To appear in the *International Journal of Production Research*
Vol. 00, No. 00, 00 Month 20XX, 1–18

A parallel machine lot-sizing and scheduling problem with a secondary resource and cumulative demand

Murat Güngör[a]*, Ali Tamer Ünal[a] and Z. Caner Taşkın[a]

[a]*Industrial Engineering Department, Boğaziçi University, 34342, Istanbul, Turkey*

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

We investigate a parallel machine multi-item lot-sizing and scheduling problem with a secondary resource, in which demands are given for the entire planning horizon rather than for every single period. All-or-nothing assumption of the discrete lot-sizing and scheduling problem is valid so that a machine is either idle or works at full capacity in a period. The objective is to minimise the number of setups and teardowns. We prove that the problem is NP-hard, and present two equivalent formulations. We show some properties of the optimal objective value, give optimality conditions, and suggest a heuristic algorithm. We discuss and formulate two possible extensions related to real-life applications. Finally, we carry out computational experiments to compare the two formulations, to determine the effect of our proposed modeling improvements on solution performance, and to test the quality of our heuristic.

Keywords: lot-sizing; scheduling; parallel machines; mixed integer linear programming; secondary resource; cumulative demand

1. Introduction

Lot-sizing and scheduling is a practical problem arising in many production environments. Although there are similarities between problems originating in different settings, each production process has its own characteristics. Motivated by a real-life application, we address in this paper a novel parallel machine multi-item lot-sizing and scheduling problem with a secondary resource. The novelty lies in the assumption that demands are given for the entire planning horizon rather than for every single period.

The problem can be described as follows: A number of items are to be produced on identical parallel machines. In order to produce an item, a piece of equipment (secondary resource) specific to that item must be installed in the machine. The planning horizon is divided into buckets for which all-or-nothing assumption is valid; that is to say, a machine is either idle or works at full capacity in a period. Total demand quantity of each item is known in advance. A constant cost is incurred for every setup and teardown (mount and dismount) of equipments on machines, and the objective is to minimise the total number of setups and teardowns.

Our motivation to study this problem comes from aluminum alloy wheel production, which actually takes place in a flow shop consisting mainly of casting, heat treatment, painting, and packaging stages. Planning of the foundry leads essentially to the problem described above. In particular, items are the wheels to be cast, and equipments are the molds used together with casting machines. Although planning time buckets are shifts, due to a common practice in automotive industry, daily shipment quantities to the car manufacturers are managed based on maintaining a weekly delivered quantity budget rather than fulfilling concrete day-based or shift-based due dates

*Corresponding author. Email: murat.gungor@boun.edu.tr

and delivery quantities. In other words, demand information is cumulative. At the beginning of each week, all molds are dismounted, and one shift is devoted to cleaning of machines for quality assurance. Thus, different weeks are somewhat independent from each other and constitute natural planning horizons by themselves.

As the demand to be fulfilled is cumulative, there is no need for inventory balance equations. The main goals are to decrease the number of changeovers, and to increase thereby the utilisation of resources. Therefore, inventory-related costs are not taken into account. In other words, excess production is not penalised unlike most lot-sizing models in the literature. Number of molds is not restricted. Setup and teardown times are ignored because they are more or less constant and do not constitute a large portion of the bucket capacity. In spite of all these simplifying assumptions, the problem turns out to be NP-hard, as we shall prove.

Lot-sizing and scheduling is a broad subject, which has been studied extensively in the literature. Besides books addressing different aspects of the topic (Jordan 1996; Kimms 1997; Quadt 2004; Suerie 2005; Pochet and Wolsey 2006; Seeanner 2013), there are several comprehensive surveys (Drexl and Kimms 1997; Karimi, Fatemi Ghomi, and Wilson 2003; Zhu and Wilhelm 2006; Quadt and Kuhn 2007; Jans and Degraeve 2008; Buschkühl et al. 2010; Copil et al. 2017).

Most models assume that the planning horizon is divided into buckets. We shall refer to them as discrete-time formulations. These are usually derived from a few basic models, namely discrete lot-sizing and scheduling problem (DLSP), continuous setup lot-sizing problem (CSLP), and proportional lot-sizing and scheduling problem (PLSP). Purely continuous formulations are rarely proposed. However, hybrid formulations, namely variants of general lot-sizing and scheduling problem (GLSP) and capacitated lot-sizing with sequence-dependent setups (CLSD), are quite popular. See Drexl and Kimms (1997) and Copil et al. (2017) for a thorough discussion of the basic models just mentioned.

Lasdon and Terjung (1971) investigate for a major tire manufacturer a problem closely related to ours. They consider multiple products with dynamic demand to be produced on identical parallel machines. In order to produce an item, a die must be installed in the machine, and the number of dies available in each period is limited. Their model is based on DLSP. The main decision variables are the numbers of machines to be used for production of an item in a period. For the Lasdon-Terjung model, Eppen and Martin (1987) give an extended reformulation, whereas Vanderbeck and Wolsey (1992) propose valid inequalities involving only the natural variables. Gicquel, Minoux, and Dallery (2011) discuss exact solution approaches for DLSP with identical parallel machines. Gicquel, Wolsey, and Minoux (2012) introduce a disaggregate formulation in which the machines are indexed explicitly. In their terminology, the Lasdon-Terjung model is an aggregate formulation. They prove that the two formulations are equivalent, and show how a family of inequalities developed by van Eijl and van Hoesel (1997) can be adapted to give valid inequalities for the aggregate formulation. Jans and Degraeve (2004) present an industrial extension of DLSP for a tire manufacturer. Quadt and Kuhn (2009) make use of aggregate variables to solve a variant of the capacitated lot-sizing problem with linked lot sizes. Kaczmarczyk (2011) applies the same idea to PLSP with identical parallel machines. Fiorotto and de Araujo (2014) develop a Lagrangian heuristic for unrelated parallel machines based on the strategy of Eppen and Martin (1987).

The problem we consider is essentially different from those studied in the articles cited above because of its cumulative demand structure, and to the best of our knowledge, it has not been addressed in the literature before. On the one hand, it can be classified within DLSP since all-or-nothing assumption is valid. On the other hand, it can be seen as a GLSP model with only one macroperiod made up of a number of fixed-length microperiods. Theoretical and empirical results concerning this new problem make up the main contributions of our paper.

Outline of the paper is as follows: in §2 we prove that the problem is NP-hard, present two mixed-integer programming formulations, and show their equivalence. Then we give some properties of the optimal objective value in §3.2, and prove several optimality conditions in §3.3. Next, in §3.4 we suggest a heuristic algorithm. Afterwards, we discuss and formulate two possible extensions to

the problem in §4. Finally, we present a detailed computational study in §5.

2. Problem definition

Let I, M, T denote the number of secondary resource types, machines, and time periods, respectively, and i, j, t the indices thereof. All machines are identical. Let $\overline{x_i}$ be the demand as average number of secondary resources of type i to be used in a period. In other words, if d_i is the total demand for items of type i, and c_i the number of items of type i a machine can produce in one period, then $\overline{x_i} = \frac{1}{T} \times \frac{d_i}{c_i}$. One and only one type of secondary resource can be installed in a machine during a period. The problem is to find an assignment of secondary resources to machines throughout the planning horizon such that all demands are satisfied and the total number of setups and teardowns is minimised. By convention, setups in the first period and teardowns in the last period are not counted. We shall see in §2.3 that one can assume $T\,\overline{x_i} \in \mathbb{Z}_{\geq 0}$ for all i without loss of generality, and under this assumption the problem is feasible if and only if $\sum_i \overline{x_i} \leq M$.

2.1 Complexity

We prove that the lot-sizing and scheduling problem defined above is NP-hard by reduction from the well-known PARTITION problem.

Theorem 1. *The problem is NP-hard.*

Proof. Let $a_1, \ldots, a_m$ and $b = \frac{1}{2}\sum_{k=1}^{m} a_k$ be positive integers. PARTITION is the following problem: Can we divide the collection $a_1, \ldots, a_m$ into two subsets such that the numbers in each subset sum up to b? Now consider an instance of our problem where $M = 2$, $I = m$, $T = b$, and $\overline{x_i} = \frac{a_i}{T}$. We will show that PARTITION has a positive answer if and only if the optimal objective value z^* is less than or equal to $2I - 4$ for this instance. Assume there exist S_1, S_2 such that $S_1 \cup S_2 = \{1, \ldots, m\}$, $S_1 \cap S_2 = \varnothing$ and $b = \sum_{k \in S_1} a_k = \sum_{k \in S_2} a_k$. Assign the secondary resources in S_j to machine j ($j = 1, 2$). Then total number of setups and teardowns is $(2|S_1| - 2) + (2|S_2| - 2) = 2I - 4$, whence $z^* \leq 2I - 4$. Conversely, suppose $z^* \leq 2I - 4$, and consider an optimal schedule. Let w_j be the number of different secondary resource types that appear in machine j. At least $2w_j - 2$ setups and teardowns must take place in j. Moreover, $w_1 + w_2 \geq I$. So $2I - 4 \geq z^* \geq (2w_1 - 2) + (2w_2 - 2) \geq 2I - 4$. Therefore $z^* = 2I - 4$ and $w_1 + w_2 = I$, implying that PARTITION has an affirmative answer. □

2.2 Disaggregate and aggregate formulations

It is natural to keep an explicit account of secondary resource–machine assignments when modeling the problem at hand. This is the essential feature of the disaggregate formulation F1 to be given below. Let y_{ijt} be a binary variable defined as 1 if in period t machine j is set up for a secondary resource of type i, and as 0 otherwise. Define another binary variable z^+_{ijt} as 1 if a setup for a secondary resource of type i is performed on machine j in transition from period $t-1$ to t, and as

0 otherwise; similarly define z^-_{ijt} for teardowns ($t > 1$).

$$\text{F1:} \quad \min \quad \sum_{i,j,t} (z^+_{ijt} + z^-_{ijt}) \tag{1a}$$

$$\text{s.t.} \quad \sum_{i} y_{ijt} \leq 1 \qquad \text{for all } j,t \tag{1b}$$

$$\frac{1}{T} \sum_{j,t} y_{ijt} \geq \overline{x_i} \qquad \text{for all } i \tag{1c}$$

$$y_{ijt} - y_{ij,t-1} = z^+_{ijt} - z^-_{ijt} \qquad \text{for all } i,j,t \tag{1d}$$

$$z^+_{ijt}, z^-_{ijt} \geq 0 \qquad \text{for all } i,j,t \tag{1e}$$

$$y_{ijt} \in \{0,1\} \qquad \text{for all } i,j,t \tag{1f}$$

The first constraint (1b), together with (1f), states that a machine is either idle or works at full capacity in a period (being set up for one and only one secondary resource). This is the so-called all-or-nothing assumption. The next set of inequalities (1c) guarantees that enough items are produced of each type. Constraint (1d) holds an account of setups and teardowns. Expressions involving z^+_{ijt}, z^-_{ijt} are subject to $t > 1$. Note that z^+_{ijt} and z^-_{ijt} can be relaxed as continuous variables in view of the objective since the y_{ijt} are binary.

The problem admits a simpler formulation – to be called F2 – since the machines in question are identical. The idea, which goes back to Lasdon and Terjung (1971), is to keep track only of the total number of secondary resources used for each type in a period. This determines the secondary resource–machine assignments implicitly. Let x_{it} be the total number of secondary resources of type i used in period t. Let s^+_{it} be the number of setups required for type i in transition from period $t-1$ to t; similarly define s^-_{it} for teardowns ($t > 1$). A complete list of indices, parameters, and decision variables for F1 and F2 can be found in Table 1.

Table 1. Indices, parameters, and decision variables for the two formulations F1 and F2.

Symbol(s)	Explanation
i,j,t	indices for secondary resource types, machines, and time periods
M,T	number of machines and time periods
$\overline{x_i}$	demand as average number of secondary resources of type i to be used in a period
x_{it}	number of secondary resources of type i used in period t
y_{ijt}	1 if machine j is set up for a secondary resource of type i in period t, and 0 otherwise
s^+_{it} /s^-_{it}	number of setups/teardowns required for secondary resources of type i in transition from period $t-1$ to t
z^+_{ijt}/z^-_{ijt}	1 if a setup/teardown for a secondary resource of type i is performed on machine j in transition from period $t-1$ to t, and 0 otherwise

The aggregate formulation F2 is given below. Expressions involving s^+_{it} and s^-_{it} are subject to $t > 1$. The set of nonnegative integers is designated by $\mathbb{Z}_{\geq 0}$.

$$\text{F2:} \quad \min \quad \sum_{i,t} (s^+_{it} + s^-_{it}) \tag{2a}$$

$$\text{s.t.} \quad \sum_{i} x_{it} \leq M \qquad \text{for all } t \tag{2b}$$

$$\frac{1}{T} \sum_{t} x_{it} \geq \overline{x_i} \qquad \text{for all } i \tag{2c}$$

$$x_{it} - x_{i,t-1} = s^+_{it} - s^-_{it} \qquad \text{for all } i,t \tag{2d}$$

$$s^+_{it}, s^-_{it} \geq 0 \qquad \text{for all } i,t \tag{2e}$$

$$x_{it} \in \mathbb{Z}_{\geq 0} \qquad \text{for all } i,t \tag{2f}$$

The first constraint (2b) ensures that the total number of secondary resources used in a period cannot exceed the number of machines. The second one (2c) is the demand fulfillment restriction. It is appropriate to call (2d) as 'setup balance equations'. Note that the continuous variables s_{it}^+, s_{it}^- assume integer values in an optimal solution by virtue of the objective since the x_{it} are integer. Note also that M is a natural upper bound for all decision variables.

2.3 Equivalence of the two formulations

Lemma 1. *The inequalities* $\sum_{j,t} y_{ijt} \geq \lceil T\,\overline{x_i} \rceil$ *and* $\sum_t x_{it} \geq \lceil T\,\overline{x_i} \rceil$ *are valid for* F1 *and* F2*, respectively, for all* i*.*

Proof. For any feasible solution of F1, the y_{ijt} will be integers, so the sums $\sum_{j,t} y_{ijt}$ will be integers greater than or equal to $T\,\overline{x_i}$. Consequently, the inequality $\sum_{j,t} y_{ijt} \geq \lceil T\,\overline{x_i} \rceil$ is satisfied for all i. The argument is similar for F2. □

From this point on, we assume without loss of generality that $T\,\overline{x_i} \in \mathbb{Z}_{\geq 0}$ for all i.

Lemma 2. *The following statements are equivalent:*
(i) F1 *is feasible.*
(ii) F2 *is feasible.*
(iii) $\sum_i \overline{x_i} \leq M$*.*

Proof. If $y = (y_{ijt})$ is a feasible solution of F1, then $x = (x_{it})$ defined as $x_{it} = \sum_j y_{ijt}$ is a feasible solution of F2 (note that y_{ijt} and x_{it} uniquely determine z_{ijt} and s_{it}). Indeed, according to this definition, constraints (1b) and (1c) imply (2b) and (2c), respectively. If F2 is feasible, then $\overline{x_i} \leq \frac{1}{T}\sum_t x_{it}$ and $\sum_i x_{it} \leq M$, implying $\sum_i \overline{x_i} \leq \frac{1}{T}\sum_i \sum_t x_{it} \leq \frac{1}{T}\sum_t M = M$. Finally, if $\sum_i \overline{x_i} \leq M$, then $\sum_i T\,\overline{x_i} \leq TM$. Hence, the TM slots in the planning horizon are sufficient to cover all demand (the assumption $T\,\overline{x_i} \in \mathbb{Z}$ is crucial here), implying that F1 is feasible. □

Let S_1 and S_2 be the feasible solution spaces of F1 and F2, respectively. For each $y \in S_1$, let $\alpha(y) = x$ be defined by $x_{it} = \sum_j y_{ijt}$. Then $\alpha(y) \in S_2$ as we have seen in the proof of Lemma 2. So α is a many-to-one mapping from S_1 to S_2. It is not reasonable to perform a setup for a secondary resource which has just been torn down from another machine in the immediately preceding period. We designate by S_1' the schedules reasonable in this sense. Thus, S_1' consists of those $y \in S_1$ for which there exists no pair of machines $j_1 \neq j_2$ such that $z_{ij_1t}^+ = 1$ and $z_{ij_2t}^- = 1$. Let us denote by $z_1(y)$ and $z_2(x)$ the objective function values of F1 and F2 evaluated at $y \in S_1$ and $x \in S_2$.

Lemma 3. *(i)* $z_1(y) \geq z_2(\alpha(y))$ *for all* $y \in S_1$*.*
(ii) $z_1(y) = z_2(\alpha(y))$ *for all* $y \in S_1'$*.*
(iii) For all $x \in S_2$*, there exists* $y \in S_1'$ *such that* $\alpha(y) = x$*.*

Proof. Let $y \in S_1$ and $x := \alpha(y)$. Summing up (1d) over j, we get $\sum_j y_{ijt} - \sum_j y_{ij,t-1} = \sum_j z_{ijt}^+ - \sum_j z_{ijt}^-$ so that $s_{it}^+ - s_{it}^- = \sum_j z_{ijt}^+ - \sum_j z_{ijt}^-$ by (2d). In view of (1a) and (2a), the inequality $s_{it}^+ + s_{it}^- \leq \sum_j z_{ijt}^+ + \sum_j z_{ijt}^-$ always holds, two sides being equal if and only if $y \in S_1'$. This proves (i) and (ii). Now let $x \in S_2$. We define $y \in S_1$ as follows: For the first period, make an arbitrary assignment of secondary resources x_{i1} to machines. Then, for each subsequent period, first perform teardowns for all i by choosing arbitrarily s_{it}^--many machines in which i has been installed in the previous period, and then perform setups for all i by choosing s_{it}^+-many empty machines. Thus $y \in S_1'$ and $\alpha(y) = x$. □

Algorithm 1 outlines the disaggregation procedure described in the proof of Lemma 3.

Algorithm 1 Disaggregation.

1: assign the x_{i1} arbitrarily to machines
2: **for** $t = 2$ **to** T **do**
3: **for all** i **do**
4: choose arbitrarily s^-_{it}-many machines in which secondary resource i has been installed in period $t-1$, and perform a teardown
5: **for all** i **do**
6: choose arbitrarily s^+_{it}-many empty machines, perform a setup of secondary resource i, and update the empty machine list

Theorem 2. *Formulations* F1 *and* F2 *are equivalent in the sense that feasibility of one implies that of the other, in which case the optimal objective values are the same.*

Proof. We have already shown the assertion about feasibility in Lemma 2. Now let z_1^* and z_2^* be the optimal objective values of F1 and F2. The first part of Lemma 3 implies $z_1^* \geq z_2^*$, whereas the other two parts imply $z_1^* \leq z_2^*$. Therefore, $z_1^* = z_2^*$ provided that the problem is feasible. □

There is an equivalence between not only the mixed-integer programs but also the linear programming relaxations of the two formulations. We prove this in the next lemma.

Lemma 4. *Linear programming bounds of* F1 *and* F2 *are the same.*

Proof. For any feasible instance of F2, if the integrality constraints are relaxed, setting $x_{it} := \overline{x_i}$ for all i, t yields a solution with zero optimal value. Similarly, setting $\lfloor \overline{x_i} \rfloor$-many y_{ijt} to 1 and one y_{ijt} to $\overline{x_i} - \lfloor \overline{x_i} \rfloor$ for all i, t, avoiding possible conflicts on machines, gives a solution with zero optimal value for the relaxation of F1. □

In case some machines become temporarily unavailable within the planning horizon, F2 is no longer a valid representation of the problem even if M is replaced by M_t in the formulation. The simplest example demonstrating this fact consists of one secondary resource with $\overline{x_i} = 1$, two periods, and two machines, where only machine j is available in period j. Then F2 yields 0 as optimal value whereas it is actually 2. Let us note that an aggregate formulation is not possible when the machines are nonidentical.

3. Algorithmic and modeling improvements

3.1 Symmetry-breaking constraints

The disaggregate formulation F1 has about M times as many variables and constraints as the aggregate formulation F2, with many alternative symmetric solutions burdening the model (Sherali and Smith 2001). For the lot-sizing problem on identical parallel machines, Jans (2009) proposes eight families of symmetry-breaking constraints. The last three of these involve setup times or costs, and are not applicable in our case, whereas the first five can be readily adapted to F1. In our notation, the second family, denoted (SBC2) by Jans (2009), reads

$$\sum_{i=1}^{I} 2^{I-i} y_{ijt} \geq \sum_{i=1}^{I} 2^{I-i} y_{i,j+1,t} \quad \text{for all } 1 \leq j \leq M-1 \text{ and } t. \tag{3}$$

For example, when $I = 3$, this can be written explicitly as $4y_{11t}+2y_{21t}+y_{31t} \leq 4y_{12t}+2y_{22t}+y_{32t} \leq 4y_{13t} + 2y_{23t} + y_{33t} \leq \cdots$ for all t. Thus, if item 1 is produced in period t, it must be on the first machine(s), because $y_{1jt} = 1$ implies $y_{1j't} = 1$ for all $j' < j$; similarly for items 2 and 3. In general,

inequalities (3) impose a unique lexicographic ordering on machines in each period. This is also true of (SBC1) that contains (3) as a subset. The families (SBC3–5) impose a partial ordering only. The computational study in Jans (2009) shows that (SBC2) turns out to be one of the most efficient families, so we use (3) as a means of improving F1.

3.2 Properties of optimal objective value

We denote the optimal objective value by z^*.

Lemma 5. *z^* is zero if and only if $\sum_i \lceil \overline{x_i} \rceil \leq M$.*

Proof. If $\sum_i \lceil \overline{x_i} \rceil \leq M$, setting $x_{it} := \lceil \overline{x_i} \rceil$ yields a feasible solution with zero objective value. Conversely, if $z^* = 0$, the assignment in any period determines the schedule for the entire planning horizon for each machine; more precisely, $y_{ijt} = 1$ implies $y_{ijt'} = 1$ for all t'. It follows that $M \geq \sum_i \lceil \overline{x_i} \rceil$ since in any feasible solution there must exist a period in which at least $\lceil \overline{x_i} \rceil$-many distinct machines are dedicated for secondary resource i. □

Lemma 6. *There exists an optimal schedule such that no machine is kept idle in any period.*

Proof. Any idle periods on a machine can be filled up by prolonging the production in neighbouring periods, which does not worsen the objective value because excess production is not penalised. □

Let us denote by S the collection of feasible solutions $y \in S_1$ such that $\sum_i y_{ijt} = 1$ for all j, t. We shall refer to such solutions as full schedules. According to Lemma 6, there always exists a full optimal schedule.

Corollary 1. *z^* must be an even integer.*

Proof. The objective value associated with any full schedule must be an even integer since every teardown is necessarily paired with a setup there. The result follows from Lemma 6. □

Lemma 7. *$z^* \leq 2I - 2$.*

Proof. Write $T\,\overline{x_i} = T\,\lfloor \overline{x_i} \rfloor + r_i$, where r_i is an integer such that $0 \leq r_i \leq T - 1$. Allocate, for each i, $T\,\lfloor \overline{x_i} \rfloor$-many secondary resources starting with the first machine. These allocations do not contribute to the objective. Next, fill up the remaining machines with the r_i-many secondary resources one by one, in ascending order with respect to the machine and time indices. At this stage, each secondary resource type increases the objective value by at most two except the first and the last types for which the contribution is at most one. The result follows. □

3.3 Optimality conditions

Due to Theorem 1 the problem is inherently difficult, and one cannot expect to solve it to optimality for large instances in reasonable time. Still, its mixed-integer formulation can be improved by means of linear constraints that do not change the optimal objective value when added to the formulation. In this subsection, we prove several optimality conditions by means of which such constraints can be obtained.

Given a solution, let $b(j, t, l)$ represent the block of assignments on machine j in the l consecutive periods $t, t+1, \ldots, t+l-1$. For $y \in S$, consider the operation of exchanging two blocks $b(j_1, t_1, l_1)$ and $b(j_2, t_2, l_2)$ to obtain a new solution y'. We assume that the blocks are of the same length ($l_1 = l_2$) unless they are on the same machine ($j_1 = j_2$) and adjacent ($t_1 + l_1 = t_2$ or $t_2 + l_2 = t_1$). So y' is well-defined. It will be convenient to define z_{ijt} to be z^+_{ijt} or $-z^-_{ijt}$ according as $z^+_{ijt} \geq 0$ or $z^-_{ijt} > 0$. Let $c_{jt}(y)$ be 1 if $z_{ijt} \neq 0$ for some i, and 0 otherwise. In other words, $c_{jt}(y)$ is 1 if there is

a changeover on machine j from period $t-1$ to t. Now we can express in a simple way the difference $d := z_1(y') - z_1(y)$ in the objective function value (1a): the equality $z_1(y) = 2\sum_{j,t} c_{jt}(y)$ holds for all $y \in S$; consequently, $d = 2\sum_{j,t}(c_{jt}(y') - c_{jt}(y)) = 2\sum_{j,t} d_{jt}$ where $d_{jt} := c_{jt}(y') - c_{jt}(y)$. For convenience, we define $c_{j1}(y)$ and $c_{j,T+1}(y)$ as 0, too.

Lemma 8. *(i) If $j_1 \neq j_2$ or the blocks are nonadjacent, then $\frac{1}{2}d = d_{j_1 t_1} + d_{j_1,t_1+l} + d_{j_2 t_2} + d_{j_2,t_2+l}$ where $l := l_1 = l_2$.*

(ii) If $j_1 = j_2 =: j$ and $t_1 + l_1 = t_2$, then $\frac{1}{2}d = d_{jt_1} + d_{j,t_2+l_2} + c_{j,t_1+l_2}(y') - c_{j,t_1+l_1}(y)$. An analogous equation holds for the case $t_2 + l_2 = t_1$.

Proof. When two blocks are exchanged, all d_{jt} are zero except those related to the borders. Writing them out explicitly, the equalities follow. □

Theorem 3. *There exists an optimal schedule such that*

$$x_{it} \geq \lfloor \overline{x_i} \rfloor \quad \text{for all } i, t. \tag{4}$$

Proof. Fix a secondary resource type i arbitrarily. It is enough to find an optimal $x \in S_2$ satisfying $x_{it} \geq \lfloor \overline{x_i} \rfloor$ for all t. Indeed, rewriting the formulation in terms of $x_{it} - \lfloor \overline{x_i} \rfloor$ instead of x_{it}, we obtain a problem of the same form with strictly smaller M, and the result follows by repeating the same argument.

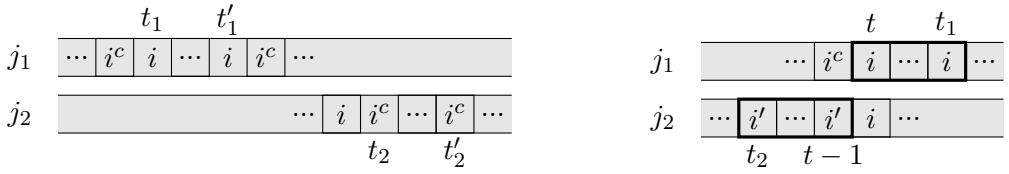
Let $x \in S_2$ be an optimal solution, and suppose $x_{it_0} < \lfloor \overline{x_i} \rfloor$ for some t_0. Take a schedule $y \in S_1'$ such that $\alpha(y) = x$ (Lemma 3). Since $z^* \neq 0$, y must be full (Lemma 6). Let $x_i^{\max} := \max_t\{x_{it}\}$, $x_i^{\min} := \min_t\{x_{it}\}$, and let n_i be the number of periods t for which $x_{it} = x_i^{\max}$ or $x_{it} = x_i^{\min}$. Consider the ordered pair $w_i(x) := (x_i^{\max} - x_i^{\min}, n_i)$. It is sufficient to show the following: as long as the inequality $x_i^{\max} - x_i^{\min} \geq 2$ is satisfied, one can obtain by exchanging suitable blocks an alternative optimal $y' \in S_1$ such that $w_i(x')$ is lexicographically smaller than $w_i(x)$, where $x' := \alpha(y')$.

Consider a list $t_1, t_1 + 1, \ldots, t_1 + l_1 - 1 =: t_1'$ of consecutive periods t for which $x_{it} = x_i^{\max}$, maximal in the sense that $x_{i,t_1-1} < x_i^{\max}$ (unless $t_1 = 1$) and $x_{i,t_1'+1} < x_i^{\max}$ (unless $t_1' = T$). Similarly, consider a list $t_2, t_2 + 1, \ldots, t_2 + l_2 - 1 =: t_2'$ of periods t for which $x_{it} = x_i^{\min}$, maximal in the sense that $x_{i,t_2-1} > x_i^{\min}$ (unless $t_2 = 1$) and $x_{i,t_2'+1} > x_i^{\min}$ (unless $t_2' = T$). We may assume without loss of generality (see Algorithm 1) that there exists a machine j_1 such that $y_{ij_1 t} = 1$ for all $t_1 \leq t \leq t_1'$, $y_{ij_1,t_1-1} = 0$, and $y_{ij_1,t_1'+1} = 0$. Also, there exists a machine j_2 such that $y_{ij_2 t} = 0$ for all $t_2 \leq t \leq t_2'$, and $y_{ij_2,t_2-1} = 1$. In case $t_2 = 1$, we may assume $y_{ij_2,t_2'+1} = 1$ as well.

First, suppose that $j_1 \neq j_2$ (Figure 1(a)) or that the lists are nonadjacent. Let $l := \min\{l_1, l_2\}$. If $l_1 \leq l_2$, exchange the blocks $b(j_1, t_1, l)$ and $b(j_2, t_2, l)$ to obtain y'. The change $d := z_1(y') - z_1(y)$ in the objective function value is given by $\frac{1}{2}d = d_{j_1 t_1} + d_{j_1,t_1+l} + d_{j_2 t_2} + d_{j_2,t_2+l}$ (Lemma 8). Since $d_{j_1 t_1} \leq 0, d_{j_1,t_1+l} \leq 0$, and $d_{j_2 t_2} = -1$, we have $d \leq 0$. So y' is still optimal, and $w_i(x') < w_i(x)$. In case $t_2 = 1$, we shall take $b(j_2, t_2' - l + 1, l)$ as the second block. If $l_1 > l_2$, exchange $b(j_1, t_1, l)$ and $b(j_2, t_2, l)$ as before; however, in this case, $\frac{1}{2}d$ may turn out to be 1 if $y_{ij_2,t_2'+1} = 0$. Nevertheless, if $y_{ij_2,t_2+1} = 0$ there must exist a distinct machine j_3 such that $y_{ij_3 t_2'} = 0$ and $y_{ij_3,t_2'+1} = 1$. Doing a second exchange of blocks $b(j_2, t_2' + 1, T - t_2')$ and $b(j_3, t_2' + 1, T - t_2')$ guarantees that $d \leq 0$.

Next, suppose that $j_1 = j_2 =: j$ and the lists are adjacent. We exchange the two blocks $b(j, t_1, l_1)$ and $b(j, t_2, l_2)$. If $t_2' + 1 = t_1$, then $\frac{1}{2}d = d_{jt_2} + d_{j,t_1+l_1} + c_{j,t_2+l_1}(y') - c_{j,t_2+l_2}(y)$ (Lemma 8). Since $d_{jt_2} = -1, d_{j,t_1+l_1} \leq 0$, and $c_{j,t_2+l_1}(y') = c_{j,t_2+l_2}(y) = 1$, we have $d < 0$. This shows that $t_2' + 1 = t_1$ cannot be true, so $t_1' + 1 = t_2$. Then $\frac{1}{2}d$ may be 1, but after a second exchange as in the preceding paragraph, we obtain a solution for which $d \leq 0$. This completes the proof. □

Corollary 2. *Every instance $(\overline{x_i}, M, T)$ of the problem is equivalent to $(\overline{x_i} - \lfloor \overline{x_i} \rfloor, M - \sum_i \lfloor \overline{x_i} \rfloor, T)$.*



(a) Machines, secondary resources, and time periods in case $j_1 \neq j_2$ (Theorem 3).

(b) Two blocks $b(j_1, t, l)$ and $b(j_2, t_2, l)$ in y to be exchanged (Theorem 4).

Figure 1. Gantt charts that help visualise the proofs of Theorems 3 and 4, where i^c represents any secondary resources other than i.

Proof. By Theorem 3, we can add the inequalities $x_{it} \geq \lfloor \overline{x_i} \rfloor$ to the formulation and still get the same optimal value. Define new variables $x'_{it} := x_{it} - \lfloor \overline{x_i} \rfloor$ for all i, t. Rewriting constraints (2b) and (2c) yields the result. □

Theorem 4. *(i) There exists an optimal schedule such that $s^+_{it} \leq 1$ for all i, t.*
(ii) There exists an optimal schedule such that $s^-_{it} \leq 1$ for all i, t.
(iii) There exists an optimal schedule such that $s^+_{i2} \leq 1, s^-_{i2} \leq 1, s^+_{iT} \leq 1$, and $s^-_{iT} \leq 1$ for all i.

Proof. (i) Fix a secondary resource type i arbitrarily. Let $x \in S_2$ be an optimal solution, and suppose $s^+_{it} > 1$ for some t. Take a schedule $y \in S'_1$ such that $\alpha(y) = x$ (Lemma 3). As $z^* \neq 0$, y must be full (Lemma 6). Since $s^+_{it} > 1$ and $y \in S'_1$, there are at least two machines j_1, j_2 such that $z^+_{ij_1t} = z^+_{ij_2t} = 1$. Let i' be the secondary resource on machine j_2 in period $t-1$, let $t_1 := \max\{\tau \geq t : y_{ij_1\tau} = 1\}$, $t_2 := \min\{\tau \leq t-1 : y_{i'j_2\tau} = 1\}$, and $l := \min\{t_1 - t + 1, t - t_2\}$. Exchange the two blocks $b(j_1, t, l)$ and $b(j_2, t_2, l)$ to obtain a new schedule y' (Figure 1(b)). Then $\frac{1}{2}d = d_{j_1t} + d_{j_1,t_1+1} + d_{j_2t_2} + d_{j_2t}$ (Lemma 8). Here $d_{jt} \leq 0, d_{j_2t} = -1$, and at most one of $d_{j_1,t_1+1}, d_{j_2t_2}$ is 1 so that $d \leq 0$. Hence, y' is still optimal.

Consider the ordered T-tuple $w_i(x) := (x_{i1}, \ldots, x_{iT})$. Clearly, $w_i(x') > w_i(x)$ where $x' := \alpha(y')$. Hence, repeating the same argument we can obtain in a finite number of steps an optimal solution for which $s^+_{it} \leq 1$ for all t. Define s_{it} to be s^+_{it} or $-s^-_{it}$ according as $s^+_{it} \geq 0$ or $s^-_{it} > 0$. Notice that the exchange above increases $s_{i'\tau}$ only for $\tau = t$, and $s_{i't}$ is increased by 2. However, as $s_{i't}$ is initially negative, it cannot exceed 1 after the exchange. Therefore, all secondary resources can be handled one by one until an optimal solution satisfying the assertion is found.

(ii) Analogous to (i).

(iii) The argument is essentially the same as in (i), and makes use of the fact that some terms drop in the expression in Lemma 8 when the endpoints happen to be the first or the last period. □

3.4 Heuristic algorithm

In this subsection, we introduce a heuristic with polynomial time complexity $O((I^3 + M)T)$, producing high-quality feasible solutions (see §5 for details of computational results). The algorithm starts with reducing the problem: for each secondary resource type i, as many as $\lfloor \overline{x_i} \rfloor$ resources are assigned to machines right away. Thus it remains to consider the fractional demands $\overline{x_i} - \lfloor \overline{x_i} \rfloor$ only, with a smaller number of machines. Notice that this reduced problem is equivalent to the original one by Corollary 2. Next, the problem is compressed; that is to say, a machine is allotted for the type with largest (fractional) demand as long as there is enough capacity. Although this step is quite reasonable, we will shortly present an example where it leads to suboptimal solutions. What is left after compression is a fairly tight scheduling problem. At this point, there will surely be at least one setup or teardown for each of the remaining secondary resource types. Therefore, it makes sense to find pairs (if any) whose demands complement each other, and further reduce the problem by assigning every such pair to one of the available machines. Next, we repeat this once again,

taking this time also the slack capacity (if any) into account, and then perform a similar search for possible triplets. Finally, in descending order with respect to demand, we do a straightforward allocation. Algorithm 2 is a pseudocode that outlines the heuristic described. Notice that finding triplets and reduction take $\mathrm{O}(I^3T)$ and $\mathrm{O}(MT)$ steps, respectively, and these two dominate the remaining parts of the algorithm. Hence, the overall complexity is $\mathrm{O}((I^3+M)T)$.

Algorithm 2 Heuristic.

1: *// reduction*
2: $j_{\text{current}} \leftarrow 1$
3: **for all** i **do**
4: **for** $j = j_{\text{current}}$ **to** $j_{\text{current}} + \lfloor \overline{x_i} \rfloor$ **do**
5: $y_{ijt} \leftarrow 1$ for all t
6: $\overline{x_i} \leftarrow \overline{x_i} - \lfloor \overline{x_i} \rfloor$
7: $j_{\text{current}} \leftarrow j_{\text{current}} + \lfloor \overline{x_i} \rfloor$
8: *// compression*
9: $slack \leftarrow M - j_{\text{current}} - \sum \overline{x_i}$
10: **while** $j_{\text{current}} < M$ **do**
11: **if** $slack \geq 1 - \max\{\overline{x_i}\}$ **then**
12: $i \leftarrow \arg\max\{\overline{x_i}\}$
13: $y_{ijt} \leftarrow 1$ for all t
14: $\overline{x_i} \leftarrow 0$
15: $j_{\text{current}} \leftarrow j_{\text{current}} + 1$
16: $slack \leftarrow M - j_{\text{current}} - \sum \overline{x_i}$
17: *// finding pairs*
18: **for** $i = 1$ **to** $I - 1$ **do**
19: **for** $i' = i$ **to** I **do**
20: **if** $\overline{x_i} + \overline{x_{i'}} = 1$ **then**
21: $y_{ijt} \leftarrow 1$ for all $t \leq T\,\overline{x_i}$
22: $y_{i'jt} \leftarrow 1$ for all $t > T\,\overline{x_i}$
23: $\overline{x_i} \leftarrow 0$
24: $\overline{x_{i'}} \leftarrow 0$
25: $j_{\text{current}} \leftarrow j_{\text{current}} + 1$
26: *// finding pairs utilising slack*
27: repeat the previous step (lines 18-25) with the conditional in line 20 relaxed to $\overline{x_i} + \overline{x_{i'}} \geq 1 - slack$, updating *slack* after line 25
28: *// finding triplets*
29: perform a search for triplets analogous to the one for pairs (lines 18-25)
30: *// straightforward allocation*
31: $t_{\text{current}} = 1$
32: **for all** i **do**
33: **if** $\overline{x_i} > 0$ **then**
34: $y_{ijt} \leftarrow 1$ for all $T\,\overline{x_i}$-many periods
35: update t_{current} and (if need be) j_{current}
36: **if** $slack > 0$ **then**
37: prolong the production of the last secondary resource on the last machine

Let us demonstrate the algorithm for a specific instance. Say $T = 10$, $I = 10$, the $\overline{x_i}$ are 1.9, 0.2, 5.8, 1.2, 3.0, 7.7, 3.2, 1.4, 0.4, 2.0, and $M = 27$. After reduction, we are left with eight secondary resource types with demands 0.9, 0.2, 0.8, 0.2, 0.7, 0.2, 0.4, 0.4, and there are only four machines. Note that the slack capacity is $4 - (0.9 + 0.2 + \ldots + 0.4) = 0.2$. Compression allots one machine to 0.9 in full. Thus, the numbers of types and machines are decreased by one, and the new slack is 0.1. Demands 0.2 and 0.8 complement each other. So do 0.2 and 0.7 if slack capacity is utilised. Finally, we detect a triplet 0.2, 0.4, 0.4. Then nothing is left, and the objective function value associated with the resulting schedule is $z = 8$.

When the heuristic is applied to the instance where $T = 10$, $I = 6$, the $\overline{x_i}$ are 0.7, 0.6, 0.6, 0.3, 0.3, 0.2 (so that the $T\,\overline{x_i}$ are 7, 6, 6, 3, 3, 2), and $M = 3$, the compression step allots one machine to the first secondary resource type for all ten periods, and it follows that $z = 8$ (see the Gantt chart in Figure 2 – the numbers on the blocks represent secondary resource types in the order given above). However, the optimal objective value is 6, which can be obtained by assigning the three

appropriate pairs to three machines. This shows that the compression step may lead to suboptimal solutions.

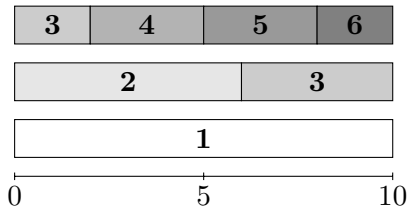


Figure 2. Gantt chart associated with the heuristic solution of the instance for which the compression step in Algorithm 2 leads to suboptimality.

4. Model extensions

In this section we extend our models to take into account two sets of constraints that might be important in real life, namely vertical constraints and minimum lot size constraints.

4.1 Vertical constraints

By vertical constraints, we mean, as in Lasdon and Terjung (1971), the following inequalities limiting by a parameter $\overline{s}$ the number of setups and teardowns performed in between two consecutive periods:

$$\sum_i (s_{it}^+ + s_{it}^-) \leq \overline{s} \quad \text{for all } t > 1. \tag{5}$$

Clearly, the problem is still NP-hard in general: for $\overline{s}$ large enough, it is equivalent to F2. Although Lemma 6 no longer holds, z^* must be an even integer in case of feasibility. Lemma 2 does not hold either: consider for $\overline{s} \leq 3$ the instance where $T = 2$, $I = 4$, the $\overline{x_i}$ are all 0.5, and $M = 2$; we have $\sum_i \overline{x_i} \leq M$, but there exists no feasible solution. Also, the argument used to prove Lemma 7 is invalid for F2 with (5); nevertheless, it is not easy to find a counterexample because the upper bound given there is rather loose. The proofs of Theorems 3 and 4 are invalid too, but computational tests suggest that the assertions may still be true.

For the problem with vertical constraints, we modify the heuristic described in §3.4 slightly as follows: every step of Algorithm 2 is to be applied as is, except for finding pairs, finding pairs utilising slack, and finding triplets. In these three steps, we first check whether inclusion of the pair/triplet into the schedule would violate the vertical constraints. If inclusion is not possible without increasing the number of changeovers, then we simply continue searching for other pairs/triplets. In the modified heuristic, the complexity of finding pairs and triplets remain the same since there are two and six possible sequences for a pair and triplet, respectively. However, the complexity of finding pairs utilising slack is increased from $O(I^2T)$ to $O(I^2T^2)$ so that the overall complexity becomes $O((I^3 + I^2T + M)T)$.

4.2 Minimum lot size constraints

Minimum lot size constraints translate in our problem as a lower bound on the number of periods a secondary resource should remain within a machine after being installed. Such a restriction might stem from shop floor requirements (e.g., materials to be processed may only be stored and

retrieved in certain large batches), or a managerial decision based on the observation that scrap rates gradually decrease as one produces more and more after a new setup.

Minimum lot size restriction can be incorporated into F2 as follows. The idea is to couple each teardown with a setup, and make sure that every teardown is associated with a setup which is performed at least m periods earlier, where m stands for the lower bound mentioned. This is guaranteed by the inequalities

$$\sum_{t=1+m}^{u+m} s_{it}^- \leq \sum_{t=1}^{u} s_{it}^+ \quad \text{for all } i \text{ and } 1 \leq u \leq T-(m-1) \tag{6}$$

together with $s_{i2}^- = \ldots = s_{im}^- = 0 = s_{i,T-(m-2)}^+ = \ldots = s_{iT}^+$, where $s_{i1}^+ := x_{i1}$ and $s_{i,T+1}^- := x_{iT}$ for all i. To see why, fix a secondary resource type i. Let us number consecutively each setup (including the ones before the first period) and teardown (including the ones after the last period) throughout the planning horizon. Thus, the number associated with a setup made in period t will be strictly smaller than one made in period t' if $t < t'$ (the numbering within a period is immaterial); similarly for teardowns. Given a feasible solution of F2, instead of making the secondary resource assignments in Algorithm 1 arbitrarily, if one adopts the policy of performing the nth teardown (not on an arbitrary machine but) on the machine with the nth setup, then the best schedule with respect to minimum lot size is obtained. That is to say, for the given solution, this policy leads for i to a schedule with the largest possible minimum lot size – call it m_i'. We can express m_i' in terms of our setup and teardown variables: for $2 \leq k \leq T+1$, let

$$t_{ik} := \min\{l \geq 1 : s_{i2}^- + \ldots + s_{ik}^- \leq s_{i1}^+ + \ldots + s_{il}^+\}$$

and $m_{ik} := k - t_{ik}$. Then, for those k with $s_{ik}^- > 0$, the integer t_{ik} stands for the period containing the setup of the last resource of type i torn down at period k. Consequently, m_{ik} denotes how many periods ago this setup was performed. Hence

$$m_i' = \min\{m_{ik} : 2 \leq k \leq T+1, s_{ik}^- > 0\}.$$

Let $m' := \min_i\{m_i'\}$. Now we can state the minimum lot size restriction as a simple inequality: $m' \geq m$. Equivalently, $s_{ik}^- > 0$ implies $m_{ik} \geq m$ for all i and $2 \leq k \leq T+1$. Writing this out, we get (6) and the equalities thereafter. Algorithm 3 outlines the assignment procedure just described.

Algorithm 3 Disaggregation under minimum lot size.

1: assign the x_{i1} arbitrarily to machines
2: number for each i every setup $s_{i1}^+, \ldots, s_{iT}^+$ and teardown $s_{i2}^-, \ldots, s_{i,T+1}^-$ in order
3: **for** $t = 2$ **to** T **do**
4: **for all** i **do**
5: **for** $n = 1 + \sum_{t'=2}^{t-1} s_{it'}$ **to** $\sum_{t'=2}^{t} s_{it'}^-$ **do**
6: perform teardown n on the machine with setup n
7: assign, by respecting the minimum lot size constraint, $\sum_i s_{it}^+$-many secondary resources to those machines which have just been emptied

As is the case with the vertical constraints, F2 with (6) is NP-hard. Lemmas 5 and 6, hence Corollary 1, hold true. We were unable to find a counterexample that disproves Theorem 3 or 4. We adapt the heuristic in §3.4 to the problem with minimum lot size restriction as follows: after line 7 in Algorithm 2, we enlarge the remaining fractional demands so as to satisfy the minimum lot size constraint. The subsequent steps are applied as is. Complexity of the new algorithm is the same as the original heuristic's.

5. Computational study

We implemented the mixed-integer programs and the heuristic with C# programming language using CPLEX 12.7 as solver, on a PC with Intel(R) Core(TM)2 Quad CPU (2.40GHz) processor and 4 GB RAM, running a 64-bit Windows 7 operating system. We generated instances as follows: Given I, T, a lower bound l, an upper bound u, and a seed, the $\overline{x_i}$ are randomly generated as fractions such that $l \leq \overline{x_i} < u$ and $T\,\overline{x_i} \in \mathbb{Z}$. In other words, each $\overline{x_i}$ is of the form $a_i + \frac{b_i}{T}$, where a_i is a random integer between l and $u-1$ (inclusive), and b_i is a random integer between 0 and $T-1$. Then we generated M as a random integer between $\lceil \sum \overline{x_i} \rceil$ and $\sum \lceil \overline{x_i} \rceil$. Thus, M is greater than or equal to the smallest integer for which the problem has a feasible solution, and less than or equal to the smallest integer for which the objective value is zero (see Lemmas 2 and 5).

We shall take $l = 0$ and $T \geq 18$ throughout. Then the $\overline{x_i}$ are discrete uniform random variables assuming Tu-many values. We have $E(\overline{x_i}) = \frac{u-1}{2} + \frac{1}{2}\frac{T-1}{T} \approx \frac{u}{2}$ and $E(\lceil \overline{x_i} \rceil) = \frac{u+1}{2}$. Consequently, $E(\lceil \sum \overline{x_i} \rceil) \approx E(\sum \overline{x_i}) \approx \frac{Iu}{2}$ and $E(\sum \lceil \overline{x_i} \rceil) = \frac{I(u+1)}{2}$. Hence, expectation of M as a random variable is approximately $\frac{I(u+0.5)}{2}$. Let us designate this quantity by $\overline{M}$. Therefore, for fixed u, the average number $\overline{M}$ of machines in our instances is directly proportional to the number I of secondary resource types. We shall assume $u = 5$ so that $\overline{M} = 2.75I$.

By absolute optimality gap we mean the best (smallest) objective function value z_f minus the best (largest) lower bound z_l, and by relative optimality gap the ratio of this difference to z_f. Absolute MIP gap tolerance is a positive number `EpAGap` such that the solver terminates processing as soon as $z_f - z_l \leq$ `EpAGap`. Relative MIP gap tolerance `EpGap` is defined analogously. We know that z^* must be an even integer (Corollary 1), consequently `EpAGap` can be set to $2 - \varepsilon$. We take ε to be 10^{-6} (this is the default value for `EpAGap`). It is possible that z_f be sometimes odd in the course of solution process, so the optimal value in the output is to be understood as the largest even integer less than or equal to z_f. Making use of the evenness of z^* together with the inequality $z^* \leq 2I - 2$ (Lemma 7), one can also show that `EpGap` can be set to $1/I$; after setting `EpAGap` as $2 - \varepsilon$, however, this is redundant.

In our first experiment, we compared F1, F1 with symmetry-breaking constraints (3), and F2 on six sets of 25 instances each, where $I \in \{4, 8, 16\}$ and $T \in \{18, 36\}$. A time limit of 600 seconds is chosen. Statistics summarizing the experiment results are given in Table 2. The column Solved shows the number of instances solved to optimality, Gap the average percent (relative) optimality gap, and Time the average CPU time in seconds.

Table 2. Comparison of formulations F1, F1 with symmetry-breaking constraints (3), and F2.

		F1			F1 with symmetry-breaking constraints (3)			F2		
T	I	Solved	Gap (%)	Time (s)	Solved	Gap (%)	Time (s)	Solved	Gap (%)	Time (s)
18	4	25	0	2.5	25	0	2.1	25	0	0
	8	21	6.91	162	17	25.13	353.5	25	0	0
	16	9	57.09	416.7	6	76	468.1	25	0	9.1
36	4	25	0	17.2	25	0	16.9	25	0	0
	8	17	18.96	255.3	7	71.2	459.8	25	0	0.1
	16	4	80.02	525.5	0	100	600	25	0	20.1

Table 2 shows that F2 performs much better than F1. This can be explained by the fact that F1 has about M times as many variables and constraints as F2, possessing many alternative symmetric solutions. Inclusion of symmetry-breaking constraints (3) make the situation even worse. Symmetry-breaking constraints reduce the extent of feasible region that must be explored, but also make size of the linear programs to be solved at each node of the branch-and-bound tree larger, which could be the reason behind the disappointing solution performance of F1 with (3). In view of Table 2, it does not make sense to further consider F1 instead of F2 unless this is dictated by a change of underlying assumptions. Therefore we focus on F2. Although CPU time is virtually zero for small instances, this situation is bound to change for large ones as the problem is shown to be

NP-hard (Theorem 1).

In our second experiment, we investigated the effect of setting the absolute MIP gap tolerance `EpAGap` to $2-\varepsilon$ in F2. For this purpose, solution performances of F2 (with `EpAGap` assuming its default value of 10^{-6}) and F2 with `EpAGap` $= 2-\varepsilon$ are compared in six sets of 25 instances each, where $I \in \{100, 200, 400\}$ and $T \in \{18, 36\}$. See Table 3 for the statistics thus obtained, which shows that this parameter's redefinition leads to a slight improvement in solution performance. As the optimal objective z^* is known to be an even integer for the model extensions as well, we shall take absolute tolerance as $2-\varepsilon$ for all the experiments from this point on.

Table 3. Effect of changing the absolute MIP gap tolerance.

		F2			F2 with `EpAGap` $= 2-\varepsilon$		
T	I	Solved	Gap (%)	Time (s)	Solved	Gap (%)	Time (s)
	100	25	0	1.7	25	0	1.6
18	200	24	0.06	32.4	24	0.06	32.9
	400	23	0.13	72.6	23	0.13	72
	100	22	0.35	79.9	24	0.2	51.4
36	200	23	4.04	85.3	23	4.04	74
	400	25	0	99.3	25	0	76.3

Solution performance of our heuristic algorithm is given in Table 4. Gap here is defined like relative optimality gap above, with z_f replaced by the objective value z_h associated with the heuristic solution. We use as lower bound the number z_l provided for F2 by the solver in 600 seconds. The column Solved shows the number of instances for which the heuristic solution is optimal (i.e., $z_h - z_l < 2$).

Table 4. Performance of the heuristic algorithm.

	Small instances				Large instances			
T	I	Solved	Gap (%)	Time (s)	I	Solved	Gap (%)	Time (s)
	4	21	16	0	100	7	10.2	0
18	8	24	4	0	200	10	8.36	0.3
	16	25	0	0	400	4	5.26	1.9
	4	22	12	0	100	8	4.35	0.1
36	8	23	1.8	0	200	11	5.02	0.4
	16	22	1.31	0	400	6	5.81	2.2

According to Table 4, the heuristic algorithm has found an optimal solution in more than 30% of the large instances in less than one second on average. The results are much better for the small instances. Hence, it is reasonable to feed the solver with an initial heuristic solution. We performed tests to see the effect of this on solution performance. We also ran the same experiments by incorporating the cut (4) into the formulation. The results are tabulated in Table 5.

Table 5. Effect of providing an initial heuristic solution and adding the cut (4).

		F2 with heuristic			F2 with (4)			F2 with heuristic and (4)		
T	I	Solved	Gap (%)	Time (s)	Solved	Gap (%)	Time (s)	Solved	Gap (%)	Time (s)
	100	25	0	1.5	25	0	3.9	25	0	3.9
18	200	24	0.06	31.6	24	0.06	33.6	24	0.06	33.5
	400	23	0.12	81.3	23	0.08	71.7	23	0.08	75.4
	100	23	0.19	62.2	24	0.12	34.4	24	0.08	32.6
36	200	23	0.37	72	23	0.13	78	23	0.15	73.3
	400	25	0	73.9	25	0	90.1	25	0	76.6

When we compare the results in Table 5 with those for F2 with `EpAGap` $= 2-\varepsilon$ in Table 3, we see that the initial heuristic solution leads to a significant improvement in gap for the parameter set $(I, T) = (200, 36)$. However, for $(I, T) = (100, 36)$, the number of instances solved to optimality is one less. The results for F2 with (4) are even better except for the relatively larger solution time for $(I, T) = (400, 36)$. The most promising implementation is F2 with heuristic and constraints (4). This combination yields generally the best statistics.

Next, we provide computational results for two realizations of the model extensions discussed in §4. First, we implemented F2 with vertical constraints (5), limiting by $\overline{s}$ the total number of setups and teardowns in between every two consecutive periods. We took $\overline{s} = 2$, and carried out tests for the same six sets of instances used in Table 2. Second, we ran F2 with minimum lot size constraints (6), setting a lower bound m on the number of periods a secondary resource should remain installed in a machine. We took $m = 3$. Statistics are summarised in Table 6. In comparison to Table 2, we see how the additional constraints (5) and (6) worsen solution performance, especially when I is large relative to T.

Table 6. Performance of the mixed-integer programs for the extended problems.

		F2 with (5) where $\overline{s} = 2$			F2 with (6) where $m = 3$		
T	I	Solved	Gap (%)	Time (s)	Solved	Gap (%)	Time (s)
	4	25	0	0	25	0	0
18	8	25	0	0.3	25	0	0.3
	16	24	0.96	26.8	24	0.95	26.5
	4	25	0	0.1	25	0	0.1
36	8	25	0	0.3	25	0	0.5
	16	22	3.05	134.5	21	3.77	158.7

Finally, in Table 7 we give statistics that show the performance of the heuristics described in §4.1 and §4.2 for the extended problems. Here the column Feasible shows the number of instances for which the algorithms were able to find a feasible solution. Note that most of the time our heuristics could come up with an optimal schedule. The average gaps are not very large as well, although they were considerably smaller for the original problem as shown by Table 4.

Table 7. Performance of the heuristics for the extended problems.

		Vertical constraints ($\overline{s} = 2$)				Minimum lot size constraints ($m = 3$)			
T	I	Feasible	Solved	Gap (%)	Time (s)	Feasible	Solved	Gap (%)	Time (s)
	4	25	21	16	0	24	20	16.67	0
18	8	25	20	11.33	0	24	19	11.46	0
	16	21	14	9.63	0	24	16	8.44	0
	4	25	22	12	0	25	22	12	0
36	8	24	21	6.23	0	25	22	5.28	0
	16	20	8	17.91	0	24	6	21.94	0

6. Conclusion and further research

We studied in this paper a parallel machine multi-item lot-sizing and scheduling problem with a secondary resource, in which demands are given for the entire planning horizon rather than for every single period. We proved that the problem is NP-hard, and presented two equivalent formulations as mixed-integer linear programs. The second formulation, using aggregate integer variables instead of individual binary variables for each machine, turned out to be more efficient. We showed some properties the optimal objective value z^* must satisfy, proposed optimality conditions, and gave a heuristic algorithm. In particular, z^* must be even so that the absolute MIP gap tolerance can be set to $2 - \varepsilon$. This redefinition slightly improves the solution performance of the aggregate formulation, which is also true of feeding the solver with an initial heuristic solution in general. Incorporation of the cut (4) leads to better results, making up the best implementation together with the heuristic.

We discussed two possible extensions to the problem, namely vertical and minimum lot size constraints. We showed how these can be formulated, and mentioned some properties of the new problems arising thereby. Furthermore, we suggested two modifications of the heuristic. Computational tests indicate that it is significantly more time-consuming to solve these extended problems.

We conjecture for the original problem that there exist optimal schedules satisfying the following

two sets of inequalities:

$$\begin{aligned} &\lfloor \overline{x_i} \rfloor \leq x_{it} \leq \lceil \overline{x_i} \rceil && \text{for all } i,t; \\ &\sum_t s_{it}^+ \leq 1, \quad \sum_t s_{it}^- \leq 1 && \text{for all } i. \end{aligned}$$

It can be interesting for further research to provide a proof or disproof of this claim, and investigate other practical situations such as some machines being temporarily unavailable within the planning horizon.

References

Buschkühl, L., F. Sahling, S. Helber, and H. Tempelmeier. 2010. "Dynamic capacitated lot-sizing problems: a classification and review of solution approaches." OR Spectrum 32 (2): 231–261.

Copil, K., M. Wörbelauer, H. Meyr, and H. Tempelmeier. 2017. "Simultaneous lotsizing and scheduling problems: a classification and review of models." OR Spectrum 39 (1): 1–64.

Drexl, A., and A. Kimms. 1997. "Lot sizing and scheduling — Survey and extensions." European Journal of Operational Research 99 (2): 221–235.

Eppen, G. D., and R. K. Martin. 1987. "Solving multi-item capacitated lot-sizing problems using variable redefinition." Operations Research 35 (6): 832–848.

Fiorotto, D. J., and S. A. de Araujo. 2014. "Reformulation and a Lagrangian heuristic for lot sizing problem on parallel machines." Annals of Operations Research 217 (1): 213–231.

Gicquel, C., M. Minoux, and Y. Dallery. 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, C., L. A. Wolsey, and M. Minoux. 2012. "On discrete lot-sizing and scheduling on identical parallel machines." Optimization Letters 6 (3): 545–557.

Jans, R. 2009. "Solving lot-sizing problems on parallel identical machines using symmetry-breaking constraints." INFORMS Journal on Computing 21 (1): 123–136.

Jans, R., and Z. Degraeve. 2004. "An industrial extension of the discrete lot-sizing and scheduling problem." IIE Transactions 36 (1): 47–58.

Jans, R., and Z. Degraeve. 2008. "Modeling industrial lot sizing problems: a review." International Journal of Production Research 46 (6): 1619–1643.

Jordan, C. 1996. Batching and Scheduling. Springer.

Kaczmarczyk, W. 2011. "Proportional lot-sizing and scheduling problem with identical parallel machines." International Journal of Production Research 49 (9): 2605–2623.

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

Kimms, A. 1997. Multi-Level Lot Sizing and Scheduling. Physica-Verlag.

Lasdon, L. S., and R. C. Terjung. 1971. "An Efficient Algorithm for Multi-Item Scheduling." Operations Research 19 (4): 946–969.

Pochet, Y., and L. A. Wolsey. 2006. Production Planning by Mixed Integer Programming. Springer.

Quadt, D. 2004. Lot-Sizing and Scheduling for Flexible Flow Lines. Springer.

Quadt, D., and H. Kuhn. 2007. "A taxonomy of flexible flow line scheduling procedures." European Journal of Operational Research 178 (3): 686–698.

Quadt, D., and H. Kuhn. 2009. "Capacitated Lot-Sizing and Scheduling with Parallel Machines, Back-Orders, and Setup Carry-Over." Naval Research Logistics 56 (4): 366–384.

Seeanner, F. 2013. Multi-Stage Simultaneous Lot-Sizing and Scheduling. Springer Gabler.

Sherali, H. D., and J. C. Smith. 2001. "Improving Discrete Model Representations via Symmetry Considerations." Management Science 47 (10): 1396–1407.

Suerie, C. 2005. Time Continuity in Discrete Time Models. Springer.

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

Vanderbeck, F., and L. A. Wolsey. 1992. "Valid Inequalities for the Lasdon-Terjung Production Model." The Journal of the Operational Research Society 43 (5): 435–441.

Zhu, X., and W. E. Wilhelm. 2006. "Scheduling and lot sizing with sequence-dependent setup: A literature review." IIE Transactions 38 (11): 987–1007.