On Discrete Lot Sizing and Scheduling on Identical Parallel Machines

Céline Gicquel, Laurence Wolsey, Michel Minoux

Discrete Lot-Sizing and Scheduling on Identical Parallel Machines

C. Gicquel L. A. Wolsey M. Minoux

Abstract

We consider the multi-item discrete lot-sizing and scheduling problem on identical parallel machines. Based on the fact that the machines are identical, we introduce aggregate integer variables instead of individual variables for each machine. For the problem with start-up costs, we show that the inequalities based on a unit flow formulation for each machine can be replaced by a single integer flow formulation without any change in the resulting LP bound. For the resulting integer lot-sizing with start-ups subproblem, we show how inequalities for the unit demand case can be generalized and how an approximate version of the extended formulation of Eppen and Martin can be constructed. The results of some computational experiments carried out to compare the effectiveness of the various mixed-integer programming formulations are presented.

Keywords: Discrete lot-sizing and scheduling Identical parallel machines Mixed-integer linear programming, Reformulation Valid inequalities

1 Introduction

In the present paper, we consider the multi-item discrete lot-sizing and scheduling problem on identical parallel machines.

The case with one machine, known as the discrete lot-sizing and scheduling problem (DLSP) has the particularity that at most one item is produced per period, that production of that item (if any) is at full capacity and that start-up or changeover costs have to be incurred when switching production from one item to another. A natural subproblem of this model is the single item discrete lot-sizing problem with start-ups, where there are linear production and storage costs and fixed production and start-up costs. For this Van Eijl and van Hoesel [11] have presented valid inequalities and van Hoesel and Kolen [12] have provided a tight and compact extended formulation.

A different single-item parallel resource model with start-ups was used by Lasdon and Terjung [7, 8]. Here the demands are integers, the amount produced is integral and there are linear production, storage and start-up costs. Eppen and Martin [4] have given a tight extended formulation and Vanderbeck and Wolsey [10] have derived several families of valid inequalities. The multi-item problem with several machines has also been studied recently by Gicquel et al. [5].

The purpose of this paper is threefold. We first introduce in Section 2 two formulations of the problem, using either machine-specific binary variables or aggregate integer variables, and provide equivalence results between these two formulations. In Section 3 we propose two ways of strengthening the second formulation involving the aggregate integer variables. On the one hand, we develop an approximate extended formulation based on the extended formulation proposed in [4], and on the other we derive a new family of valid inequalities, that can be seen as an extension to the parallel-machine case of the valid inequalities proposed in [11] for the single-machine problem. Finally in Section 4 we provide some computational results carried out to compare the various formulations of the problem. Our results indicate that it is possible to solve instances with 10 or more items, up to 10 machines and up to 50 periods to optimality, and that for instances with up to 150 periods, solutions within 5% of optimality can be obtained in less than 10 minutes.

2 Problem Formulations and Equivalence Results

We first give a formulation for the problem with start-up costs and *machine specific* 0-1 variables. There are I items, T periods and K identical machines. As the machines are identical and produce just one item at full capacity in a period, the demand is measured in units of full capacity. The data consists of the demands d_t^i for item i in period t, the (machine-independent) production cost p_t^i of producing item i on one of the machines in period t, the (machine-independent) start-up cost q_t^i of starting to produce item i on a machine in period t and h_t^i the storage cost for item i at the end of period t. Note that one can assume without loss of generality that $d_t^i \in \{0, 1, \ldots, K\}$. Also an additional item, item 0, is introduced to represent the case when a machine is idle.

The variables are:
- $y_t^{ik} = 1$ if item $i \in \{0, \ldots, I\}$ is produced on machine k in period t, and $y_t^{ik} = 0$ otherwise
- $z_t^{ik} = 1$ if production of item i on machine k is started in period t, and $z_t^{ik} = 0$ otherwise,
- s_t^i is the stock of item i at the end of period t.

The corresponding formulation is:

$$v^{BIN} = \min \sum_{i=1}^{I}\sum_{k=1}^{K}\sum_{t=1}^{T}(p_t^i y_t^{ik} + q_t^i z_t^{ik}) + \sum_{i=1}^{I}\sum_{t=1}^{T} h_t^i s_t^i \tag{1}$$

$$s_{t-1}^i + \sum_{k=1}^{K} y_t^{ik} = d_t^i + s_t^i \qquad \forall\, i,t \tag{2}$$

$$z_t^{ik} \geq y_t^{ik} - y_{t-1}^{ik} \;\forall\, i,k,t \tag{3}$$

$$z_t^{ik} \leq y_t^{ik} \;\forall\, i,k,t \tag{4}$$

$$y_{t-1}^{ik} + z_t^{ik} \leq 1 - \sum_{j:j\neq i} (y_t^{jk} - z_t^{jk}) \;\forall\, i,k,t \tag{5}$$

$$\sum_{i=0}^{I} y_t^{ik} = 1 \;\forall\, k,t \tag{6}$$

$$s \in \mathbb{R}_+^{IT}, \qquad y, z \in \{0,1\}^{(I+1)KT} \tag{7}$$

Note that inequality (5) is a strengthened version of the more standard inequality $y_{t-1}^{ik} + z_t^{ik} \leq 1$, see Constantino [2].

Let Q^{yz} be the polyhedron $\{(y,z) \in \mathbb{R}_+^{(I+1)KT} \times \mathbb{R}_+^{(I+1)KT} : (3)-(6)\}$ and $X^{yz} = Q^{yz} \cap \mathbb{Z}^{(I+1)KT} \times \mathbb{Z}^{(I+1)KT}$.

We now introduce the *aggregate* variables

$$Y_t^i = \sum_{k=1}^{K} y_t^{ik} \;\text{ and }\; Z_t^i = \sum_{k=1}^{K} z_t^{ik} \tag{8}$$

and the resulting aggregated formulation

$$v^{INT} = \min \sum_{i=1}^{I}\sum_{t=1}^{T}(p_t^i Y_t^i + q_t^i Z_t^i + h_t^i s_t^i) \tag{9}$$

$$s_{t-1}^i + Y_t^i = d_t^i + s_t^i \quad \forall\, i,t \tag{10}$$

$$Z_t^i \geq Y_t^i - Y_{t-1}^i \;\forall\, i,t \tag{11}$$

$$Z_t^i \leq Y_t^i \quad \forall\, i,t \tag{12}$$

$$Y_{t-1}^i + Z_t^i \leq K - \sum_{j:j\neq i} (Y_t^j - Z_t^j) \;\forall\, i,t \tag{13}$$

$$\sum_{i=0}^{I} Y_t^i = K \;\forall\, t \tag{14}$$

$$s \in \mathbb{R}_+^{IT}, \qquad y, z \in \mathbb{Z}_+^{(I+1)T}. \tag{15}$$

Note that the slack variable in (11), $\Omega_{t-1}^i = Z_t^i - Y_t^i + Y_{t-1}^i$, can be seen as the number of machines turned off in period $t-1$.

As above, let Q^{YZ} be the polyhedron $\{(Y,Z) \in \mathbb{R}_+^{(I+1)T} \times \mathbb{R}_+^{(I+1)T} : (11)-(14)\}$ and $X^{YZ} = Q^{YZ} \cap \mathbb{Z}^{(I+1)T} \times \mathbb{Z}^{(I+1)T}$.

It is easily verified that if $(y,z) \in Q^{yz}$ and Y, Z are defined as in (8), then $(Y,Z) \in Q^{YZ}$. Below we establish the converse.

Theorem
i) If $(Y,Z) \in Q^{YZ}/X^{YZ}$, then there exists $(y,z) \in Q^{yz}/X^{yz}$ satisfying (8),
ii) $Q^{yz} = \text{conv}(X^{yz})$ and $Q^{YZ} = \text{conv}(X^{YZ})$.

Proof
The proof involves an extended formulation for Q^{yz} and Q^{YZ}. As in Belvaux and Wolsey [1], we define the changeover variables as $w_t^{ijk} = 1$ if item i is produced on machine k in period $t-1$ and item j is produced in period t and $w_t^{ijk} = 0$ otherwise, and the polyhedron P^{yzw}:

$$\sum_i w_t^{ijk} = y_t^{jk} \ \forall\ j,k,t \tag{16}$$

$$\sum_j w_t^{ijk} = y_{t-1}^{ik} \ \forall\ i,k,t \tag{17}$$

$$\sum_{i:i\neq j} w_t^{ijk} = z_t^{jk} \ \forall\ j,k,t \tag{18}$$

$$\sum_i y_1^{ik} = 1 \ \forall\ k \tag{19}$$

$$w \in \mathbb{R}_+^{(I+1)^2KT}, \qquad y, z \in \mathbb{R}_+^{(I+1)KT}. \tag{20}$$

We also introduce the aggregate changeover variables

$$W_t^{ij} = \sum_{k=1}^{K} w_t^{ijk} \tag{21}$$

leading to the aggregated polyhedron P^{YZW}:

$$\sum_i W_t^{ij} = Y_t^j \ \forall\ j,t \tag{22}$$

$$\sum_j W_t^{ij} = Y_{t-1}^i \ \forall\ i,t \tag{23}$$

$$\sum_{i:i\neq j} W_t^{ij} = Z_t^j \ \forall\ j,t \tag{24}$$

$$\sum_i Y_1^i = K \tag{25}$$

$$W \in \mathbb{R}_+^{(I+1)^2T}, \qquad Y, Z \in \mathbb{R}_+^{(I+1)T}. \tag{26}$$

Constantino [2] has shown that Q^{yz} is the projection of P^{yzw} onto the space of the y,z variables. The argument uses the max-flow/min cut theorem applied to (16)-(17) viewed as a transportation network. As P^{yzw} is an integral polytope, this implies that Q^{yz} is also an integral polytope. An identical argument holds for P^{XYZ} and Q^{YZ}.

It remains to show that if $(Y,Z,W) \in P^{YZW}$, then there exists a decomposed machine specific solution $(y,z,w) \in P^{yzw}$ satisfying the aggregation equations (8) and (21). However (22),(23),(26)

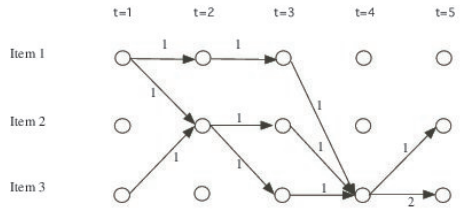

Figure 1: *An Integer K-flow (with K=3)*

define a flow of K units through the network shown in Figure 1. By standard results on flows, any such flow decomposes into K unit flows, and any integer flow decomposes into K unit integer flows.

The result follows as now any $(Y, Z) \in Q^{YZ}$ corresponds to a $(Y, Z, W) \in P^{YZW}$ which, as just shown, corresponds to a $(y, z, w) \in P^{yzw}$ and hence to a $(y, z) \in Q^{yz}$, and integer solutions also lead to integer solutions.

It follows that the aggregated and disaggregated formulations are equivalent both as integer and linear programs. Let v_{LP}^{BIN}, v_{LP}^{INT} denote the values of the linear programming relaxations of (1)-(7) and (9)-(15) respectively.

Corallary
$v^{INT} = v^{BIN}$ and $v_{LP}^{INT} = v_{LP}^{BIN}$.

The potential benefits from use of the aggregated formulation are i) its smaller size and ii) the fact that the most obvious problems of symmetry of the solutions of the disaggregated model are removed. Note also that whereas complete enumeration of the y vectors in the disaggregated model gives $(2^K)^{(I+1)T}$ distinct vectors, the number of Y vectors in the aggregated model is $(K+1)^{(I+1)T}$.

3 Strengthening the Aggregate Formulation

Here we consider ways to strengthen the formulation (9)-(15), and in particular the single item set X^{LT}:

$$
\begin{aligned}
s_{t-1} + Y_t &= d_t + s_t \ \forall\, t && (27)\\
Z_t &\geq Y_t - Y_{t-1} \ \forall\, t && (28)\\
Z_t &\leq Y_t \ \forall\, t && (29)\\
Y, Z &\in \{0, 1, \cdots, K\}^T && (30)
\end{aligned}
$$

treated by Lasdon-Terjung [8].

From now on we will use the notation $d_{ut} \equiv \sum_{j=u}^{t} d_j$, $Y_{ut} \equiv \sum_{j=u}^{t} Y_j$, etc., with a superscript i to denote the item when appropriate.

Two ways to strengthen this formulation have been proposed. Eppen and Martin [4] gave an extended formulation for conv(X^{LT}) with $O(K^2 T d_{1T})$ variables and $O(KT d_{1T})$ constraints. Vanderbeck and Wolsey [10] gave three families of valid inequalities along with separation heuristics. Here we give an approximate version of the extended formulation, and also show how a family of inequalities developed by Van Eijl and Van Hoesel [11] can be adapted to give valid inequalities for this model.

3.1 An Approximate Extended Formulation

We modify the extended formulation given by Eppen and Martin [4]. The approximation is obtained by aggregating nodes and arcs whenever the stock level exceeds some user specified value Δ.

We now describe the construction, given an approximation parameter Δ and the fact Y_t is bounded by the number of machines K. One constructs a directed graph $D = (V, A)$ with nodes (t, α, β) where t represents the time period, α represents Y_{1t} and β represents Y_t for $d_{1t} \leq \alpha \leq d_{1t} + \Delta$ and nodes (t, α) for $d_{1t} + \Delta < \alpha \leq d_{1T}$. The arcs $(t, \alpha, \beta, \gamma)$ depart from node (t, α, β) and arrive at either node $(t+1, \alpha+\gamma, \gamma)$ or node $(t+1, \alpha+\gamma)$ with corresponding variables $u(t, \alpha, \beta, \gamma)$. The second set of arcs depart from node (t, α) and arrive at either node $(t + 1, \alpha + \gamma, \gamma)$ or node $(t + 1, \alpha + \gamma)$ with corresponding variables $v(t, \alpha, \gamma)$. See Figure 2.

Thus the arc variables are defined as follows:
- $u(t, \alpha, \beta, \gamma) = 1$ if $Y_{1t} = \alpha, Y_t = \beta, Y_{t+1} = \gamma$, and 0 otherwise with $d_{1t} \leq \alpha \leq \min(d_{1t} + \Delta, d_{1T})$, $d_{1T} \geq \alpha + Y_{t+1} \geq d_{1,t+1}$, $Y_t, Y_{t+1} \in \{0, \cdots, K\}$, and
- $v(t, \alpha, \gamma) = 1$ if $Y_{1t} = \alpha, Y_{t+1} = \gamma$, and 0 otherwise with $d_{1T} \geq \alpha > d_{1t} + \Delta$, $d_{1T} \geq \alpha + Y_{t+1} \geq d_{1,t+1}$, $Y_t, Y_{t+1} \in \{0, \cdots, K\}$.

The resulting "approximate dynamic programming" formulation is:

$$
\begin{aligned}
u(0,0,0) &= 1 && (31)\\
\sum_{\gamma} u(t,\alpha,\beta,\gamma) &= \sum_{\gamma} u(t-1,\alpha-\beta,\gamma,\beta) + v(t-1,\alpha-\beta,\beta)\ \forall \text{nodes } (t,\alpha,\beta) && (32)\\
\sum_{\gamma} v(t,\alpha,\gamma) &= \sum_{\beta} \big[\sum_{\gamma} u(t-1,\alpha-\beta,\gamma,\beta) + v(t-1,\alpha-\beta,\beta)\big]\ \forall \text{nodes } (t,\alpha) && (33)\\
Y_t &= \sum_{\alpha,\gamma} \gamma\big[\sum_{\beta} u(t-1,\alpha,\beta,\gamma) + v(t-1,\alpha,\gamma)\big]\ \forall\ t && (34)\\
Z_t &\geq \sum_{\alpha,\beta,\gamma} (\gamma-\beta)^{+} u(t,\alpha,\beta,\gamma)\ \forall\ t && (35)\\
Z_t &\geq Y_t - Y_{t-1}\ \forall\ t && (36)\\
Z_t &\leq Y_t\ \forall\ t && (37)\\
Y, Z \in [0,K]^T, &\quad u(t,\alpha,\beta,\gamma) \in \mathbb{R}^1_+\ \forall\ t,\alpha,\beta,\gamma,\quad v(t,\alpha,\beta) \in \mathbb{R}^1_+\ \forall\ t,\alpha,\beta, && (38)
\end{aligned}
$$

where $D = \max_t d_t$. Note that when $\Delta = d_{1T}$, the v variables and the equations (33) disappear and one obtains the original extended formulation from [4] with equality in (35).

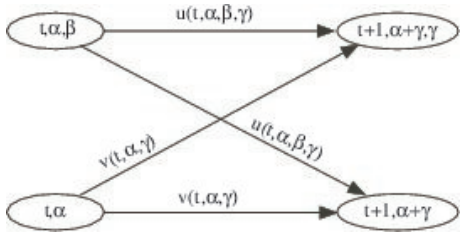

Figure 2: *The variables in the approximate extended formulation*

3.2 A Family of Valid Inequalities

We show that the valid inequalities of Van Eijl and Van Hoesel [11] derived for the case where $Y_t \in \{0, 1\}$ and $d_t \in \{0, 1\}$ for all t are also valid for X^{LT}. Specifically we break the integer demands d_t into demands of one unit. Thus given an interval $[t, l]$, we say that the q^{th} unit of

demand occurs in period t_q if $d_{t,t_q-1} < q$ and $d_{t,t_q} \geq q$. Thus $t \leq t_1 \leq \cdots \leq t_q \leq \cdots \leq t_p \leq l$.

Proposition
The inequality

$$s_{t-1} + \sum_{q=1}^{p} (Y_{t+q-1} + Z_{t+q,t_q}) \geq p \tag{39}$$

is valid for X^{LT}, where $d_{tl} = p$ and $t + q < t_q$ for $q = 1, \ldots, p$.

Proof
Let $\alpha(t, p)$ be the expression on the left of the inequality. We use induction.
Initial step. From flow conservation and the definition of t_1, $s_{t-1} + Y_{t,t_1} \geq 1$. So either $s_{t-1} \geq 1$, or $Y_{t,t_1} \geq 1$. However $Y_{t,t_1} \geq 1$ if and only if $Y_t + Z_{t+1,t_1} \geq 1$, and the claim follows.
Inductive step. We assume that $\alpha(t, p) \geq p$ is valid. There are two cases.
Case 1. $Y_{t+p} + Z_{t+p+1,t_{p+1}} \geq 1$. Adding this inequality to the inequality $\alpha(t, p) \geq p$ gives $\alpha(t, p + 1) \geq p + 1$.
Case 2. $Y_{t+p} + Z_{t+p+1,t_{p+1}} = 0$. This implies that $Y_u = 0$ for $u = t + p, \cdots, t_{p+1}$, which in turn implies that

$$s_{t-1} + Y_{t,t+p-1} = s_{t-1} + Y_{t,t_{p+1}} \geq d_{t,t_{p+1}} \geq p + 1,$$

where the first inequality follows from the flow conservation equations (27) and the second from the definition of t_{p+1}. However $\alpha(t, p + 1) \geq s_{t-1} + Y_{t,t+p-1}$ and the claim follows.

Observe that the inequality (39) is dominated by $s_{t-1} + Y_{t,t_p} \geq d_{t,t_p}$ if $t + p \geq t_p$.

4 Numerical results

We now present the results of some computational experiments carried out to compare the effectiveness of the various mixed-integer programming formulations discussed in sections 2 and 3 in solving the problem under study.

4.1 Problem instance generation

We randomly generated instances of the problem using a procedure adapted from that described in [9] for the multi-item DLSP with a single machine. More precisely, the various instances tested differ with respect to the following characteristics:
- *Problem dimension*: The problem dimension is represented by the number of items I, the number of periods T and the number of machines K. We define 4 sets of instances:
  - set A (small instances): $I = 10, T = 50, K = 2$
  - set B (instances with a large number of periods): $I = 10, T = 150, K = 2$
  - set C (instances with a large number of items): $I = 25, T = 50, K = 2$
  - set D (instances with a large number of machines): $I = 10, T = 50, K = 10$
- *Costs*: For each item, inventory holding costs (resp. startup costs) have been randomly generated from a discrete uniform $DU(5, 10)$ distribution (resp. from a discrete uniform $DU(100, 200)$ distribution).
- *Production capacity utilization*: Production capacity utilization ρ is defined as the ratio between the total cumulated demand ($\sum_{i=1}^{N} d_{1T}^i$) and the total cumulated available capacity ($K \times T$). ρ was varied between 0.75 and 0.95, in steps of 0.05.
- *Demand pattern*: Integer demands $d_t^i \in \{1, ...K\}$ for each item have been randomly generated according to the following procedure:

1. We randomly select an item i^* from a discrete uniform $DU(1, N)$ distribution and generate a value $d_T^{i^*}$ from a discrete uniform $DU(1, K)$ distribution.
2. For each item i, except item i^*, we randomly select a period t_i from a discrete uniform $DU(1, T)$ distribution and generate a value $d_{t_i}^i$ from a discrete uniform $DU(1, K)$ distribution.
3. For each entry in a $N \times T$ matrix, except for the entries corresponding to the (i, t) combinations for which we set $d_t^i > 0$ in steps 1 or 2, we randomly generate a number α_{it} from a discrete uniform $DU(1, NT)$ distribution.
4. While the total cumulated demand $(\sum_{i=1}^N d_{1T}^i)$ does not exceed ρKT, we consider the entries (i, t) one by one in the increasing order of the corresponding value α_{it} and generate a value d_t^i from a discrete uniform $DU(1, K)$ distribution.
5. When the total cumulated demand reaches ρKT, we examine whether the corresponding instance is feasible by checking that $\sum_{i=1}^N d_{1t}^i \leq Kt$ for all t. If the instance is infeasible, we repeat steps 1 to 4.

For each possible combination of problem dimension and production capacity utilization, 5 instances were generated, resulting in a total of $4 \times 5 \times 5 = 100$ instances.

4.2 Computational experiments

All tests were run on a Pentium 4 (2.8 Ghz) with 505 Mb of RAM, running under Windows XP. We used a standard MILP software (CPLEX 11.1) with the solver default settings to solve the problems with one of the following formulations:
- AGG: Aggregate formulation (9)-(15).
- AGG1: Aggregate formulation (9)-(15) strengthened by the valid inequalities proposed in [10] for each item. These valid inequalities are added to the formulation at the root node of the Branch & Bound tree. The corresponding cutting-plane generation algorithm is based on the heuristic separation routine described by the authors of [10].
- AGG2: Aggregate formulation (9)-(15) strengthened by the valid inequalities (39) proposed in section 3 for each item. These valid inequalities are added to the formulation at the root node of the Branch & Bound tree using a standard cutting-plane generation procedure.
- AGG12: Aggregate formulation (9)-(15) first strengthened by the valid inequalities (39) proposed in section 3 and then by the valid inequalities proposed in [10].
- EXT: Extended formulation proposed by Eppen and Martin in [4] for each item. We use the barrier algorithm embedded in CPLEX to solve the initial linear relaxation of the problem.
- APP0.5: Approximate extended formulation (31)-(38) with $\Delta_i = 0.5 d_{i,1,T}$. We use the barrier algorithm embedded in CPLEX to solve the initial linear relaxation of the problem.

Tables 1 to 4 display the computational results. For each set of 25 instances and each formulation, we provide:
- *Var.* and *Const.*: the number of variables and constraints.
- *#VI1* (resp. *#VI2*): the average number of valid inequalities from the family proposed in [10] (resp. proposed in section 3 of the present paper) added to the formulation by the cutting-plane generation procedure.
- *#Feas* (resp. *#Opt*): the number of instances out of the corresponding 25 instances for which a feasible solution (resp. a guaranteed optimal solution) could be found within 30 minutes of computation.
- Gap_0: the integrality gap, i.e. the relative difference between the lower bound provided by the linear relaxation of the problem and the value of an optimal solution. For formulations AGG1, AGG2 and AGG12, we consider the lower bound obtained after the cutting-plane generation procedure has stopped.
- $\#Nodes$: the average number of nodes explored by the Branch & Bound procedure before a guaranteed optimal solution is found or the computation time limit is reached.
- CPU_{IP}: the average computation time in seconds required to find a guaranteed optimal solution. If one could not be found, we use the computation time limit of 1800 seconds.

- *Gap*: the average relative gap value obtained after 30 minutes of computation between the best integer solution and the best lower bound found.

4.3 Comparison of formulations AGG1, AGG2 and AGG12

We first discuss the results obtained with formulations AGG1 and AGG2 (see Tables 1-4). These results show that the proposed valid inequalities (39) are more efficient at strengthening the aggregate formulation than the ones previously proposed in [10]. Namely, as can be seen e.g. for set A instances:
- the reduction of the initial integrality gap (Gap_0) is greater with formulation AGG2 than with formulation AGG1 (2% vs 13%).
- the number of cutting-planes generated at the root node of the Branch & Bound tree is lower with formulation AGG2 than with formulation AGG1 (1951 vs 8463).
- the average computation time is significantly lower with formulation AGG2 than with formulation AGG1 (17s vs 594s).
Moreover, results from Tables 1 to 3 also suggest that, in general, combining both families of valid inequalities (formulation AGG12) does not lead to shorter computation times. A noticeable exception can be found for the case where a large number of resources is involved (see Table 4). Specifically for set D instances, the average computation time is reduced from 711s to 636s by using AGG12.

4.4 Comparison of formulations AGG2, EXT and APP0.5

We now compare the strengthened aggregate formulation AGG2 with the extended formulation EXT. Results from Table 1 show that formulation EXT is more efficient than formulation AGG2 at solving small instances. Namely, the average computation time is reduced from 17s with formulation AGG2 to 6s with formulation EXT for set A instances.

However, formulation AGG2 is more efficient at solving large instances (see Tables 2-4). Indeed, the number of instances for which a feasible solution could be found within the computation time limit is higher with formulation AGG2 than with formulation EXT (75 vs 56 instances). This can be explained by the fact that, even if formulation EXT provides lower bounds of good quality, its size significantly increases with the problem dimensions, especially with the number of periods and/or resources. As a consequence, solving its linear relaxation is computationally demanding and only a limited number of nodes can be explored before the time limit is reached.

Moreover, the use of the approximate formulation APP0.5 does not improve the results obtained with formulation EXT with respect to the number of instances for which a feasible solution could be found within the computation time limit. This may be explained by the fact that aggregating part of the nodes and arcs in the corresponding network leads to a reduction in the size of the formulation but also to a decrease in the quality of the lower bounds provided by the linear relaxation of the problem. This is due to the fact that equalities $Z_t = \sum_{\alpha,\gamma}(\gamma - \beta)^+ u(t,\alpha,\beta,\gamma)$ used in the extended formulation proposed by [4] to evaluate startup costs have to be replaced by the weaker inequalities (35)-(36) in the approximate extended formulation.

4.5 Mathematical programming-based heuristic

Finally, we devised a simple heuristic as a first attempt to combine the advantages of formulations EXT and AGG2, namely the quality of the lower bound provided by EXT and the quality of the feasible solutions found by AGG2. More precisely, we implemented the following procedure:

1. We solve the linear relaxation of the problem using extended formulation EXT or approximate extended formulation APP0.5.
2. We look for the variables Y_{it} with an integer non-negative value in the optimal continuous solution and fix them to this value.
3. We solve the resulting problem using formulation AGG2.

	Var.	Const.	VI1	VI2	Feas/Opt	$\text{Gap}_0(\%)$	Nodes	$\text{CPU}_{\text{IP}}(\text{s})$	Gap(%)
AGG	1550	2050	_	_	25/0	45	4.5e6	1800	14
AGG1	1550	2050	8463	_	25/22	13	6981	594	0.4
AGG2	1550	2050	_	1951	25/25	2	25	17	0
AGG12	1550	2050	4130	1951	25/25	1.6	118	49	0
EXT	16449	7666	_	_	25/25	0.3	1	6	0
APP0.5	13096	6574	_	_	25/25	1	5	15	0

Table 1: Results for set A instances: $I = 10, T = 50, K = 2$

	Var.	Const.	VI1	VI2	Feas/Opt	$\text{Gap}_0(\%)$	Nodes	$\text{CPU}_{\text{IP}}(\text{s})$	Gap(%)
AGG1*	4650	6150	_	_	_	_	_	_	_
AGG2	4650	6150	_	17546	25/5	3	1287	1541	2.2
AGG12*	4650	6150	_	_	_	_	_	_	_
EXT	155790	59014	_	_	16/10	1.5	20	1491	0.5
APP0.5	129812	50501	_	_	16/10	2	10	1353	0.6

* Formulations AGG1 and AGG12 could not be solved due to exceeded memory capacity limitations.

Table 2: Results for set B instances: $I = 10, T = 150, K = 2$

Table 5 show the computational results obtained with this heuristic for the instances of sets B and D. We display:
- *Formulation*: the formulation (EXT or APP0.5) used to solve the initial linear relaxation.
- *NVfix*: the average number of variables Y_{it} fixed at step 2 of the heuristic.
- *#Feas*: the number of instances out of the corresponding 25 instances for which a feasible solution could be found within the computation time limit of 600s.
- CPU_{tot}: the average total computation time in seconds before the heuristic stops or before the computation time limit is reached.
- Gap_{heur}: the average relative gap between the lower bound provided by the linear relaxation of formulation EXT or APP0.5 and the best integer solution found.

Results from table 5 show that we are able to obtain with the proposed heuristic better integer solutions within a shorter computation time for set B instances. Namely, the average gap after 10 minutes of computation (Gap_{heur}=1.1%) is smaller than the gap obtained with formulation AGG2 after 30 minutes of computation (Gap=2.2%). However, no improvement was obtained for set D instances.

Finally we observe that the inequalities and reformulations presented above can be used to treat problems with sequence-dependent changeover costs in which constraints (11)-(15) are replaced by (22)-(26).

References

[1] G. Belvaux, L.A. Wolsey, Modelling practical lot-sizing problems as mixed-integer programs, Management Science 47, 993-1007 (2001).

	Var.	Const.	VI1	VI2	Feas/Opt	$\text{Gap}_0(\%)$	Nodes	$\text{CPU}_{\text{IP}}(\text{s})$	Gap(%)
AGG1	3850	5200	15159	_	25/14	7	3360	817	0.8
AGG2	3850	5200	_	2070	25/25	0.7	128	17	0
AGG12	3850	5200	6432	2070	25/25	0.5	159	77	0
EXT	16777	11233	_	_	25/25	0.3	8	17	0
APP0.5	12822	9327	_	_	25/25	7	37	28	0

Table 3: Results for set C instances: $I = 25, T = 50, K = 2$

	Var.	Const.	VI1	VI2	Feas/Opt	$\text{Gap}_0(\%)$	Nodes	$\text{CPU}_{\text{IP}}(\text{s})$	Gap(%)
AGG1	1550	2050	5937	_	25/14	5	25246	777	0.3
AGG2	1550	2050	_	814	25/18	6	97293	711	0.3
AGG12	1550	2050	4923	814	25/19	4	13266	636	0.2
EXT	831214	96775	_	_	15/12	1	5	1185	1
APP0.5	619493	68242	_	_	13/11	1	3	1240	0.5

Table 4: Results for set D instances: $I = 10, T = 50, K = 10$

	Formulation	NVfix	Feas	$\text{CPU}_{\text{tot}}(\text{s})$	$\text{Gap}_{\text{heur}}(\%)$
set B	EXT	159	25	298	1.1
set D	APP0.5	235	14	687	0.5

Table 5: Results obtained with the heuristic for sets B and D instances

[2] M. Constantino, A polyhedral approach to production planning models: startup costs and times and lower bounds on production. PhD thesis (1995), Université catholique de Louvain, Belgique.

[3] A. Drexl, A. Kimms, Lot sizing and scheduling - Survey and extensions, European Journal of Operational Research 9, 221-235 (1997).

[4] G.D Eppen, R.K. Martin, Solving multi-item capacitated lot-sizing problems using variable redefinition, Operations Research 35, 832-848 (1987).

[5] C. Gicquel, M. Minoux, Y. Dallery, A tight MIP formulation for the discrete lot sizing and scheduling problem with parallel resources, International Conference on Computers and Industrial Engineering, Université Technologique de Troyes, Troyes, France, 2009.

[6] R. Jans, Z. Degraeve, Meta-heuristics for dynamic lot sizing: a review and comparison of solution approaches, European Journal of Operational Research 177, 1855-1875 (2007).

[7] L.S. Lasdon, Optimization theory for large systems, Macmillan, New York, 1970.

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

[9] M. Salomon, M. Solomon, L.N. van Wassenhove, Y. Dumas, S. Dauzère-Peres, Solving the discrete lotsizing and scheduling problem with sequence dependent set-up costs and set-up times using the travelling salesman problem with time windows, European Journal of Operational Research 100, 494-513 (1997).

[10] F. Vanderbeck and L.A. Wolsey, Valid inequalities for the Lasdon-Terjung production model, Journal of the Operational Research Society 43, 435-441 (1992) .

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

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