MIP approaches for a lot sizing and scheduling problem on multiple production lines with scarce resources, temporary workstations, and perishable products

Willy A. O. Soler[a], Maristela O. Santos[b] and Kerem Akartunalı[c]

[a]Universidade Federal de Mato Grosso do Sul, Campo Grande, Brazil; [b]Universidade de São Paulo, São Carlos, Brazil; [c]University of Strathclyde, Glasgow, UK.

ARTICLE HISTORY
Compiled July 1, 2019

ABSTRACT
This paper addresses a lot sizing and scheduling problem inspired from a real-world production environment apparent in food industry. Due to the scarcity of resources, only a subset of production lines can operate simultaneously, and those lines need to be assembled in each production period. In addition, the products are perishable, and there are often significant sequence-dependent setup times and costs. We first propose a standard mixed integer programming model for the problem, and then a reformulation of the standard model in order to allow us to define a branching rule to accelerate the performance of the branch-and-bound algorithm. We also propose an efficient relax-and-fix procedure that can provide high-quality feasible solutions and competitive dual bounds for the problem. Computational experiments indicate that our approaches provide superior results when benchmarked with a commercial solver and an established relax-and-fix heuristic from the literature.

KEYWORDS
Lot sizing and scheduling problem; Production; Scarce resources; Branch and Bound; Heuristics.

1. Introduction

The lot sizing and scheduling problem (LSP) aims to simultaneously determine the production quantities and the sequence of production while minimizing costs. The LSP has received great attention from researchers because of its importance in the global economy and also due to the challenges to solve it in several practical situations (Almada-Lobo, Clark, Guimarães, Figueira, and Amorim (2015); Glock, Grosse, and Ries (2014)). Several mixed integer programming approaches have been proposed to deal with LSP, as detailed in the recent review of Copil, Wörbelauer, Meyr, and Tempelmeier (2017).

In this paper, we address an LSP originated from a Brazilian food company that produces meat products. In this production setting, various production lines share scarce resources (machines and workers) that need to be designated to the production lines that will be open at the beginning of the each period. Moreover, the assembly of the production lines may require the creation of temporary workstations and changes

Emails: willy.oliveira@ufms.br; mari@icmc.usp.br; kerem.akartunali@strath.ac.uk (Corresponding Author)

in the plant layout, and hence it is desirable for production lines to be assembled at the start of each production period (typically a day) and remain open until the end of the period. Since each product can be produced only by a specific production line, the choice of which production lines to assemble in any given period affects the set of products produced therein. Moreover, for each production line, changeovers between items cause sequence-dependent setup costs and times, e.g., due to different cleaning procedures required for the transition from product A to B vs. from product B to A. All products are perishable and hence require monitored temperature storage, which impacts the inventory holding costs significantly. The typical customers are supermarkets and restaurants, where delays in meeting customer demand are likely to result in loss of customers, and thus backlogging is only allowed with high penalties.

To sum up, we have three sets of decisions to make in each period of the planning horizon: i) which production lines to assemble; ii) the sequence of production of the lots in each line; and iii) the size of the production lots on each line. The aim of the problem is to minimize the sum of costs associated with production (namely, inventory holding, backlogging and setup) and costs associated with the assembly of the production lines, while respecting constraints with regards to the fulfillment of customer demands, the capacities of resources, the capacities of production lines, and the shelf life of products.

We remark that the main difference between the traditional LSP on parallel machines (considered, for example, in Almeder and Almada-Lobo (2011); Meyr and Mann (2013); Xiao, Yang, Zhang, Zheng, and Gupta (2015)) and the problem addressed in this paper is the need to decide in each period which production lines to assemble, considering the limited available capacity of the production resources required to perform these assembly operations. It is then natural to ask "why not acquire the necessary quantities of resources in order to keep all production lines active during all production periods?" There are two key answers to this question:

i) acquiring new resources often require an infeasible investment, because the plant needs to be enlarged, various machines need to be acquired and workers need to be hired; and
ii) if all production lines remain in production all the time, then there will be idle machines and idle workers, because demands for individual products vary significantly from one period to another, and the production of large amounts of perishable products may significantly increase the inventory holding costs and the number of products discarded due to limited shelf life.

Therefore, in this paper we propose specific models and solution approaches to deal with this problem.

A pioneer MIP model was proposed by Smith-Daniels and Ritzman (1988) to deal with the simultaneous lot sizing and scheduling problem. The model was motivated by observations made in process industries, and the aim is to minimize the sum of the inventory holding and backlogging costs weighting the completion times of the products. In the model, the products are grouped into families, and sequence dependent setup times are considered between products of different families. Specific characteristics, such as secondary production resources and perishable products are not considered. Due to the limited computational resources of the time, the model was tested only in small instances.

Other related research was carried out by Özdamar and Birbil (1998), who studied a capacitated lot sizing and loading problem (CLSLP) in industries with multiple facilities. Each facility can be active or not in each period of a finite planning horizon. Due to technological limitations, for each item, just a subset of existing facilities can

produce it. The CLSLP differs from the problem described in this paper, because CLSLP does not consider: i) sequence dependent setups (hence no scheduling); ii) the perishibility of the items; and iii) the capacity of secondary resources to assemble the facilities. A hybrid heuristic approach combining simulated annealing, tabu search and genetic algorithm was proposed and computational tests were performed considering (relatively small) instances (up to 20 items, 5 facilities and 6 periods) and indicated that the CLSLP is a very challenging problem from a computational perspective.

In another study, Almeder and Almada-Lobo (2011) investigated a lot sizing and scheduling problem on multiple machines, where each machine needs to be equipped with secondary resources (tools) to produce the items. These secondary resources can be used on every machine and due to the scarcity of these resources, their utilization needs to be synchronized. Therefore, variables and constraints were incorporated in two classical lot sizing and scheduling models to capture the start time and the end time of utilization of each tool in each machine. A specific solution approach was not proposed, and the analysis of the computational performance of the models showed that standard branch-and-cut algorithms can not provide good solutions in acceptable computational times.

Afzalirad and Rezaeian (2016) studied a pure scheduling problem on unrelated parallel machines with sequence dependent setup times, and precedence and resource constraints. As in Almeder and Almada-Lobo (2011), all machines are available anytime, but to process each job, some secondary resources are required. An integer programming model was proposed to minimize the completion time of the jobs, and the resource utilization was managed using knapsack constraints. A genetic algorithm and an artificial immune system (AIS) were proposed to find solutions for the problem.

The unrelated parallel machine scheduling problem was also studied by Villa, Vallada, and Fanjul-Peyro (2018) considering additional scarce resource. The addressed problem is similar to the problem considered in Afzalirad and Rezaeian (2016) in the sense that processing each job requires an amount of additional resources (that may be a tool or a human resource). Several heuristic approaches were introduced including construction and improvement procedures, where construction procedures are adaptations of heuristics proposed in the literature for other problems, while improvement procedures are based on local search.

Finally, Güngör, Ünal, and Taşkın (2017) addressed a lot sizing and scheduling problem on parallel machines motivated by foundry planning in aluminium alloy wheel production. In this problem, the machines (production lines) are identical and in order to produce an item, a specific secondary resource (mold) needs to be installed in the machine. Besides that, the demands are given for the entire planning horizon rather than for every single period (cumulative demand) and the all-or-nothing assumption is valid for all production periods. Güngör et al. (2017) showed that the problem is $\mathcal{NP}$-hard, proposed valid inequalities and a polynomial-time heuristic algorithm for the problem.

In addition to the fact that the LSP is $\mathcal{NP}$-hard (James & Almada-Lobo, 2011), the real-world instances of the problem are very challenging from a computational perspective. Moreover, even the pure lot sizing problem with multiple items sharing resources is often computationally challenging (Akartunalı, Fragkos, Miller, & Wu, 2016; Doostmohammadi & Akartunalı, 2018), and sequencing lot sizes necessitates novel approaches (Suerie & Stadtler, 2003) due to further complications, as also further discussed in the recent lot sizing review of Brahimi, Absi, Dauzère-Pérès, and Nordli (2017). Therefore, many researchers proposed heuristic approaches to solve the LSP, and MIP heuristics in particular achieved promising results for problems arising in

various practical applications (Sel & Bilgen, 2014; Toledo, da Silva Arantes, Hossomi, França, & Akartunalı, 2015). Starting with the MIP formulation of the problem on hand, these heuristics decompose the problem into several small sub-problems, where only a subset of the complicating binary/integer variables are determined. In this paper, we propose an MIP heuristic to tackle the specific LSP in practice, in addition to developing and evaluating branching rules and reformulations of the problem.

The paper is organized as follows: In Section 2, we detail the problem characteristics. In Section 3, we present a standard MIP formulation of the specific problem on hand. In Section 4, we discuss a reformulation and a branching rule for use in standard branch-and-bound, and then present in detail the proposed heuristic employing a relax-and-fix approach. In Section 5, we present and discuss computational results in order to evaluate: (i) the efficiency of the proposed approaches, (ii) the impact of considering the perishability of the products and scarce resources, and (iii) the impact of changing the value of some input parameters on the solution structure of the problem and on the computational performance of the proposed approaches. Finally, Section 6 concludes the paper with future research.

2. Problem description

This paper deals with an LSP problem inspired by the Brazilian food industry that produces packaged meat for retail and exportation. In this type of industry, the product catalogue is usually broad, including meat from different animals that can be seasoned, frozen or *in natura*. The plant operates with various production lines that share the same scarce production resources, such as machines, tools, and workers. For example, a particular cutting machine is used to process beef as well as pork, and these two types of meat are often produced in different production lines. On the other hand, the workers can operate all of the production lines, but often number of workers is not sufficient to operate all production lines simultaneously. The production lines cannot remain working all the time, because the necessary resources to keep it working are shared between production lines, and therefore the lines are assembled in each production period respecting the amount of available resources. As for each item, there is only one line capable to produce it, hence the choice of which production lines to assemble impacts which items to produce. The demand is dynamic and known as a result of advanced customer orders, and as mentioned before, backlogging is allowed but highly penalized.

The produced items are perishable with different shelf life. For example, *in natura* meat can remain in stock only few days, while frozen meat can remain in stock for up to two years. As the items require monitored temperature storage, the inventory holding costs are significant and as only a subset of items can be produced in each period and as the items are perishable, the inventory management is a challenging activity for the production managers.

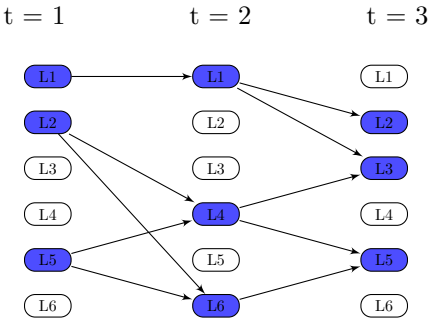
This production environment is also characterized by the existence of significant sequence dependent setup times and costs when there are changes between items in the same production line. Consider, for example, a case with only one production line and three items: fine seasoned steak (p_1), thick steak for barbecue (p_2) and fine unseasoned steak (p_3). The necessary setup procedures to change from (p_1) to (p_2) are cleaning and cutting machine configuration, while from (p_1) to (p_3) only cleaning is needed, and from (p_3) to (p_1) cleaning and seasoning preparation are needed. Considering this, changes involving different items imply different setup procedures, which are converted

into different setup times and costs. Moreover, a general cleaning is carried out at the end of each production day in order to avoid contamination. This cleaning involves the disassembly of the production lines and the temporary workstations, hence not allowing a setup carryover from a day to the next, and the first setup of each day is performed along the assembly procedures. Therefore, there is no setup time/cost for the first produced item in each line and period.

In addition, we need to decide which production lines to assemble in each production period, not violating the available capacity of each resource required to assemble the lines. Figure 1 presents a framework of the studied industrial environment. We consider a planning horizon with three periods ($t = 1$, $t = 2$ and $t = 3$) and consider six production lines (L1,. . .,L6) sharing three scarce production resources (R1, R2 and R3). The assembled lines are represented by blue rectangles, while the not assembled lines are represented by white rectangles. For example, in period $t = 1$ the lines L1, L2 and L5 were assembled, while the lines L3, L4 and L6 are not producing. The arrows indicate the movement of resources between different periods.

Figure 1 also presents a table with the amount of each resource required to assemble each production line and the amounts of available (A), consumed (C) and idle (I) resources in each period. The amount of consumed resource is given by the sum of resources used in each assembled production line, while the amount of idle resource is the difference between the amount of available resource and the amount of consumed resource. We can see that in each period, we can not assemble more production lines because some resources are not available.

Figure 1. Framework of the considered production environment



	Required			t = 1			t = 2			t = 3		
	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3
L1	2	3	1	assembled			assembled			not assembled		
L2	1	1	4	assembled			not assembled			assembled		
L3	1	2	2	not assembled			not assembled			assembled		
L4	1	2	1	not assembled			assembled			not assembled		
L5	2	1	3	assembled			not assembled			assembled		
L6	1	1	3	not assembled			assembled			not assembled		
A	5	6	9	5	6	9	5	6	9	5	6	9
C	-	-	-	5	5	8	4	6	5	4	4	9
I	-	-	-	0	1	1	1	0	4	1	2	0

3. Standard formulation

In order to model the problem, we first propose the CLSDPRL (Capacitated Lot-sizing with Sequence Dependent setup costs, Perishable products, scarce Resources

and multiple production Lines) formulation, an extension of the CLSD (Capacitated lot-sizing with sequence dependent setup costs) formulation proposed by Haase (1996). The parameters and variables, as well as domains of variables, are presented in Table 1. We note that for the set of resources K, we use the index $k = 0$ to indicate "workers" (which is often a single type of resource, as observed in our visits to factories) and any index $k \geq 1$ to indicate the different types of machines or tools.

Table 1. Parameters and variables for the CLSDPRL formulation

Parameters	
T, L	Set of periods (indexed by t), and of production lines (indexed by l)
J, K	Set of items (indexed by i, j), and of resources (indexed by k)
d_{jt}, C_{lt}	Demand of item j, and production capacity of line l, in period t
a_{lj}, m_{lj}	Per unit production time, and the minimum lotsize of item j on line l
h_j, b_j	Inventory and backlogging costs per unit of item j
sc_{lij}, st_{lij}	Setup cost and time for changeover from item i to j on line l
r_{kl}	Amount of resource k necessary to assemble the production line l
R_{kt}	Capacity of resource k in period t
sl_j, ac_l	Shelf life of item j, and cost to assemble the line l
γ_{lt}	Maximum number of items that can be produced on line l in period t
P_l	Set of items that can be produced on line l
Variables	
$ve_{jt}^{t'} \geq 0$	Stock of item j of age t' available in period t, where $t' \leq \min\{t, sl_j\}$
$q_{jt}^{t'} \geq 0$	Quantity of item j used to meet the demand of period t using $ve_{jt}^{t'}$
$I_{jt}, B_{jt} \geq 0$	Inventory and backlogging of item j at the end of period t
$x_{ljt} \geq 0$	Amount of item j produced on line l and period t
$V_{ljt} \geq 0$	Auxiliary variables to represent the order of production of item j on line l and period t
$y_{ljt} \in \{0,1\}$	1 if item j is the first item produced on line l and period t, and 0 otherwise
$z_{lijt} \in \{0,1\}$	1 if there is change of production from item i to j on line l and period t, and 0 otherwise
$\delta_{lt} \in \{0,1\}$	1 if the line l is assembled in period t, and 0 otherwise

In our model, we consider the following characteristics: i) dynamic and deterministic demand that must be fully met until the end of the planning horizon; ii) no setup carryover or crossover between adjacent periods; iii) for each product, only one production line capable of producing it; and iv) perishable items. We note that setup carryover and crossover are not considered, and hence no setup cycles exist. The perishability aspect is modelled using an adaptation of the approach proposed by Costa, dos Santos, Alem, and Santos (2014), which consists of controlling the age of each product in stock. Then, CLSDPRL follows:

$$\text{Min} \sum_{t,j}(h_j I_{jt} + b_j B_{jt}) + \sum_{l,t,i,j} sc_{lij} z_{lijt} + \sum_{l,t} ac_l \delta_{lt} \tag{1}$$

$$\text{s.t.} \sum_{t'=0}^{sl_j} q_{jt}^{t'} = d_{jt} + B_{j,t-1} - B_{jt}, \forall j, t \tag{2}$$

$$\sum_{j \in P_l} a_{lj} x_{ljt} + \sum_{i,j} st_{lij} z_{lijt} \leq C_{lt}, \forall l, t \tag{3}$$

$$m_{lj}\left(y_{ljt} + \sum_{i \in P_l} z_{lijt}\right) \leq x_{ljt} \leq \frac{C_{lt}}{a_{lj}}\left(y_{ljt} + \sum_{i \in P_l} z_{lijt}\right), \forall l, j \in P_l, t \tag{4}$$

$$\sum_{j \in P_l} y_{ljt} \leq \delta_{lt}, \forall l, t \tag{5}$$

$$\sum_{i,j \in P_l} z_{lijt} \leq (\gamma_{lt} - 1)\delta_{lt}, \forall l, t \tag{6}$$

$$y_{ljt} + \sum_{i \in P_l} z_{lijt} \leq \delta_{lt}, \forall l, j \in P_l, t \tag{7}$$

$$y_{ljt} + \sum_{i \in P_l} z_{lijt} \geq \sum_{i \in P_l} z_{ljit}, \forall l, j \in P_l, t \tag{8}$$

$$V_{ljt} \geq V_{lit} + 1 - \gamma_{lt}(1 - z_{lijt}), \forall l, t, i, j \in P_l \tag{9}$$

$$\sum_{l} r_{kl}\delta_{lt} \leq R_k, \forall k, t \tag{10}$$

$$ve_{j,t+1}^{t^{'}+1} = ve_{jt}^{t^{'}} - q_{jt}^{t^{'}}, \forall j, t, t^{'} \leq sl_j - 1 \tag{11}$$

$$ve_{jt}^{0} = \sum_{l} x_{ljt}, \forall j, t \tag{12}$$

$$q_{jt}^{t^{'}} \leq ve_{jt}^{t^{'}}, \forall j, t, t^{'} \leq sl_j \tag{13}$$

$$I_{jt} = \sum_{t^{'}=0}^{sl_j} ve_{jt}^{t^{'}} - \sum_{t^{'}=0}^{sl_j} q_{jt}^{t^{'}}, \forall j, t \tag{14}$$

The objective function (1) represents the sum of the inventory holding, backlogging, and setup costs, and costs to assemble the production lines. Constraints (2) are the inventory balance constraints, and constraints (3) represent the big-bucket capacity constraints for production lines. Constraints (4) ensure minimum lot sizes are respected and an item can only be produced if the respective production is setup, while constraints (5) dictate that if a production line is assembled, only one item can be the first item produced on that line. Constraints (6) introduce maximum number of items that can be produced on each production line and period, and constraints (7) ensure that the production can only occur on an assembled production line. Constraints (8) ensure the flow balance for sequencing of lots, and (9) are the subtour elimination constraints. Constraints (10) are the capacity constraints for resources. Constraints (11)-(13) model the perishability aspect. More specifically, constraints (11) iteratively update the age of the products in stock, while constraints (12) compute the amount of freshly produced stock and constraints (13) ensure that amount used of a particular age does not exceed the available stock of that age. Finally, constraints (14) compute the inventory of each item for each period. We also note that we set $B_{jT} = 0$, $\forall j$ to ensure all demands are satisfied until the end of the planning horizon.

4. Branching rule and relax-and-fix heuristic

4.1. Branching rule

We note that due to constraints (7), when we have $\delta_{lt} = 0$, then there can be no production on line l in period t, i.e., $y_{ljt} = 0, \forall j$ and $z_{lijt} = 0, \forall i, j$. Therefore, if in a given node of the branch-and-bound tree the value of δ_{lt} is fixed to zero, then we can

also fix directly the value of $J + J^2$ binary variables to zero.

In addition, we can introduce a new binary variables w_{ljt}, where $w_{ljt} = 1$ if item j is produced on line l in period t, and $w_{ljt} = 0$ otherwise. Clearly, we have that

$$w_{ljt} = y_{ljt} + \sum_i z_{lijt}, \quad \forall l, j, t, \tag{15}$$

We can also replace constraints (4) by (16), and constraints (6) and (7) by (17).

$$m_{lj} w_{ljt} \leq x_{ljt} \leq \frac{C_{lt}}{a_{lj}} w_{ljt}, \forall l, j \in P_l, t; \tag{16}$$

$$\sum_{j \in P_l} w_{ljt} \leq \gamma_{lt} \delta_{lt}, \forall l, t. \tag{17}$$

We define then the CLSDPRL^w formulation obtained by adding constraints (15) and the variables $w_{ljt} \in \{0, 1\}, \forall l, j, t$ into CLSDPRL and replacing (4) by (16), and (6) and (7) by (17). We note that since (15) holds, if $w_{ljt} = 0$, then $y_{ljt} = 0$ and $z_{lijt} = 0, \forall i$, and also $z_{ljit} = 0, \forall i$ (by (8)). Therefore, we can exploit the fact that in a given branch-and-bound tree node, when w_{ljt} is fixed to zero, then also the value of $1 + 2J$ binary variables can be fixed to zero.

These observations motivated us to use CLSDPRL^w and perform branch-and-bound with a priority on branching of variables $\delta_{lt}, \forall l, t$ and $w_{ljt}, \forall l, j, t$. In each search tree node, given an optimal solution of the linear relaxation, we set our branching rule as follows: if there is any variable δ_{lt} with a fractional value, we firstly perform the branching in these variables before any other binary variables; and, if all variables δ_{lt} assume binary values, but there is any variables w_{ljt} with a fractional value, we perform the branching in this variables before the remaining binary variables. We note that our branching rule simply sets a precedence relation between the variables to be branched and is used in conjunction with other existing rules of Cplex solver. As any commercial solver rather operates in a "black box" fashion with its own existing rules, our branching rule simply encourages the solver for our rule when selecting the variable to branch on.

Finally, we note that in Soler and Santos (2017), the binary production variables w_{ljt} were employed in the traditional CLSD model on a single machine, and a priority rule branching these variables first was evaluated. Computational results indicated that this approach can significantly improve the computational performance of the standard branch-and-bound.

4.2. A relax-and-fix procedure

The relax-and-fix is a constructive procedure proposed for production planning problems (Dillenberger, Escudero, Wollensak, & Zhang, 1994). The procedure operates by decomposing the set of the complicating binary/integer variables into subsets, which are preferably ordered with decreasing significance. In each iteration, the integrality requirements are only imposed on the binary/integer variables of the current subset, while the binary/integer variables are either linearly relaxed or have their value fixed in a previous iteration.

We propose a relax-and-fix procedure (RFH) that consists of two major iterations.

- *First iteration*: Solve a sub-problem of CLSDPRL^w enforcing the integrality

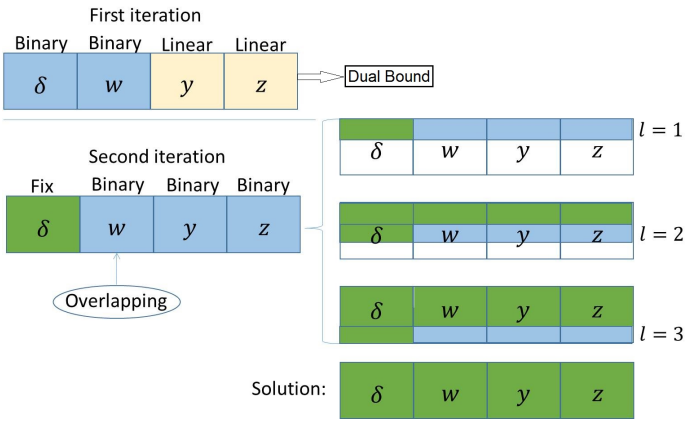
requirements of $\delta_{lt}, \forall l, t$ and $w_{ljt}, \forall l, j, t$ (while $y_{ljt}, \forall l, j, t$ and $z_{lijt}, \forall l, i, j, t$ are linearly relaxed);

- *Second iteration*: Solve a sub-problem of CLSDPRL^w with the values of $\delta_{lt}, \forall l, t$ fixed from the first iteration and enforcing the integrality requirements of all remaining binary variables.

The decisions about which production lines to assemble are made in the first iteration considering the lot sizing component of the problem as well as the capacity constraints of resources required, and the second iteration evaluates the lot sizing component of the problem together with the sequencing decisions. This approach empirically avoids myopic decisions, as the variables $w_{ljt}, \forall l, j, t$ are optimized in both iterations (overlapping variables).Even though our framework cannot theoretically guarantee feasible solutions (unless backlogging is allowed to the last period by removing $B_{jT} = 0$), its strengths such as limited fixing after first iteration and use of an effective formulation in the first iteration enable it to work very effectively in practice, as we will further discuss in Section 5. We also note that this procedure cannot be applied using the standard formulation CLSDPRL.

An advantage of this framework is that the subproblem referring to the second iteration can be further decomposed into L subproblems ($SP^l, l = 1, \ldots, L$), each with a single production line and only the periods where $\delta_{lt} = 0$ does not hold. This is based on the observation that the constraints (10) are already satisfied by the solution obtained in the first iteration. Each subproblem SP^l can be effectively solved using a branch-and-cut algorithm with the branching rule proposed in Soler and Santos (2017). Figure 2 presents an overview of our RFH heuristic.

Figure 2. A framework for RFH procedure



It is also important to note that, if necessary, the second iteration of the RFH heuristic can be modelled as a parallel algorithm since there is no data dependence among the subproblems $SP^l, l = 1, \ldots, L$. Each subproblem can be solved by a branch-and-bound algorithm running on independent cores of a computer with at least L cores. Alternative parallel schemes can be developed if the number of cores is less or (much) more than L: In the former, more than one subproblem share the same core and hence these subproblems are sequentially solved. In the latter, multiple cores can be allocated for each subproblem and hence branch-and-bound algorithm can run in parallel (note solvers like Cplex run branch-and-bound in parallel by default). Our proposition to parallelize the second iteration can adopt tools such as MPI (OpenMPI), OpenMP, and pthreads library depending on hardware and software infrastructures.

In the literature, relax-and-fix has been primarily employed as an heuristic procedure. However, we remark that the dual bound obtained solving the subproblem of the first iteration of a relax-and-fix procedure is always a dual bound of the original problem, which is obviously at least as strong as the bound obtained by the pure linear relaxation (and potentially much stronger). To the best of our knowledge, this property (formally stated in Proposition 4.1) is only noted in Akartunalı and Miller (2009), albeit with limited empirical results and a lack of further discussion. Therefore, we aim to elaborate further on this significant property with thorough computational experiments in the following section.

Proposition 4.1. *Let z_D be the dual bound obtained by solving the subproblem referring to the first iteration of a relax-and-fix procedure applied to solve a problem P. Then, z_D is a dual bound for the original problem P.*

Proof. W.l.o.g., let P be a minimization problem. First, note that in the first iteration of relax-and-fix, a subproblem P_1 is solved where some (but not all) of the binary variables of P are linearly relaxed, along with all constraints of P. Therefore, P_1 is a valid relaxation of P, and $\overline{z} \leq z^*$ holds, where $\overline{z}$ and z^* are the optimal objective function values for P_1 and P, respectively. Since z_D is a dual bound for P_1, we have $z_D \leq \overline{z}$. Hence, $z_D \leq z^*$. □

Note that although the optimal solution $\overline{z}$ of P_1 is naturally a dual bound for the original problem P, it is not always possible to optimally solve P_1 in limited computational times, and hence z_D is used in such cases.

5. Computational results

5.1. Features of the test instances

In order to empirically evaluate the effectiveness of the proposed approaches, we propose a set of 100 test instances with a broad range of scenarios. As the real-world data cannot be disclosed due to confidentiality reasons, we randomly generated test instances following the parameter settings we observed in a number of visits to factories of varying sizes and discussing with key stakeholders, in order to be in line with the practice. We grouped the instances into five classes (to represent industries from small- to large-size), each with 20 test instances and its own parameter settings, as presented in Table 2. Here, the first four columns indicate the fundamental characteristics of the problem such as number of time periods, and the remaining parameters φ^d, φ^r_k, φ^r_0, ϕ^b, ϕ^e and γ were defined based on our observations, which we elaborate in the following discussion.

In practice, demands are known two weeks in advance, hence we set $|T| \in \{10, 12, 14\}$ to represent companies that operate five, six or seven days per week (may include Saturday/Sunday or not). The number of production lines ($|L|$) ranges from 7 to 10, employing 5 to 7 resources ($|K|$). Companies producing meat from various animals (beef, pork and poultry) have product catalogues with more than 100 items ($|J|$), while industries specialised in meat from one (or two) animals have a product catalogue with around 45 (or 80 to 90) items. We note that, when necessary for the diversity of instances, we generated integer parameters using uniform distributions (indicated as, e.g., $p \in U[a, b]$). Next, we discuss further characteristics of the test instances, where parameters are listed in the same order as in Table 1.

Table 2. Characteristics of the five classes of test instances

Class	$\|T\|$	$\|L\|$	$\|J\|$	$\|K\|$	φ^d	φ^r_k	φ^r_0	ϕ^b	ϕ^e	γ
1	10	7	45	5	100	0.8	0.6	0	0	6
2	10	10	80	6	100	0.8	0.6	0	0	8
3	14	10	90	6	90	0.6	0.5	0	0	8
4	12	10	110	7	90	0.6	0.55	100	150	8
5	14	10	110	7	90	0.6	0.55	50	150	8

- $d_{jt} \in U\left[0, \frac{C_{lt} - \min\limits_{i,j}\{st_{lij}\}\gamma - \varphi^d}{|P_l|}\right]$, with φ^d as specified in Table 2. For each period and line, the parameter φ^d introduces a slack between the capacity used to produce current period's demand and the production capacity of the line. Since each production line can not operate in all periods (due to the scarcity of resources and orders), the slack capacities of the periods a line l is assembled enable the production of the demands of the periods when l is not assembled. Hence, the smaller the value of φ^d, the more often production lines need to be assembled. The values adopted for this parameter were empirically determined in order to reproduce real-world scenarios we have observed;
- $C_{lt} = 480$ (minutes in a production day of eight hours);
- $a_{lj} = 1$ and $m_{lj} = 2$ (unit processing times do not significantly vary);
- $h_j \in U[1, 10]$ and $b_j = 10h_j$;
- $st_{lij} \in U[15, 45]$ and $sc_{lij} = 2st_{lij}$ (setup costs proportional to the setup times);
- $r_{kl} \in U[0, 2]$ when $k > 0$ (i.e., all machines) and $r_{0l} \in U[5, 10]$ (i.e., workers);
- $R_{kt} = \max\left\{\max\limits_{l=1,\ldots,L}\{r_{kl}\}, \varphi^r_k \sum\limits_{l\in[L]} r_{kl}\right\}$, with φ^r_k as specified for $k \geq 1$ and $k = 0$ in Table 2. The parameter φ^r_k is the maximum percentage of production lines that can operate simultaneously. For example, if $\varphi^r_k = 0.8$ for resource k, at most 80% of the production lines can operate simultaneously. The values adopted in Table 2 represent a range of real-world observations;
- $sl_j \in U[4, T]$;
- $ac_l = \sum_k rc_k r_{kl}$, where $rc_k \in U[\phi^b, \phi^e]$ with ϕ^b and ϕ^e as specified in Table 2. The parameter rc_k represents the unit price of resource k.When the resources are homogeneous, their costs are almost identical and hence the cost to assemble lines is not considered, i.e., $\phi^b = \phi^e = 0$ for classes 1, 2, and 3. On the other hand, an extensive product catalogue necessitates a high number of significantly different resources, hence varying values were adopted for classes 4 and 5;
- $\gamma_{lt} = \gamma$, with γ as specified in Table 2. This parameter is set to 6 when the product catalogue has less than 80 products, and 8 otherwise;
- P_l built by randomly allocating each item to a line.

5.2. Performance analysis and discussion of results

To effectively evaluate the efficiency of the proposed approaches with respect to state-of-the-art benchmarks, we consider the following computational experiments:

(1) SM: the standard model CLSDPRL was solved using the branch-and-bound

algorithm of the Cplex solver with default settings;

(2) WF: the CLSDPRL^w model was solved using the branch-and-bound algorithm of the Cplex solver with default settings;

(3) BR: the CLSDPRL^w model was solved using the branch-and-bound algorithm of the Cplex solver with the branching rule proposed in Section 4.1;

(4) RF: the relax-and-fix procedure proposed in Dillenberger et al. (1994) (and also used by James and Almada-Lobo (2011) and Sel and Bilgen (2014) recently) was applied to solve the problem;

(5) RFH: the RFH heuristic proposed in Section 4.2 based on the CLSDPRL^w model was used to solve the problem.

We implement all the mathematical models and heuristics in C++ with the library Concert Technology of the Cplex 12.6. We ran all tests on a computer with two processors (Intel Xeon E5-2680v2, 2.8GHz, 10 cores, 2 threads/core, 25 MB SmartCache) and 128 GB RAM memory. For each instance, the best feasible solution (z_f) and the best dual bound ($\overline{z}$) was obtained. The percentage deviation of the best feasible solution from the lower bound (GAP) was computed as $GAP = 100\left(\frac{z_f - \overline{z}}{z_f}\right)$.

The subproblems regarding to RF and RFH were solved by the default branch-and-bound algorithm of Cplex. The available computational time for RF was equally divided between the $|T|$ iterations. For RFH, preliminary tests indicated that the computational time to solve the subproblem of the second iteration of RFH does often not exceed 200 seconds, since this subproblem can be decomposed into $|L|$ easy subproblems as discussed before. Therefore, we fixed the time limit to solve the first iteration of RFH to $(MT - 200)$ seconds, where MT is the maximum running time in seconds. We also note that we did not parallelize the subproblems of the second iteration in order to be fair to the other methods benchmarked here.

We note that constraints (18) can replace constraints (5). Since optimization solvers may behave differently for a given formulation or approach when an equation is used instead of an inequality (e.g. due to default cut generation or heuristics working more/less effectively), we have run preliminary tests with both of these options for all proposed approaches. This enables us to provide more fair comparisons between different approaches, as we chose the more effective constraints for each method individually. For the standard model, significantly better results were obtained, in particular in class 3, using the constraints (5). On the other hand, for the RFH approach, slightly better results were obtained using constraints (18), and therefore, we replace constraints (5) by constraints (18) in the CLSDPRL^w model for the RFH procedure.

$$\sum_{j \in P_l} y_{ljt} = \delta_{lt}, \forall l, t. \tag{18}$$

Table 3 presents the computational results obtained for each class of instances with maximum running times of 1 hour and 15 minutes. The rows AGap, BGap, and WGap indicate the average, best, and worst gaps, respectively, while Time is the observed running time (in seconds), FS is the number of instances for which a feasible solution is found, and OPT is the number of instances solved until optimality. The best performances are highlighted in bold for each class of instances.

For the maximum running time of 1 hour, we observe that CLSDPRL^w (WF) performs in general better than CLSDPRL (SM), and the proposed branching rule improves the performance of the branch-and-bound algorithm of Cplex further when solving the CLSDPRL^w model. It is also worth noting that CLSDPRL proved opti-

Table 3. Computational results for 3600 and 900 seconds

Class		Maximum time = 3600 seconds					Maximum time = 900 seconds				
		SM	WF	BR	RF	RFH	SM	WF	BR	RF	RFH
1	AGap	0.11	**0.02**	**0.02**	6.30	0.08	0.16	**0.02**	**0.02**	6.31	0.09
	BGap	**0.00**	**0.00**	**0.00**	3.13	**0.00**	**0.00**	**0.00**	**0.00**	3.14	**0.00**
	WGap	0.60	**0.06**	**0.06**	15.37	1.35	0.73	**0.09**	**0.09**	15.39	1.35
	Time	2316	1293	935	80	**48**	695	484	354	80	**48**
	FS	20	20	20	20	20	20	20	20	20	20
	OPT	8	14	**17**	0	13	8	13	**14**	0	11
2	AGap	0.69	**0.09**	**0.09**	6.27	0.12	2.07	0.97	**0.13**	6.29	0.15
	BGap	0.11	0.02	**0.00**	3.95	**0.00**	0.12	0.09	**0.03**	9.82	**0.03**
	WGap	1.95	**0.23**	0.27	9.79	0.50	5.33	4.24	**0.31**	3.97	0.51
	Time	3600	3600	3600	563	**426**	900	900	900	563	**426**
	FS	20	20	20	20	20	20	20	20	20	20
	OPT	0	0	1	0	**2**	0	0	0	0	**2**
3	AGap	54.34	59.44	35.26	23.29	**10.79**	79.98	83.57	82.13	—	**29.71**
	BGap	15.92	24.62	11.14	12.84	**1.93**	66.27	58.12	66.64	—	**4.44**
	WGap	82.27	91.17	72.59	43.74	**36.26**	88.29	94.06	91.81	—	**80.77**
	Time	3600	3600	3600	**1814**	3538	900	900	900	—	**844**
	FS	19	20	20	20	20	16	19	19	0	**20**
	OPT	0	0	0	0	0	0	0	0	0	0
4	AGap	65.50	53.23	48.80	14.68	**9.01**	77.83	68.64	78.77	—	**31.31**
	BGap	58.22	26.88	15.40	8.93	**3.05**	72.45	45.27	61.16	—	**6.61**
	WGap	76.01	78.82	74.62	20.82	**16.68**	80.97	90.22	90.62	—	**55.24**
	Time	3600	3600	3600	**2838**	3600	900	900	900	—	900
	FS	8	17	16	19	**20**	7	16	17	0	**19**
	OPT	0	0	0	0	0	0	0	0	0	0
5	AGap	70.72	68.70	65.40	31.65	**15.26**	—	87.70	88.33	—	**49.60**
	BGap	54.60	45.27	51.50	19.16	**5.64**	—	83.81	87.43	—	**30.85**
	WGap	81.48	88.63	86.14	46.78	**26.49**	—	90.24	89.23	—	**70.27**
	Time	3600	3600	3600	3600	3600	900	900	900	—	900
	FS	12	12	14	10	**20**	0	6	2	0	**15**
	OPT	0	0	0	0	0	0	0	0	0	0
Mean	AGap	38.27	36.29	29.91	16.44	**7.05**	—	48.18	49.88	—	**22.17**
	BGap	25.89	19.37	15.62	12.05	**2.39**	—	37.48	43.07	—	**8.66**
	WGap	48.46	51.78	46.74	27.30	**16.26**	—	55.77	54.41	—	**41.63**
	Time	3343	3139	3067	**1779**	2242	859	817	791	—	**624**
Total	FS	79	89	90	89	**100**	63	81	78	40	**94**
	OPT	8	14	**18**	0	15	8	13	**14**	0	13

mality for only 8 instances from class 1, whereas CLSDPRL^w proved optimality for 14 instances from class 1, and CLSDPRL^w with the proposed branching rule proved optimality for 17 instances from class 1, and 1 instance from class 2. The RFH approach found the optimal solution for 13 instances from class 1, and 2 instances from class 2. Also note that RFH has near-optimal results in class 1, primarily due to its advantage that it uses the most effective formulation of WF for its first iteration.

Next, we note that for maximum running time of 1 hour, RF failed to provide feasible solutions for 11 instances (1 from class 4 and 10 from class 5), while the CLSDPRL model failed to provide feasible solutions for 21 test instances (1 from class 3, 12 from class 4, and 8 from class 5), and CLSDPRL^w was not able to find feasible solutions for 11 test instances (3 from class 4 and 8 from class 5). When our branching rule was used to solve the CLSDPRL^w model, feasible solutions were not found for 10 test instances (4 from class 4 and 6 from class 5). On the other hand, RFH provided very high quality solutions for all test instances, in particular for more challenging problems where it significantly outperforms all other approaches. RFH outperformed i) CLSDPRL both in running time and GAP for all test instances, ii) CLSDPRL^w for all classes except classes 1 and 2 (albeit with a minor difference in average gaps),

and iii) BR for the more challenging classes of 3, 4 and 5 (and with no significant difference of performance in classes 1 and 2). It is also noteworthy to remark that RFH outperforms RF substantially with respect to the average gap for all classes of test instances.

For the maximum running time of 15 minutes, similar observations can be made, except that CLSDPRL^w with our branching rule performed worse than the default CLSDPRL^w. RFH again outperformed all other methods with respect to i) the ability to find feasible solutions, ii) the quality of solutions found, and iii) the running time. We also note that there are a few instances from classes 4 and 5, for which no method could find a feasible solution with this maximum running time.

Table 4. Average dual bounds (ADB) and number of instances that each method provided the best dual bound (NBB)

Maximum running time = 3600 seconds										
	SM		WF		BR		RF		RFH	
Class	ADB	NBB	ADB	NBB	ADB	NBB	ADB	NBB	ADB	NBB
1	50874	3	50927	12	50927	18	18094	0	50168	0
2	63202	0	63543	7	63545	13	22882	0	62560	0
3	113226	0	113719	0	114293	2	31853	0	115818	18
4	172026	0	173906	3	174662	4	43623	0	176031	13
5	208449	1	209746	1	211573	7	47447	0	211981	11
Mean	121555	0.8	122368	4.6	123000	8.8	32780	0	123312	8.4
Maximum running time = 900 seconds										
1	50845	3	50923	10	50924	17	18094	0	50168	0
2	62729	0	63172	4	63529	16	22882	0	62560	0
3	113430	1	113228	2	113979	7	—	0	113835	10
4	171400	0	172240	2	172630	4	—	0	173360	14
5	—	0	204676	1	206248	1	—	0	209415	18
Mean	—	0.8	120848	3.8	121462	9	—	0	121867	8.4

In Table 4, we present, for each class and method, the average dual bounds (ADB) and the number of instances that each method provided the best dual bound (NBB). We observe that for instances from classes 1 and 2, the proposed branching rule provided best dual bounds for a greater number of instances than the other methods. On the other hand, for instances from the remaining three classes, RFH outperforms all other methods with respect to both ADB and NBB, indicating that it is not only a superior method for obtaining very high quality solutions but also dual bounds, in particular when problems are computationally challenging. It is in particular impressive that the quality of dual bounds obtained by RFH is competitive with the quality of dual bounds obtained by BR even when BR is superior. To the best of our knowledge, this is the first paper showing very promising use of a relax-and-fix procedure for obtaining dual bounds.

Table 5. Results for instances from class 5 with maximum running time = 3 hours.

Class 5 - Maximum running time = 3 hours							
Approach	AGap	BGap	WGap	Time	FS	ADB	NBB
SM	36.11	13.39	75.05	10800	17	208956	0
WF	51.87	16.57	83.57	10800	14	214521	1
BR	34.83	9.87	71.55	10800	17	213188	2
RF	16.02	9.65	21.13	**3880**	18	47537	0
RFH	**7.25**	**2.13**	**14.98**	10186	**20**	**215362**	**17**

We have also run tests for the 20 challenging instances from class 5 with maximum running time of 3 hours in order to provide further insights with respect to the capabilities of all methods compared. As the results in Table 5 demonstrate, the standard

model (SM) performed better than the CLSDPRL^w model (WF), while the branching rule (BR) achieved slightly better results than the standard model. The RF heuristic spent significantly less time than other methods, albeit with no feasible solution for 2 instances, and extremely weak dual bounds relative to other methods. The RFH heuristic outperformed other methods with respect to the quality of solutions (including the best and worst cases), the number of instances with a feasible solution found, and the quality of the dual bounds.

5.3. Impact of the novel features and sensitivity analysis

5.3.1. Impact of the novel features

Next, we study the impact on the solution structure generated by the main novel characteristics, namely scarce production resources and perishable products. Therefore, we modify the CLSDPRL^w formulation to obtain two different problems:

i) a problem without the scarcity of the production resources; and
ii) a problem without perishable products.

In order to obtain a problem without scarce production resources, we only need to remove constraints (10). On the other hand, in order to obtain a problem without perishable products, we need to remove the continuous variables $ve_{jt}^{t'}$ and $q_{jt}^{t'}, \forall j, t, t'$ and the associated constraints (11)-(14), and replace constraints (2) by constraints (19).

$$I_{j,t-1} - B_{j,t-1} + \sum_{l} x_{ljt} = d_{jt} + I_{jt} - B_{jt}, \forall j, t. \tag{19}$$

We have tested these models on the same data set from Section 5.1, using the branch-and-cut algorithm of Cplex with the branching rule (BR) and maximum running time of 1 hour. Table 6 presents the results obtained for each class and problem, where we report for the modified models, in line with Table 3, gaps (average, worst, and best), running time, number of instances with a feasible solution, and number of instances optimally solved. In addition, for all models, we present aspects referring to the structure of the solutions; more specifically, inventory holding cost (HCost), backlogging cost (BCost), setup cost (SCost), cost to assemble production lines (ACost), and total cost (TCost). Recall from Table 2 that the coefficients for costs to assemble production lines is et to 0 for classes 1, 2, and 3, hence the associated entries are marked with "-".

Gaps and running times particularly indicate that, except for class 2, the problem without scarce resources is much easier to solve. This is in particular the case for larger classes of 3, 4 and 5, where feasible solutions found have relatively very small gaps. On the other hand, the problem without the perishability of products is easier to solve than the original problem, however, the instances from classes 3, 4, and 5 still remain challenging. Therefore, the most difficult novel characteristic is the scarce production resources.

Based on cost structures of solutions, we observe that without scarce resources, setup and assembly costs increase, while inventory holding and backlogging costs significantly decrease. This is not unexpected, as no restriction on resources allow more setups and assembly of production lines in exchange for savings in backlogging and inventory. On the other hand, in the case without perishable products, we note that solutions are very similar to solutions of the original problem for instances from classes 1 and 2, while

Table 6. Impact of scarce resources and perishable products.

	Without scarce resources					
Class	1	2	3	4	5	Mean
AGap	0.01	0.31	0.60	0.21	0.93	0.41
BGap	0.00	0.00	0.36	0.03	0.09	0.10
WGap	0.06	0.68	1.10	0.51	2.76	1.02
Time	236	3294	3600	3480	3451	2812
FS	20	20	20	20	16	19.2
OPT	19	2	0	0	0	4.2
SCost	11555	20042	30520	19729	21358	20641
HCost	2863	3823	7209	43846	58061	23161
BCost	1826	1264	2189	5072	5529	3176
ACost	-	-	-	73396	95665	33812
TCost	16244	25129	39918	142043	180612	80790
	Without perishable products					
AGap	0.01	0.06	17.95	20.67	51.00	17.94
BGap	0.00	0.00	0.77	4.53	8.92	2.84
WGap	0.04	0.22	45.31	51.39	80.54	35.50
Time	644	3032	3600	3600	3600	2895
FS	20	20	20	20	20	20
OPT	17	5	0	0	0	4.4
SCost	7536	14040	17586	16047	16023	14246
HCost	21858	25701	67030	66475	109860	58185
BCost	21527	23855	61615	86137	349338	108494
ACost	-	-	-	59227	82506	28347
TCost	50921	63596	146231	227886	557728	209272
	Original problem (BR)					
SCost	7540	14052	17026	15936	15585	14028
HCost	21862	25694	73654	80922	120595	64546
BCost	21533	23857	106034	277708	701624	226151
ACost	-	-	-	58762	82034	28159
TCost	50935	63603	196715	433328	919837	332884

for other classes, backlogging costs are significantly reduced (with other costs only marginally varying). This can be explained that perishability of products prohibits stocking for too long (and hence motivating backlogging instead). Since both of these modified problems are essentially relaxations of the original problem, total costs are lower, though this difference is more striking for the case without scarce resources.

5.3.2. Sensitivity analysis

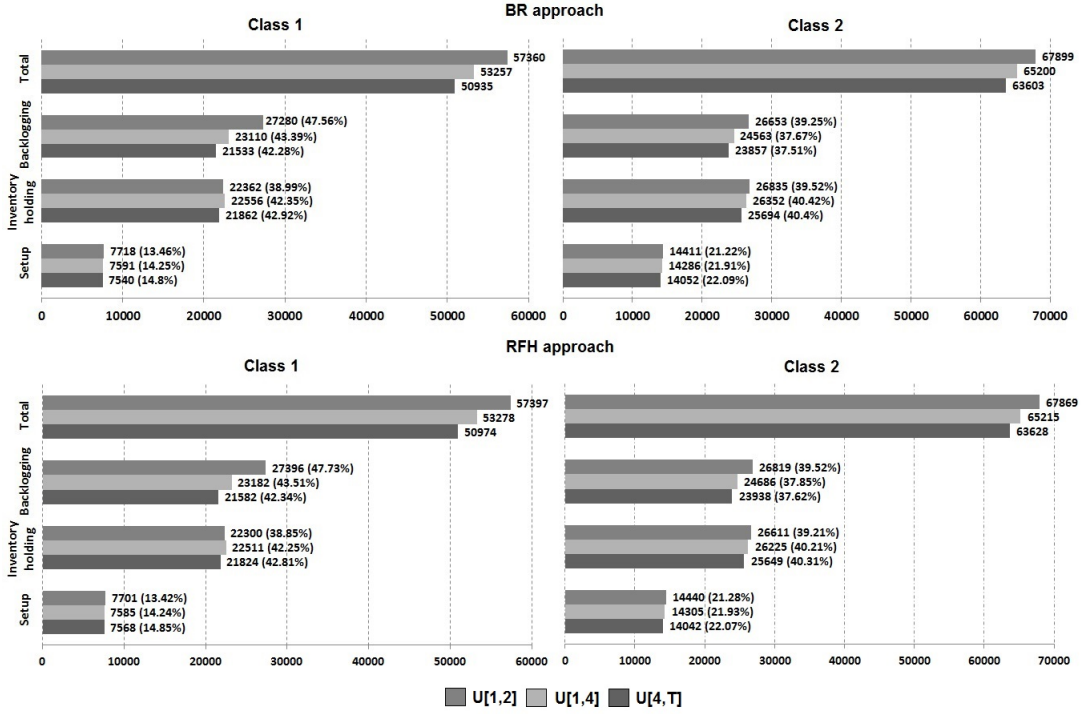
Next, we perform sensitivity analysis to study the impact of key parameters on the structure of the solutions as well as on the computational performance of RFH and BR. Specifically, we investigate parameters sl_j and ac_l testing on classes 1 and 2.

Sensitivity analysis with respect to shelf-life sl_j:

As noted in Table 2, sl_j was specified to be in $U[4, T]$. For our analysis, we also consider $sl_j \in U[1, 4]$, and $sl_j \in U[1, 2]$ for the test instances of classes 1 and 2.

In Figure 3, each chart presents in the order of total, backlogging, inventory holding and setup costs (as well as their percentage in the total) for each approach and range of shelf-life parameters. We consistently observe for both approaches that the shorter the shelf-life of products, the significantly higher the backlogging costs (both in absolute and percentage terms). On the other hand, while other costs present slight variations,

Figure 3. Sensitivity analysis for sl_j: BR (top) and RFH (bottom)



total costs always increase with shorter shelf-life, e.g., in class 1, total costs increase around 4.5% from $sl_j \in U[4,T]$ to $sl_j \in U[1,4]$, and further 7.7% from $sl_j \in U[1,4]$ to $sl_j \in U[1,2]$.

In Table 7, we present the quality of the solutions found by each method, class, and range of shelf-life. In general, RFH was able to find good quality solutions, albeit with gaps slightly worse in comparison to BR (except for $sl_j \in U[1,2]$ in class 2). On the other hand, running times of RFH are significantly better than running times of BR. Considering mean gaps and running times, in particular for RFH, the problem becomes more challenging using smaller values of shelf-life.

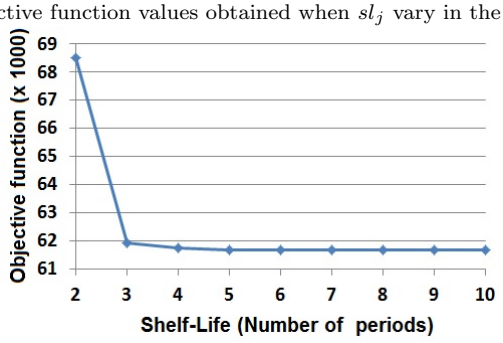
Table 7. Sensitivity analysis for sl_j (computational impact)

Class	$sl_j \in$	BR				RFH			
		AGap	BGap	WGap	Time	AGap	BGap	WGap	Time
1	$U[4,T]$	0.02	0.00	0.06	935	0.08	0.00	1.35	48
	$U[1,4]$	0.02	0.00	0.14	1091	0.06	0.00	0.62	55
	$U[1,2]$	0.01	0.01	0.09	885	0.08	0.01	1.04	112
2	$U[4,T]$	0.09	0.00	0.27	3600	0.12	0.00	0.50	426
	$U[1,4]$	0.09	0.01	0.22	3408	0.11	0.01	0.31	551
	$U[1,2]$	0.47	0.01	2.68	3559	0.39	0.01	1.65	1052
Mean	$U[4,T]$	0.06	0.00	0.17	2268	0.10	0.00	0.93	237
	$U[1,4]$	0.05	0.01	0.18	2249	0.08	0.01	0.46	303
	$U[1,2]$	0.24	0.01	1.39	2222	0.24	0.01	1.34	582

Finally, to demonstrate the impact of shelf-life on the objective function value, we have randomly selected an instance from class 1 (T = 10) and tested the cases of $sl_j = 2, \ldots T$. The optimal objective function value with respect to shelf-life is shown in Figure 4, where the objective function value significantly decreases when the shelf-life increases from 2 to 3 periods for this specific instance. Once $sl_j = 5$ is reached, the objective function value is stabilized. Similar behaviour can be observed with other

instances as well.

Figure 4. Objective function values obtained when sl_j vary in the set $\{2, 3, \ldots, T\}$

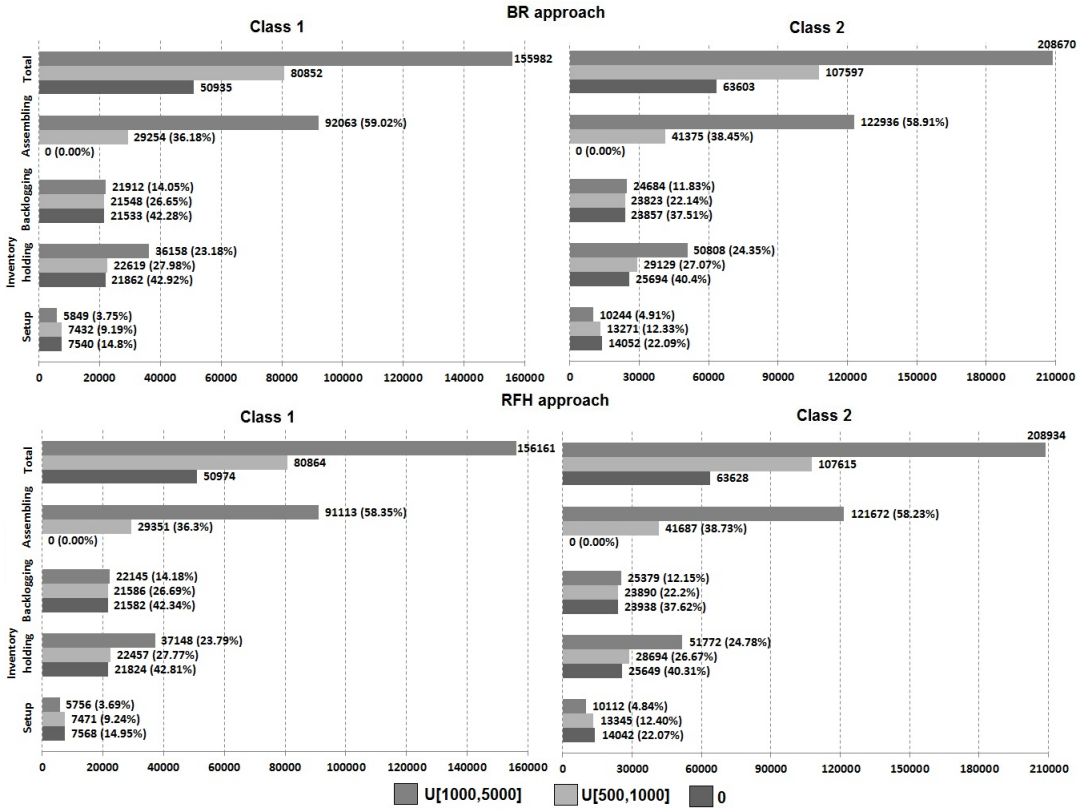


Sensitivity analysis with respect to cost to assemble production lines (ac_l):

Originally, ac_l was set to 0 for all instances of classes 1 and 2. For our analysis, we consider $ac_l \in U[500, 1000]$ and $ac_l \in U[1000, 5000]$.

Similar to previous analysis, in Figure 5, each chart presents in the order of total, assemble, backlogging, inventory holding and setup costs (as well as their percentage in the total) for each approach and range of assemble cost.

Figure 5. Sensitivity analysis for ac_l: BR (top) and RFH (bottom)



As ac_l values increase, besides the assemble costs, the inventory holding costs also increase, while the setup cost decrease and the backlogging cost remain stable (though its percentage decreases). As higher ac_l values discourage assemble of production lines, this directly reduces setups of products and encourages more inventory to be carried.

In order to analyze the computational performance of the methods, we present in Table 8 the average, best, and worst gaps, and the running times for each method, class, and range of ac_l values. In general, RFH provides high quality solutions that are slightly worse than the solutions found by BR, while the running times of RFH are significantly smaller than the running times of BR. An interesting observation is that while increasing ac_l values result in more challenging problems for RFH, the exact opposite behaviour is seen for BR.

Table 8. Sensitivity analysis for ac_l (computational impact)

Class	$ac_l \in$	BR				RFH			
		AGap	BGap	WGap	Time	AGap	BGap	WGap	Time
1	{0}	0.02	0.00	0.06	935	0.08	0.00	1.35	48
	U[500,1000]	0.01	0.00	0.01	352	0.02	0.00	0.15	54
	U[1000,5000]	0.01	0.00	0.01	257	0.13	0.00	0.80	181
2	{0}	0.09	0.00	0.27	3600	0.12	0.00	0.50	426
	U[500,1000]	0.04	0.01	0.11	2898	0.06	0.01	0.16	328
	U[1000,5000]	0.27	0.01	2.43	3048	0.41	0.01	1.63	1567
Mean	{0}	0.06	0.00	0.17	2268	0.10	0.00	0.93	237
	U[500,1000]	0.03	0.00	0.06	1625	0.04	0.00	0.15	191
	U[1000,5000]	0.14	0.00	1.22	1652	0.27	0.00	1.21	874

Sensitivity analysis with respect to parameter φ^d:

As noted in Table 2, the value of parameter φ^d was set to 100 in classes 1 and 2, in order to represent real-world scenarios observed in the industry. In order to perform sensitivity analysis with respect to this parameter, additional computational tests were performed using $\varphi^d = 60$ and $\varphi^d = 20$.

As seen with generally increased times and gaps for both methods and classes in Table 9, the problem becomes computationally more difficult to solve when the value of φ^d is decreased.For all considered values of φ^d, RFH provides high quality solutions that are on average slightly worse than the solutions found by BR, albeit with significantly smaller times than BR. Moreover, the worst GAP obtained by RFH is significantly better than the worst GAP obtained by BR when $\varphi^d = 20$.

Table 9. Sensitivity analysis for φ^d (computational impact)

Class	φ^d	BR				RFH			
		AGap	BGap	WGap	Time	AGap	BGap	WGap	Time
1	100	0.02	0.00	0.06	935	0.08	0.00	1.35	48
	60	0.02	0.01	0.08	784	0.22	0.01	1.30	78
	20	0.02	0.01	0.07	1051	0.19	0.01	0.86	161
2	100	0.09	0.00	0.27	3600	0.12	0.00	0.50	426
	60	0.13	0.01	0.86	3600	0.26	0.01	1.02	536
	20	0.64	0.05	5.78	3600	0.74	0.06	2.81	1242
Mean	100	0.06	0.00	0.17	2268	0.10	0.00	0.93	237
	60	0.07	0.01	0.47	2192	0.24	0.01	1.16	307
	20	0.33	0.03	2.93	2326	0.46	0.04	1.84	701

Next, in order to study the impact of the parameter φ^d on the structure of the solutions found by the proposed approaches, Table 10 presents the percentages of the setup cost (SCost), inventory holding cost (HCost), and backlogging cost (BCost) in the total cost (TCost). We observe that for both approaches and classes, when the value of φ^d decreases, the percentage of the setup cost decreases and the percentage of the backlogging cost increases, while the percentage of the inventory holding cost remain stable. Obviously, when φ^d is reduced, the demands (and hence total cost) increase.

Table 10. Sensitivity analysis for φ^d (impact on the structure of the solutions)

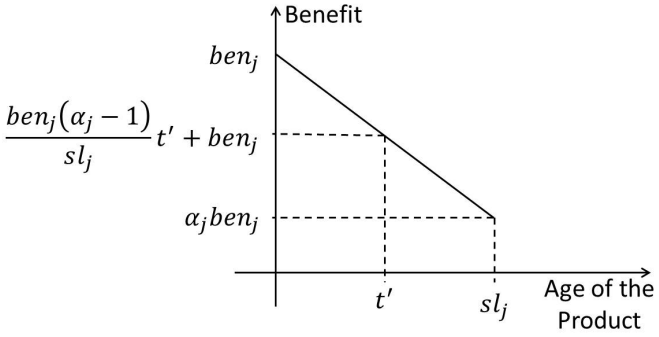
Class	φ^d	BR				RFH			
		SCost(%)	HCost(%)	BCost(%)	TCost	SCost(%)	HCost(%)	BCost(%)	TCost
1	100	14.80	42.92	42.28	50935	14.85	42.81	42.34	50974
	60	12.89	42.43	44.69	58597	12.85	42.22	44.93	58721
	20	10.40	40.64	48.97	71532	10.33	40.57	49.10	71654
2	100	22.09	40.40	37.51	63603	22.07	40.31	37.62	63628
	60	19.59	40.88	39.52	71950	19.44	40.77	39.79	72471
	20	16.88	41.20	41.92	83360	16.85	40.99	42.16	83448
Mean	100	18.85	41.52	39.63	57269	18.86	41.42	39.72	57301
	60	16.58	41.58	41.84	65273	16.49	41.42	42.09	65596
	20	13.89	40.94	45.17	77446	13.84	40.79	45.37	77551

5.4. An extension with residual shelf-life pricing

The model presented in Section 3 is suitable to address production planning problems faced by various food industries such as meat, where the only concern is the expiration of the products due to shelf-life, as also observed in our technical visits. However, as discussed in Lütke Entrup, Günther, Van Beek, Grunow, and Seiler (2005), some industries such as short-life dairy may benefit from a different pricing approach, where the price of a product depends on its residual shelf-life by the time it is delivered to the customer. Therefore, in this section, we discuss an extension of our model to include shelf-life dependent pricing, and we perform computational tests to study the efficiency of our proposed approaches in this setting.

Lütke Entrup et al. (2005) consider a lot sizing and scheduling problem with shelf-life dependent pricing in the yoghurt production. The authors assume that the manufacturer yields a financial benefit if the products have a longer residual shelf-life when delivered. More specifically, they propose a shelf-life dependent benefit as a linear function of the age of the product. Let ben_j be the benefit when one unit of the product j of age 0 is used to meet demand, and α_j be the fraction of ben_j when one unit of age sl_j is delivered to the customer. Figure 6 presents the benefit function, including the expression of the benefit when one unit of age $t^{'}$, $0 \leq t^{'} \leq sl_j$, is used to meet demand.

Figure 6. Benefit obtained according to the age of the products



In this extended problem, the objective is to maximize the total benefits obtained, discounting the total costs described in Section 3. Therefore, the original objective function (1) is replaced by (20), and the new model is denoted by MP-CLSDPRL (Maximum Profit CLSDPRL).

$$\max \sum_{j,t,t'} \left(\frac{ben_j(\alpha_j - 1)}{sl_j} t' + ben_j \right) q_{jt}^{t'} - \sum_{t,j} (h_j I_{jt} + b_j B_{jt}) - \sum_{l,t,i,j} sc_{lij} z_{lijt} - \sum_{l,t} ac_l \delta_{lt} \quad (20)$$

We have modified the test instances of Section 5.1 with the addition of $ben_j \in U[5, 50]$ and $\alpha_j = 0.2$ in order to test the MP-CLSDPRL model. We have only considered the solution approaches BR and RFH with a maximum running time of 1 hour. Results presented in Table 11 indicate that our heuristic offers superior quality feasible solutions for classes 3, 4, and 5, overcoming the solution quality issue observed for the BR approach, while also obtaining almost identical solution quality for classes 1 and 2. Moreover, our heuristic is able to provide competitive dual bounds, in particular performing significantly better than the BR approach for the large size instances of classes 3, 4, and 5. Finally, we note that RFH is able to provide feasible solutions for all test instances (finding solutions with non-positive profits only for two instances from class 5), while BR can provide feasible solutions only for 86 test instances (finding solutions with non-positive profits for 29 test instances, more specifically, 4 from class 3, 13 from class 4, and 12 from class 5).

Table 11. Computational results for the MP-CLSDPRL model

Class	BR						RFH					
	AGap	BGap	WGap	Time	FS	NBB	AGap	BGap	WGap	Time	FS	NBB
1	0.01	0.00	0.01	443	20	20	0.02	0.00	0.07	43	20	0
2	0.02	0.01	0.07	2838	20	20	0.02	0.01	0.08	452	20	0
3	73.45	7.00	482.36	3600	20	1	5.20	0.82	21.18	3543	20	19
4	149.35	43.89	446.85	3600	14	3	10.83	4.23	18.70	3584	20	17
5	200.82	150.36	513.63	3600	12	8	40.09	9.79	192.63	3600	20	12
Mean	84.73	40.25	288.59	2816	17.2	10.4	11.23	2.97	46.53	2244	20	9.6

To conclude this section, we also compare the results obtained by the original CLSDPRL model (Table 3, max. time 3,600 seconds) with the results of the MP-CLSDPRL model. We first observe that for instances in classes 1 and 2, the gaps and running times of each method are very similar for both CLSDPRL and MP-CLSDPRL models. On the other hand, we remark that for instances of classes 3, 4, and 5, the BR approach presents significantly higher gaps for MP-CLSDPRL than the gaps for CLSDPRL, which is primarily driven by the fact that the BR approach was not able to find solutions with positive profit for some instances of the MP-CLSDPRL model. A similar gap difference is also observed for the RFH approach within the instances from class 5, where RFH was not able to find solutions with positive profit for some instances of the MP-CLSDPRL model, hence resulting in a higher gaps for MP-CLSDPRL. Finally, we note that for instances of classes 3 and 4, RFH is able to achieve similar gaps for both models.

6. Conclusion and future research

In this paper, we described a new lot sizing and scheduling problem stemming from the food industry and proposed mixed integer programming formulations to model this. In order to build effective solution methods for use in practice, a branching rule to accelerate the convergence of branch-and-bound algorithms to solve the proposed models as

well as a relax-and-fix procedure with a nontraditional decomposition approach were proposed. The computational results indicate that in particular the proposed heuristic framework can obtain superior results for challenging problems.

There are a number of directions for potential future research. First of all, we note that the nature of the problem with two main components, a lot sizing problem and a scheduling problem integrated together with linking constraints, deserves interest with regards to developing and evaluating customized decomposition approaches. In an ongoing work, we are currently investigating Lagrangian relaxation based heuristic approaches, which may provide further insights into the complexities of the problem. Moreover, long computational times suggest that other heuristic and metaheuristic approaches such as genetic algorithms and variable neighborhood search are worth investigating from a practical and comparative perspective. Finally, the integrated nature of the problem also requires a better theoretical understanding and therefore, in the future, we are planning to perform a thorough polyhedral analysis on some specific subproblems involving integrating constraints.

References

Afzalirad, M., & Rezaeian, J. (2016). Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions. *Computers & Industrial Engineering*, *98*, 40–52.

Akartunalı, K., Fragkos, I., Miller, A., & Wu, T. (2016). Local cuts and two-period convex hull closures for big-bucket lot-sizing problems. *INFORMS Journal on Computing*, *28*(4), 766–780.

Akartunalı, K., & Miller, A. (2009). A heuristic approach for big bucket multi-level production planning problems. *European Journal of Operational Research*, *193*, 396–411.

Almada-Lobo, B., Clark, A., Guimarães, L., Figueira, G., & Amorim, P. (2015). Industrial insights into lot sizing and scheduling modeling. *Pesquisa Operacional*, *35*(3), 439–464.

Almeder, C., & Almada-Lobo, B. (2011). Synchronisation of scarce resources for a parallel machine lotsizing problem. *International Journal of Production Research*, *49*(24), 7315–7335.

Brahimi, N., Absi, N., Dauzère-Pérès, S., & Nordli, A. (2017). Single-item dynamic lot-sizing problems: An updated survey. *European Journal of Operational Research*, *263*(3), 838 - 863.

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

Costa, A. M., dos Santos, L. M. R., Alem, D. J., & Santos, R. H. (2014). Sustainable vegetable crop supply problem with perishable stocks. *Annals of Operations Research*, *219*(1), 265–283.

Dillenberger, C., Escudero, L. F., Wollensak, A., & Zhang, W. (1994). On practical resource allocation for production planning and scheduling with period overlapping setups. *European Journal of Operational Research*, *75*(2), 275–286.

Doostmohammadi, M., & Akartunalı, K. (2018). Valid inequalities for two-period relaxations of big-bucket lot-sizing problems: Zero setup case. *European Journal of Operational Research*, *267*(1), 86 – 95.

Glock, C. H., Grosse, E. H., & Ries, J. M. (2014). The lot sizing problem: A tertiary study. *International Journal of Production Economics*, *155*, 39–51.

Güngör, M., Ünal, A. T., & Taşkın, Z. C. (2017). A parallel machine lot-sizing and scheduling problem with a secondary resource and cumulative demand. *International Journal of Production Research*, 1–14.

Haase, K. (1996). Capacitated lot-sizing with sequence dependent setup costs. *OR Spectrum*,

18(1), 51–59.

James, R. J., & Almada-Lobo, B. (2011). Single and parallel machine capacitated lotsizing and scheduling: New iterative mip-based neighborhood search heuristics. *Computers & Operations Research*, *38*(12), 1816–1825.

Lütke Entrup, M., Günther, H.-O., Van Beek, P., Grunow, M., & Seiler, T. (2005). Mixed-integer linear programming approaches to shelf-life-integrated planning and scheduling in yoghurt production. *International Journal of Production Research*, *43*(23), 5071–5100.

Meyr, H., & Mann, M. (2013). A decomposition approach for the general lotsizing and scheduling problem for parallel production lines. *European Journal of Operational Research*, *229*(3), 718–731.

Özdamar, L., & Birbil, Ş. I. (1998). Hybrid heuristics for the capacitated lot sizing and loading problem with setup times and overtime decisions. *European Journal of Operational Research*, *110*(3), 525–547.

Sel, Ç., & Bilgen, B. (2014). Hybrid simulation and mip based heuristic algorithm for the production and distribution planning in the soft drink industry. *Journal of Manufacturing Systems*, *33*(3), 385–399.

Smith-Daniels, V., & Ritzman, L. P. (1988). A model for lot sizing and sequencing in process industries. *The International Journal of Production Research*, *26*(4), 647–674.

Soler, W. A. O., & Santos, M. O. (2017). A new branching rule to solve the capacitated lot sizing and scheduling problem with sequence dependent setups. *Trends in Applied and Computational Mathematics*, *18*, 515–529.

Suerie, C., & Stadtler, H. (2003). The capacitated lot-sizing problem with linked lot sizes. *Management Science*, *49*(8), 1039–1054.

Toledo, C. F. M., da Silva Arantes, M., Hossomi, M. Y. B., França, P. M., & Akartunalı, K. (2015). A relax-and-fix with fix-and-optimize heuristic applied to multi-level lot-sizing problems. *Journal of Heuristics*, *21*(5), 687–717.

Villa, F., Vallada, E., & Fanjul-Peyro, L. (2018). Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource. *Expert Systems with Applications*, *93*, 28–38.

Xiao, J., Yang, H., Zhang, C., Zheng, L., & Gupta, J. N. (2015). A hybrid lagrangian-simulated annealing-based heuristic for the parallel-machine capacitated lot-sizing and scheduling problem with sequence-dependent setup times. *Computers & Operations Research*, *63*, 72–82.