Discrete lot sizing and scheduling on parallel machines: description of a column generation approach

António J.S.T. Duarte[†], J.M.V. Valério de Carvalho[‡]

[†] *Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Bragança, Portugal*
E-mail: aduarte@ipb.pt

[‡] *Departamento de Produção e Sistemas, Universidade do Minho, Portugal*
E-mail: vc@dps.uminho.pt

Abstract

In this work, we study the discrete lot sizing and scheduling problem (DSLP) in identical parallel resources with (sequence-independent) setup costs and inventory holding costs. We propose a Dantzig-Wolfe decomposition of a known formulation and describe a branch-and-price and column generation procedure to solve the problem to optimality. Preliminary results show that the lower bounds provided by the reformulated model are stronger than the lower bounds provided by the linear programming (LP) relaxation of the original model.

Keywords: DLSP, lot sizing, scheduling, setup costs, column generation, branch-and-price.

1 Introduction

Since the introductory work of Wagner and Whitin [Wagner and Whitin, 1958] a great amount of research has been done on the discrete lot sizing and scheduling problem (DLSP). The original model has been extended from single-item to multiple-item and from single resource to multiple-resource configurations. Also, additional constraints and different cost structures have been studied. Other studies aim at proposing and/or strengthening compact mixed integer linear (MILP) formulations in order to solve larger and more complex instances. Examples of relevant research works on this problem are [van Hoesel et al., 1991], [van Hoesel and Kolen, 1994], [van Eijl and van Hoesel, 1997], [Gicquel et al., 2011] and [Gicquel et al., 2012].

Most of the published research for problems with parallel resources is devoted to heuristics.

In this work we propose a Dantzig-Wolfe decomposition to a common integer linear (ILP) formulation and a branch-and-price algorithm to solve the problem to optimality. For the single resource problem a similar column generation approach is presented in [Cattrysse et al., 1993].

For the parallel resource configurations the authors are not aware of similar approaches, although the used decomposition is very close the one used in [Manne, 1958] and in [Lasdon and Terjung, 1971]. However, on those works, the problem of finding the optimal integer solution was not addressed. Also, the problem does have some similarities with the capacitated lot sizing and scheduling problem for which there is also some published research involving column generation, such as [Degraeve and Jans, 2007] and [Caserta and Voß¸, 2013].

A relatively recent review of methods for this problem can be found in [Karimi et al., 2003].

In section 2 we provide a formal description of the problem. In section 3 we present a compact original ILP formulation. In section 4 we present a minimum cost flow model that can be used to readily compute upper bounds. In section 5 a Dantzig-Wolfe decomposition for the ILP formulation is proposed along with the resulting master problem and subproblem. In section 6 a dynamic programming approach to the resulting subproblem is presented. Three different branching schemes to solve the problem to optimality are presented in section 7. Finally we present some results showing that the lower bounds provided by the reformulated model are stronger than the lower bounds provided by the linear programming relaxation of the original model.

2 Problem description

There are R identical parallel resources, indexed with $r = 1, \ldots, R$, I items to be processed, indexed with $i = 1, \ldots, I$, and T discrete and equal periods of time, indexed with $t = 1, \ldots, T$. In each time period, any given machine will be producing one *demand unit* of a given item or will be idle.

Without loss of generality, we define the *demand unit* for a given item as the quantity of that item that is possible to process in one machine during one time period. In practice, this can be seen as a minimum lot size for each item. From this point on, demands will be expressed in integer demand units.

Each item has the following associated coefficients: a vector of demands along the planning horizon, $d_i = \{d_{i1}, \ldots, d_{iT}\}$; a startup cost, s_i, which is the cost of starting the production of a different item in a given resource, which is resource and time independent; an inventory holding cost, h_i, defined as the cost of holding one demand unit of item i over one time period (time independent).

The objective is to decide a production schedule (assigning machines to items over the different time periods) that minimizes the sum of startup and holding costs while meeting the required demands (back-orders are not allowed).

3 ILP formulation

Because the resources are identical, in our formulation, we use the aggregate variables, as defined in [Gicquel et al., 2012]. The complete set of variables is:

x_{it} : number of resources producing item i on period t. Variables x_{i0} are defined in order to account for the number of startups in period 1 and should be made equal to a value that reflects the state of the various resources at the start of period 1;

y_{it} : number of resources where production of item i is started on period t and a startup cost is incurred;

z_{it} : number of demand units of item i carried as inventory from period t to period $t+1$. Variables z_{i0} are defined and should be fixed to reflect the inventory level at the start of period 1.

The complete ILP formulation is the following:

$$\min \quad \sum_{i=1}^{I} \sum_{t=1}^{T} (s_i y_{it} + h_i z_{it}) \tag{1}$$

$$\text{s.t.} \quad z_{i(t-1)} + x_{it} = d_{it} + z_{it} \qquad i = \{1, \ldots, I\},\ t = \{1, \ldots, T\} \tag{2}$$

$$y_{it} \geq x_{it} - x_{i(t-1)} \qquad i = \{1, \ldots, I\},\ t = \{1, \ldots, T\} \tag{3}$$

$$\sum_{i=1}^{I} x_{it} \leq R \qquad t = \{1, \ldots, T\} \tag{4}$$

$$x_{it} \geq 0 \text{ and integer} \qquad i = \{1, \ldots, I\},\ t = \{1, \ldots, T\} \tag{5}$$

$$y_{it} \geq 0 \text{ and integer} \qquad i = \{1, \ldots, I\},\ t = \{1, \ldots, T\} \tag{6}$$

$$z_{it} \geq 0 \text{ and integer} \qquad i = \{1, \ldots, I\},\ t = \{1, \ldots, T\} \tag{7}$$

Note that x_{i0} and z_{i0} are actually constants that reflect the initial state of the resources and the initial inventory levels. From this point on, for simplicity and without loss of generality we will assume these constants to be 0.

The objective function (1) sums the startup costs and the holding inventory costs. Constraints (2) express the inventory balance at each period. Constraints (3) ensure that a startup cost is incurred whenever the number of resources used for a given item increases. Finally, constraints (4) limit the number of resources used in each time period, and constraints (5), (6) and (7) specify the type and limits of the variables.

Using a similar formulation and a standard optimization package on a personal computer, [Gicquel et al., 2012] reported that they could not solve instances with $I = 10$, $R = 2$ and $T = 50$ within 30 minutes of computation. It is clear that solving this formulation directly is not practical, even for small instances.

4 Minimum cost flow formulation

When performing branch-and-bound it is important to be able to compute upper bounds. In this section we propose a minimum cost flow formulation for the DLSP. The formulation is incomplete in the sense

that inventory costs are accounted but not the startup costs, which means that the optimal solutions of the network flow problem, when they exist, are feasible to the DLSP, but not guaranteed to be optimal. A similar network for single item problems appears on [Zangwill, 1968].

Consider the following acyclic directed network. There is one supply node, S, whose supply is equal to RT. Consider also a set of T transshipment nodes, one for each time period, named $T_1, \ldots, T_T$. There are arcs from S to T_t with cost 0 and capacity equal to R.

Each of the T_t nodes will be connected to I demand nodes named $D_{1t}, \ldots, D_{It}$. The demand on the D_{it} nodes will be equal to d_{it} and the arcs from T_t to D_{it} have a cost of 0 and unlimited capacity (in practice, the limit will be R). The flow on these arcs has the same meaning as variables x_{it} of the ILP formulation.

Another set of directed arcs will depart from each D_{it} node to the node $D_{i(t+1)}$. These arcs have a cost equal to h_i and unlimited capacity. The flow on these arcs has the same meaning as variables z_{it} of the ILP formulation.

Finally, in order to balance the supply and the demand, consider an additional demand node, D_{idle}, whose demand, d_{idle}, is computed as[1]

$$d_{idle} = RT - \sum_{i=1}^{I} \sum_{t=1}^{T} d_{it}$$

Finally, an arc with cost equal to zero and unlimited capacity, should connect S and D_{idle}. The flow on this arc represents the global capacity excess on the resources.

The complete network is represented on Figure 1. Note that z_{i0} and z_{iT} can be used to account for, respectively, initial and final inventory levels, if there is need for them to be non-zero.

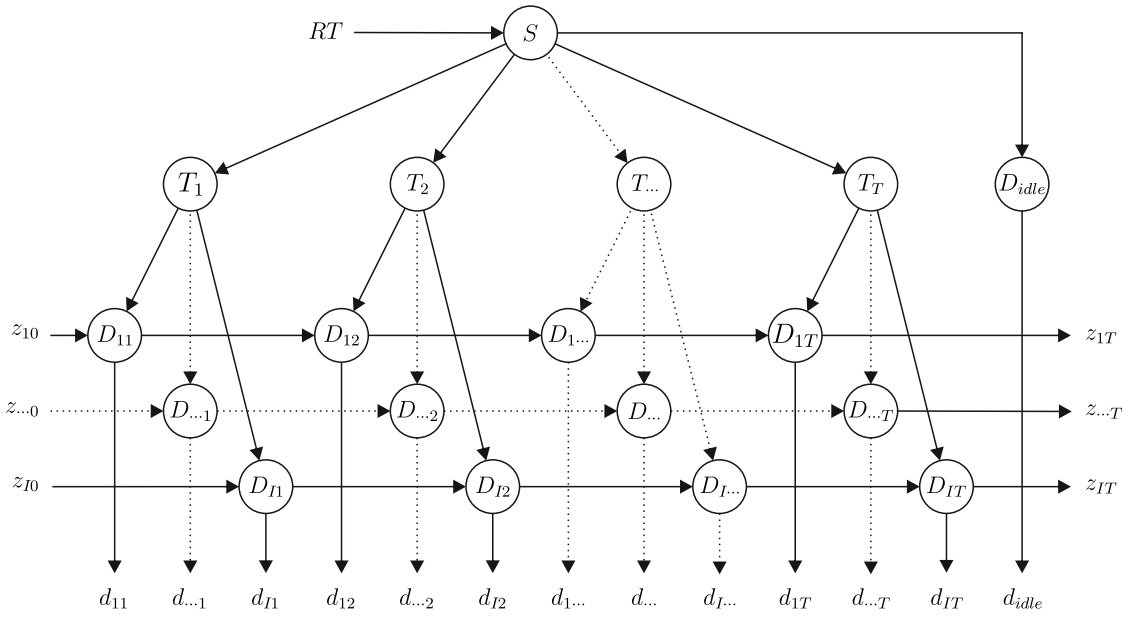

Figure 1: Minimum cost flow network representation.

Because the flow in arcs (T_t, D_{it}) has the same meaning as variables x_{it} of the ILP formulation, this network can be used to compute feasible solution to the DLSP that can be used as upper bounds, taking advantage of fast and widely available state-of-the-art minimum cost flow algorithms.

5 Dantzig-Wolfe decomposition

In this section we apply and present a standard Dantzig-Wolfe decomposition to the ILP formulation presented in section 3.

[1]Note that, if d_{idle} is negative, the problem is infeasible due to a global lack of resource capacity. If d_{idle} is non-negative, the problem can still be infeasible due to demand imbalances over time. A trivial way to check feasibility is to use the same principle to compute the idle capacity at every time period t', *i.e.*, $d^{t'}_{idle} = Rt' - \sum_{i=1}^{I} \sum_{t=1}^{t'} d_{it}$.

The ILP formulation has a block angular structure. With the exception of (4), which are coupling constraints, all other constraints can be grouped into I blocks, one for each product item. In our decomposition we will leave constraints (4) in the master problem and group all the constraints that refer to item i to a polyhedron named P_i.

Because any polyhedron P_i is a convex region, any point belonging to P_i can be represented as a convex combination of extreme points. Let p_{ik} be such points. For any P_i polyhedron there will be K_i extreme points, so that $k = 1, \ldots, K_i$. Let $\lambda_{ik} \geq 0$ be the weight of each extreme point in a given combination such that, for any given i, $\sum_{k=1}^{K_i} \lambda_{ik} = 1$. After variable substitution, the master problem will be:

$$\min \quad \sum_{i=1}^{I} \sum_{k=1}^{K_i} c_{ik} \lambda_{ik} \tag{8}$$

$$\text{s.t.} \quad \sum_{i=1}^{I} \sum_{k=1}^{K_i} a_{ikt} \lambda_{ik} \leq R \qquad t = \{1, \ldots, T\} \tag{9}$$

$$\sum_{k=1}^{K_i} \lambda_{ik} = 1 \qquad i = \{1, \ldots, I\} \tag{10}$$

$$\lambda_{ik} \geq 0 \text{ and integer} \qquad i = \{1, \ldots, I\},\ k = \{1, \ldots, K_i\} \tag{11}$$

In this reformulated model, columns can be interpreted as potential schedules for a single item, i, where c_{ik} is the cost of the schedule (including startup and inventory holding costs) and a_{ikt} is number of resources used by the schedule in period t.

Because it is not practical to enumerate all the potential single item schedules, they have to be dynamically generated. Based on the dual solution of the master problem, the subproblems will generate valid and cost attractive schedules to be included in the solution of the master problem.

Each P_i polyhedron will give origin to a different subproblem. Let π_t and ν_i be the dual variables associated with constraints (9) and (10), respectively. Subproblem i will have the following formulation:

$$\min \quad \sum_{t=1}^{T} (s_i y_{it} + h_i z_{it} - \pi_t x_{it}) - \nu_i \tag{12}$$

$$\text{s.t.} \quad z_{i(t-1)} + x_{it} = d_{it} + z_{it} \qquad t = \{1, \ldots, T\} \tag{13}$$

$$y_{it} \geq x_{it} - x_{i(t-1)} \qquad t = \{1, \ldots, T\} \tag{14}$$

$$0 \leq x_{it} \leq R \text{ and integer} \qquad t = \{1, \ldots, T\} \tag{15}$$

$$y_{it} \geq 0 \text{ and integer} \qquad t = \{1, \ldots, T\} \tag{16}$$

$$z_{it} \geq 0 \text{ and integer} \qquad t = \{1, \ldots, T\} \tag{17}$$

The subproblem is a single item DLSP on parallel resources. Note that the bounds on x_{it} in constraints (15) are included to avoid the generation of invalid schedules that will never be part of an optimal integer solution to the master problem.

After optimization, for a new column, $c_{ik} = \sum_{t=1}^{T} (s_i y_{it} + h_i z_{it})$ and, hence, the subproblem optimal objective function value is the reduced cost of that column. A generated column is added to the master problem, only if its reduced cost is negative. Also, coefficients a_{ikt} of the new column are equal to x_{it}.

Clearly, if the solution of the reformulated model has only integer variables, then an integer solution to DLSP can be computed. Nevertheless, one relevant characteristic of this problem is that an integer solution to DLSP can also be computed from non-integer variables of the reformulated model, whenever the solution of the reformulated model corresponds to an integer solution in the space of the original variables. This is fully exploited in the branch-and-price algorithm, because the solution in the space of the original variables has to be computed to derive the branching constraints; the branching scheme is presented in section 7.

The following proposition defines the set of conditions that a solution to the master problem must possess in order to be an integer solution to the DLSP:

Proposition 1. *For a solution to the DLSP problem to be integer, it is sufficient that all λ_{ik} variables are integer or that all x_{it} variables are integer, with*

$$x_{it} = \sum_{k=1}^{K_i} a_{ikt} \lambda_{ik} \tag{18}$$

Proof. This proposition can be easily demonstrated because λ_{ik} are binary variables that represent a single item schedule among all the resources, and, if they are all integer, they represent a valid solution. Variables x_{it} are the original formulation variables that represent the number of resources used by item i in time period t. Thus, if all x_{it} are integer, they represent a valid solution. □

Consider a new free decision variable, y'_{it} defined as $y'_{it} = x_{it} - x_{i(t-1)}$. This decision variable represents the change in the number of resources producing item i from period $t-1$ to period t. If there is an increase in the number of resources used, y'_{it} will be positive (equal to the formerly defined y_{it}) and, if there is a decrease, it will be negative. Given this definition, the following proposition is also true:

Proposition 2. *Given the sets of variables x_{it}, y'_{it} and z_{it}, if one of those sets is integer, then, the others must also be integer.*

Proof. Variables y'_{it} represent the variation in the number of used resources for a given item and can be computed from x_{it} as stated above. Hence if one of the sets is integer the other is also integer. Variables z_{it} are inventory levels and so $z_{it} = z_{i(t-1)} + x_{it} - d_{it}$. Because d_{it} are integer values, the previous reasoning still applies. □

6 Subproblem optimization

In this section we present a dynamic programming algorithm to solve the subproblem, a single item DLSP. The algorithm evaluates function $F_t(z, r)$ that represents the minimum cost to get z inventory level at the end of period t with r resources setup for the production of the considered item. If we assume that all resources are idle at instant 0, and the initial inventory is 0, then, $F_0(0, 0) = 0$. At each stage transition, we must decide how many resources will be allocated to the production of the considered item, i. Let $x_{it} = \{0, \ldots, R\}$ be that value. Then, from state (z, r) at stage $t-1$ we can reach, at stage t, states $(z' = z - d_{it} + x_{it}, r' = x_{it})$ as long as $z' \geq 0$, because inventory can not be negative. The objective function will be computed in the following way:

$$F_t(z', r') = \begin{cases} F_{t-1}(z, r) - \pi_t r' + h_i z' + s_i(r' - r) & \text{if } r' > r \\ F_{t-1}(z, r) - \pi_t r' + h_i z' & \text{if } r' \leq r \end{cases} \tag{19}$$

At each stage, the maximum theoretical number of states will be equal to $(R+1)(z_t^+ - z_t^- + 1)$, where z_t^- and z_t^+ are bounds on the inventory level at the end of period t and can be computed as follows:

$$z_t^- = \max\left(0, d_{i(t+1)} - R + z_{t+1}^-\right) \tag{20}$$

$$z_t^+ = \min\left(\sum_{l=1}^{t}(R - d_{il}), \sum_{l=t+1}^{T} d_{il}\right) \tag{21}$$

In equation (20) computation is recursive and should be initialized with $z_T^- = 0$, stating that the minimum inventory at the end of period T should be 0 (see discussion on section 4). The computations reflect the fact that, when the demand exceeds R, there will be need for inventory at the end of the previous period or periods.

Concerning the equation (21), the maximum inventory is the minimum value between the achievable inventory at the end of period t using maximum capacity and the maximum inventory needs to satisfy demand from inventory for the rest of the planning horizon (once again, assuming that the final inventory should be 0).

Note that these bounds can be used to improve (7) in the ILP formulation and (17) in the subproblem formulation and can be easily modified in the presence of initial and final inventories.

The above mentioned number of states is the theoretical maximum because if, for some state, $F_t(z, r)$ equals or exceeds ν_i, further transitions from that state can be ignored, because the reduced cost of the new column would not be negative and, hence, the column would not be attractive.

7 Branching

Solving the relaxed master problem to optimality does not guarantee an integer solution. For that reason, in order to find an integer optimal solution it is necessary to identify and eliminate fractional solutions. Branching is a standard procedure to achieve that goal.

As it is widely known, when performing column generation, branching on the master problem variables (λ_{ik}) is not a good idea, because it leads to column regeneration whenever a branching decision of the type $\lambda_{ik} \leq 0$ is made.

Given proposition 2, presented in section 5, the sets x_{it}, y'_{it} and z_{it} are natural candidates for branching. The choice should be made based on the results of computational performance tests.

Note that the original variables y_{it} cannot be used for branching because, although integrality on x_{it} implies integrality on y_{it}, the converse is not true. For example, consider the number of resources used (a_{ikt} vectors) in two four-period schedules for a given item: (0,4,4,4) and (4,4,3,1). Suppose that, in the optimal solution of a given node, both λ_{ik} are at a level of 0.5. As it can be easily seen, $x_{ik} = (2, 4, 3.5, 2.5)$ while $y_{ik} = (2, 2, 0, 0)$. This solution would be fractional, while the y_{it} vector would be integer. In this case, the vector y'_{it} would be (2,2,-0.5,-1) and, hence, not integer.

The following subsections present the 3 possible branching schemes along with the adjustments to the subproblem structure.

7.1 Branching on x_{it}

When branching upon the x_{it} variables, in node j, two branches of the problem are created. On one branch (the left branch) the constraint

$$x_{it} \leq \lfloor x_j^* \rfloor \tag{22}$$

is added, where x_j^* represents some non-integer value. On the other branch (the right branch) the following constraint is added instead:

$$x_{it} \geq \lceil x_j^* \rceil \tag{23}$$

With respect to finding the optimal solution of the model at a given node j, it is necessary to call the subproblems for attractive columns not yet included in the master problem. In node j, besides the initial constraints, the master problem has other sets of constraints, denoted as P_{it}^j, with $i = 1, \ldots, I$ and $t = 1, \ldots, T$, resulting from all the branching decisions imposed on each different variable x_{it}.

Let $\rho_{it,j}^p$ be the dual variable associated with constraint p, with $p \in P_{it}^j$. Thus, in order for the subproblem to correctly identify the attractive columns, in the objective function (12) and in the recursive equation (19), π_t must be replaced with $(\pi_t + \rho_{it}^j)$, where ρ_{it}^j is the sum of all dual variables, $\rho_{it,j}^p$, associated with constraints $p \in P_{it}^j$, which are imposed on the variable x_{it} at node j, *i.e.*, $\rho_{it}^j = \sum_{p \in P_{it}^j} \rho_{it,j}^p$.

7.2 Branching on z_{it}

Branching on the z_{it} variables requires some additional manipulations. Developing $z_{it} = z_{i(t-1)} + x_{it} - d_{it}$ recursively yields the following (assuming the starting inventory is 0):

$$z_{it} = \sum_{l=1}^{t} (x_{il} - d_{il}) \tag{24}$$

To translate z_{it} to the master space, once again, equation (18) should be used. Using the same approach as before, on node j we want to branch on variable z_{it}, whose fractional value is z_j^*. The left and right branching constraints will be, respectively:

$$z_{it} \leq \lfloor z_j^* \rfloor \tag{25}$$

$$z_{it} \geq \lceil z_j^* \rceil \tag{26}$$

Using the same notation as in section 7.1, if ρ_{it}^j is the sum of the dual variables that refer to constraints imposed on the variable z_{it}, the modification to objective function (12) and to the recursive equation (19) is the replacement of h_i by $(h_i - \rho_{it}^j)$.

7.3 Branching on y'_{it}

Let ${y'_j}^*$ be the fractional value of y'_{it} that we wish to branch upon on node j. The constraints to impose on the left and right branches are, respectively,

$$y'_{it} \leq \lfloor {y'_j}^* \rfloor \tag{27}$$

$$y'_{it} \geq \lceil {y'_j}^* \rceil \tag{28}$$

On these equations, y'_{it} can be replaced with $x_{it} - x_{i(t-1)}$ and projected to the master problem space using equation (18). Once again, as in the previous sections, let ρ^j_{it} be the sum of the dual variables whose associated constraints refer to variable y'_{it}.

In this case, the modifications to the subproblem structure are more complex than in the previous branching schemes presented on sections 7.1 and 7.2.

In the case of the ILP formulation there is the need of creating a set of variables to account for decreases in the number of used resources. Let's name those variables y^-_{it}. In the objective function (12) a new term associated with this new variables must be included rendering the following objective function:

$$\sum_{t=1}^{T} \left((s_i - \rho^j_{it}) y_{it} + h_i z_{it} - \pi_t x_{it} + \rho^j_{it} y^-_{it} \right) - \nu_i \tag{29}$$

Also, an additional set of constraints must be included (similar to constraints (14)):

$$y^-_{it} \geq x_{i(t-1)} - x_{it} \qquad t = \{1, \ldots, T\} \tag{30}$$

Also, in the subproblem formulation that resulted from the decomposition, the y_{it} variables have no upper bound because it is implicitly assumed that their coefficients on the objective function are always positive. Because this last assumption is no longer true, an upper bound on y_{it} equal to $\max\left(0,\ x_{it} - x_{i(t-1)}\right)$ must be enforced in the ILP subproblem formulation. The same logic applies to the y^-_{it} variables: an upper bound equal to $\max\left(0,\ x_{i(t-1)} - x_{it}\right)$ must be enforced. For simplicity, the necessary additional constraints are omitted here.

The recursive equation (19) needs also to be modified and, after the necessary modifications, it will be:

$$F_t(z', r') = \begin{cases} F_{t-1}(z, r) - \pi_t r' + h_i z' + (s_i - \rho^j_{it})(r' - r) & \text{if } r' > r \\ F_{t-1}(z, r) - \pi_t r' + h_i z' + \rho^j_{it}(r - r') & \text{if } r' \leq r \end{cases} \tag{31}$$

With this changes, the subproblem will correctly process the additional dual information.

8 Computational results

To assess the quality of the bounds, a total of 10 instances was solved. The first 3 instances are very small and were used for debugging purposes. The other 7 instances are also small, with only 4 items, 4 resources and 5 time periods (10 periods for the last two), and were generated using the procedure described in [Gicquel et al., 2012].

The results are shown in table 1 and the columns have the following meaning: columns R, I and T are, respectively, the number of resources, the number of items and the number of periods; column *UC* is the used capacity, in percentage; column *ILPRel* is the optimal value of the LP relaxation of the ILP formulation presented in section 3; column *Network* is the optimal value for the network flow model presented in section 4; column *MRoot* is the optimal value for the master problem of the reformulated model at the root node, with the asterisk denoting integer solutions where no branching was necessary; the last column is the value of the integer optimal solution, after branching as needed.

Table 1: Preliminary computational results.

Instance	R	I	T	UC	ILPRel	Network	MRoot	Optimal
1	3	3	4	100	21.25	28	25*	25
2	4	3	4	100	80.00	92	84	84
3	4	3	4	81.25	55.25	74	57	57
4	4	4	5	95	1426.33	2454	1534*	1534
5	4	4	5	90	1118.25	3219	1350	1350
6	4	4	5	85	667.33	2005	753*	753
7	4	4	5	80	990.67	1688	1105*	1105
8	4	4	5	75	1221.33	2088	1412*	1412
9	5	5	10	100	2813.17	5411	3282.5	3368
10	5	5	10	88	1574.32	4688	1673*	1673

Although the tested instances were very small it can be seen that the reformulated model is stronger than the original formulation, as the lower bound given by its LP relaxation is better than the one given

by the LP relaxation of the original model. With the exception of instance 9, this bound is equal to the optimal solution value.

Concerning the network flow model, it can be seen that the provided upper bound can be arbitrarily bad. Because the model only considers inventory holding costs, when this costs are small compared to the startup costs (as it happens in instances 4 to 10)[2] the bound is very high.

As these preliminary results were encouraging, an implementation was developed in C# (Microsoft .NET framework 4.5) using ILOG CPLEX 12.5.0.1 for optimization, with the default parameters, and run in a laptop with a Intel Core i7 3610QM @ 2.30GHz CPU. The branching scheme is based on the x_{it} variables, as described in section 7.1.

All the test instances were generated using the procedure described in [Gicquel et al., 2012]. Furthermore, the instances have similar characteristics, namely, there are 4 sets of instances:

- set A: small instances ($R = 2$, $I = 10$ and $T = 50$);
- set B: instances with a large number of periods ($R = 2$, $I = 10$ and $T = 150$);
- set C: instances with a large number of items ($R = 2$, $I = 25$ and $T = 50$);
- set D: instances with a large number of resources ($R = 10$, $I = 10$ and $T = 50$).

These sets were combined with 5 levels of used capacity (75%, 80%, 85%, 90% and 95%). For each combination, 3 instances were generated, resulting in a total of 60 instances.

The computational results are shown in table 2, where each line contains aggregate results for the 3 instances in each combination described above, and the columns have the following meaning: column *UC* refers to the used capacity; columns *Nodes* and *Cols* are the average number of nodes in the branch-and-price tree and the average number of columns generated, respectively; columns *TMIP* and *TBP* are average times (in seconds) to solve to optimality the ILP formulation presented in section 3 (*TMIP*) using the CPLEX MIP Solver and the proposed branch-and-price framework (*TBP*), respectively; columns *SMIP* and *SBP* show the number of instances solved to optimality using each procedure within a time limit of 30 minutes; column *LBInc* shows the average increase, in percentage of the ILP formulation LP relaxation bound, to the LP relaxation of the reformulated model[3]; finally, column *Gap* shows the average gap, in percentage, between the LP relaxation of the root node and the optimal (or best) solution found[4].

Table 2: Extended computational results.

Instance set	UC	Nodes	Cols	TMIP	SMIP	TBP	SBP	LBInc	Gap
	75	1.7	602.3	2.36	3	0.39	3	89.4	0.01
A: $R = 2$,	80	3.0	512.0	0.90	3	0.55	3	70.0	0.06
$I = 10$ and	85	1.0	774.0	4.09	3	0.45	3	85.6	0.00
$T = 50$	90	5.7	1031.0	1.20	3	0.41	3	67.0	0.16
	95	34.0	1686.0	4.85	3	0.66	3	69.0	0.32
	75	251.3	4795.7	219.47	2	14.42	3	87.3	0.30
B: $R = 2$,	80	3844.3	22495.0	-	0	255.79	3	96.4	0.26
$I = 10$ and	85	2051.3	35514.0	-	0	19.07	2	75.9	0.43
$T = 150$	90	617.0	18415.7	-	0	91.90	3	78.8	0.32
	95	2624.0	78555.7	-	0	1046.95	2	75.6	0.48
	75	3.7	583.0	1.20	3	0.52	3	88.0	0.03
C: $R = 2$,	80	1.0	716.0	3.20	3	0.57	3	90.9	0.00
$I = 25$ and	85	35.7	1040.7	5.44	3	0.53	3	105.3	0.05
$T = 50$	90	55.7	1010.7	5.09	3	0.50	3	85.9	0.13
	95	700.3	1710.7	4.73	3	1.10	3	71.5	0.18
	75	30.3	573.0	0.99	3	5.60	3	8.9	0.29
D: $R = 10$,	80	5.7	858.3	1.44	3	5.90	3	10.2	0.03
$I = 10$ and	85	1228.3	3080.0	1.97	3	50.21	3	6.3	0.37
$T = 50$	90	465.3	5410.0	1.94	3	144.79	3	7.8	0.28
	95	3185.0	12928.3	6.85	3	214.27	2	9.3	0.67

These results seem to confirm the quality of the lower bound of the reformulated model, as shown by the increase in the bound when comparing with the ILP formulation and the small gap to the optimal solution.

[2] holding costs are drawn from a discrete uniform distribution between 5 and 10 and startup costs from a discrete uniform distribution between 100 and 200.

[3] Using the notation presented in section 8, $LBInc = 100 \times (MRoot - ILPRel) / ILPRel$.

[4] If *Best* represents the optimal or best solution found, $Gap = 100 \times (Best - MRoot) / MRoot$.

Except for the instances in set D, where the generic MIP solver outperformed the proposed framework, in all the other instances the results are favorable to the proposed framework. For the instances in set B, which are difficult to solve with the ILP formulation, our framework solved 13 out of 15 instances, while the MIP solver solved only 2.

Future research efforts should try to fully understand the results for set D and improve the performance for those set of instances, probably with the help of additional cuts, different branching schemes and/or with an heuristic approach.

9 Acknowledgments

The authors want to thank the anonymous reviewers of the IO2013 conference for the insightful comments to the first version of this paper.

References

[Caserta and Voß, 2013] Caserta, M. and Voß, S. (2013). A math-heuristic Dantzig-Wolfe algorithm for capacitated lot sizing. *Annals of Mathematics and Artificial Intelligence*, pages 1–18.

[Cattrysse et al., 1993] Cattrysse, D., Salomon, M., Kuik, R., and van Wassenhove, L. N. (1993). A dual ascent and column generation heuristic for the discrete lotsizing and scheduling problem with setup times. *Management Science*, 39(4):477–486.

[Degraeve and Jans, 2007] Degraeve, Z. and Jans, R. (2007). A new Dantzig-Wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Operations Research*, 55(5):909–920.

[Gicquel et al., 2011] Gicquel, C., Minoux, M., and Dallery, Y. (2011). Exact solution approaches for the discrete lot-sizing and scheduling problem with parallel resources. *International Journal of Production Research*, 49(9):2587–2603.

[Gicquel et al., 2012] Gicquel, C., Wolsey, L. A., and Minoux, M. (2012). On discrete lot-sizing and scheduling on identical parallel machines. *Optimization Letters*, 6(3):545–557.

[Karimi et al., 2003] Karimi, B., Fatemi Ghomi, S. M. T., and Wilson, J. M. (2003). The capacitated lot sizing problem: a review of models and algorithms. *Omega*, 31(5):365–378.

[Lasdon and Terjung, 1971] Lasdon, L. S. and Terjung, R. C. (1971). An efficient algorithm for multi-item scheduling. *Operations Research*, 19(4):946–969.

[Manne, 1958] Manne, A. S. (1958). Programming of economic lot sizes. *Management Science*, 4(2):115–135.

[van Eijl and van Hoesel, 1997] van Eijl, C. A. and van Hoesel, C. P. M. (1997). On the discrete lot-sizing and scheduling problem with Wagner-Whitin costs. *Operations Research Letters*, 20(1):7–13.

[van Hoesel and Kolen, 1994] van Hoesel, S. and Kolen, A. (1994). A linear description of the discrete lot-sizing and scheduling problem. *European Journal of Operational Research*, 75(2):342–353.

[van Hoesel et al., 1991] van Hoesel, S., Wagelmans, A., and Kolen, A. (1991). A dual algorithm for the economic lot-sizing problem. *European Journal of Operational Research*, 52(3):315–325.

[Wagner and Whitin, 1958] Wagner, H. M. and Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management Science*, 5(1):89–96.

[Zangwill, 1968] Zangwill, W. I. (1968). Minimum concave cost flows in certain networks. *Management Science*, 14(7):429–450.