negligible, as virtually all the CPU was used on the solution of the lot-sizing MIP.

The instances used are artificial problems. In instances `p2i3` there are 2 machines producing 3 items, and in `p4i6` there are 4 machines producing 6 items. The machine-item compatibilities for instances `p2i3` are the following:

Machine 1	can produce items 1, 2
Machine 2	can produce items 2, 3

For instances `p4i6`, the machine-item compatibilities are:

Machine 1 and 2	can produce items 1, 2, 3, 4
Machine 3 and 4	can produce items 3, 4, 5, 6

The remaining data are presented in table 1.

The data for each of these instances will be used for several time horizons; for example we will call `p2i3t4` an instance with the data of `p2i3`, for a four-period ($T = 4$) planning, with the big-bucket model. Instance `p2i3t4s2` is the equivalent for a small-bucket model with two subperiods per period ($S = 2$). On these instances, the demand for each period is slightly smaller than the production capacity, but nevertheless backlog is necessary for most of them. As the costs of backlogging are much higher than the others, minimising costs reverts to minimising backlogs, for most of the cases.

Results obtained by the MIP solution of the small-bucket *Model 2* are presented in table 2.

Results obtained by iterative lot-sizing and scheduling are presented in table 3. For Algorithm 1, we limited the time for each MIP lot-sizing solution to 15 seconds, and accepted the heuristic solution it found in that time as an input for scheduling. For avoiding cycles, we limited to three the number times a cut is relaxed or removed, for any cuts concerning a given set of items, machine, and period; after that, only larger (possibly invalid) cuts are accepted.

The results show a clear superiority of the iterative lot-sizing and scheduling solution of Algorithm 1 with respect to that of the small-bucket MIP, both in terms of quality and time required for reaching it. There is a tendency for the quality of the solution to degrade for a larger number of planning periods. One explanation for this is that the limit imposed to the solution time of the

Instance name	Solution	Lower bound	CPU time (s)
`p2i3t4s2`	1954.745		<1
`p2i3t8s2`	4008.680		71
`p2i3t16s2`	7616.680	3151.921	3600
`p2i3t32s2`	16686.425	2507.764	3600
`p2i3t4s10`	1157.505	787.717	3600
`p2i3t8s10`	3769.275	158.347	3600
`p2i3t16s10`	13588.695	101.518	3600
`p2i3t32s10`	39426.175	116.700	3600
`p4i6t4s2`	1805.860	1582.742	3600
`p4i6t8s2`	1811.930	1291.175	3600
`p4i6t16s2`	1827.200	1009.716	3600
`p4i6t32s2`	2153.990	1015.081	3600
`p4i6t4s10`	307.330	6.120	3600
`p4i6t8s10`	1812.520	12.240	3600
`p4i6t16s10`	3825.780	24.480	3600
`p4i6t32s10`	13750.750	48.960	3600

Table 2: Results obtained by the MIP solver *Xpress-MP Optimizer* for the lot-sizing and scheduling with the small bucket *Model 2*, with CPU time limited to 3600 seconds.

Instance name	Solution found	CPU time (s)
`p2i3t4`	955.885	<1
`p2i3t8`	3712.265	3.5
`p2i3t16`	6022.32	43.
`p2i3t32`	19936.15	367.
`p4i6t4`	12.8	4.1
`p4i6t8`	22.36	303.
`p4i6t16`	36.01	729.
`p4i6t32`	796.87	1389.

Table 3: Results obtained by the hybrid lot-sizing + scheduling of Algorithm 1. The MIP solver used for the lot-sizing subproblem is *Xpress-MP Optimizer*, with CPU time limited to 15 seconds for each solution.

lot-sizing (15 seconds) is quite short the larger problems; actually, analysis of the log of the process has shown that in some cases the gaps for the MIP solution were close to 100%. As could be expected, there is a trade-off between the the time allowed and the solution quality.

It is important to notice that the quality of the MIP solver is essential for obtaining good results by Algorithm 1. Actually, for easy instances, with large capacity excesses, the MIP solver could often find the optimal solution for *Model 2*, probably due to the quality of its presolver and cut generator. We have made some tests with open source solvers, but no solution could be obtained in useful time, except for tiny instances.

7 Conclusion

Production planning with lot-sizing has recently become more common in many industries, partly due to the enormous progress that has been observed in MIP solvers. For situations where the cleaning times and/or costs are important, lot-sizing decisions should also take into account the sequence of operations in each machine. However, this makes the problem much more difficult, and even with state-of-the-art solvers the exact solution is possible only for very small instances.

The strategy described in this paper separates lot-sizing decisions from scheduling, for being able to solve the lot-sizing problem more efficiently. Scheduling of the operations in each machine is dealt with in a different module; if its solution is not feasible, constraints are derived for avoiding the same solution in the lot-sizing. This process is repeated until the solution for the whole problem is feasible.

As the constraints that are included in the lot-sizing are not always valid, we present a method for removing some of the invalid ones. Constraint removal is a sensitive process, as it is easy to enter cycles of addition/removal of the same set of inequalities. For avoiding cycling, we propose a method based on the analysis of the history of each constraint, which from a certain point only allows increases on the cut size. This guarantees that, from that point on, no cycle can occur, and hence the method converges to a feasible solution.

Typically, the solution of the method proposed for iterated lot-sizing and scheduling is of better quality, and obtained in a shorter time, than the time-limited solution provided by a MIP solver for an equivalent problem.

A better control of the pool of constraints, and a more elaborated decision concerning those that are added and removed, is one of the most interesting research directions. This could be exploited, for example, using concepts from tabu search as a framework for avoiding cycles.

References

[1] Production Planning by Mixed Integer Programming. *Yves Pochet and Laurence A. Wolsey.* Springer, 2006.

[2] S. Lin. Computer solutions to the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, 1965.

[3] LISCOS: Large Scale Integrated Supply Chain Optimisation Software. Internet repository, http://www.liscos.fc.ul.pt/, 2002.

[4] K. Nonobe and T. Ibaraki. Formulation and tabu search algorithm for the resource constrained project scheduling problem. In C.C. Ribeiro and P. Hanse, editors, *Essays and Surveys in Metaheuristics*, pages 557–588. Kluwer Academic Publishers, 2002.

[5] Ruslan Sadykov and Laurence A. Wolsey. Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates. *INFORMS Journal on Computing*, 18:209–217, 2006.

[6] Christian Timpe. Solving mixed planning & scheduling problems with mixed branch & bound and constraint programming. Report in the liscos european project, BASF-AG, GVC/S-C13, 2001.

[7] L. A. Wolsey. MIP modelling of changeovers in production planning and scheduling problems. *European Journal of Operational Research*, 99:154–165, 1997.