Progressive Selection Method for the Coupled Lot-Sizing and Cutting-Stock Problem

Tao Wu
Advanced Analytics Dept., Dow Chemical, 48642 Midland, MI, USA, danielwu9999@gmail.com

Kerem Akartunalı
Dept. of Management Science, University of Strathclyde, Glasgow, G4 0GE, UK, kerem.akartunali@strath.ac.uk

Raf Jans
Dept. of Logistics and Operations Management, HEC Montréal, H3T 2A7 Montréal (Québec), Canada, raf.jans@hec.ca

Zhe Liang
School of Economics & Management, Tongji University, Shanghai, 200092, China, liangzhe@tongji.edu.cn

The coupled lot-sizing and cutting-stock problem has been a challenging and significant problem for industry, and has therefore received sustained research attention. The quality of the solution is a major determinant of cost performance in related production and inventory management systems, and therefore there is intense pressure to develop effective practical solutions. In the literature, a number of heuristics have been proposed for solving the problem. However, the heuristics are limited in obtaining high solution qualities. This paper proposes a new progressive selection algorithm that hybridizes heuristic search and extended reformulation into a single framework. The method has the advantage of generating a strong bound using the extended reformulation, which can provide good guidelines on partitioning and sampling in the heuristic search procedure so as to ensure an efficient solution process. We also analyze per-item and per-period Dantzig–Wolfe decompositions of the problem and present theoretical comparisons. The master problem of the per-period Dantzig–Wolfe decomposition is often degenerate, which results in a tailing-off effect for column generation. We apply a hybridization of Lagrangian relaxation and stabilization techniques to improve the convergence. The discussion is followed by extensive computational tests, where we also perform detailed statistical analyses on various parameters. Comparisons with other methods indicate that our approach is computationally tractable and is able to obtain improved results.

Key words: Integer programming; Cutting-stock; Lot-sizing; Heuristics; Column generation; Dantzig–Wolfe decomposition.

---

1. Introduction

Lot-sizing and cutting-stock are two fundamental problems that often arise in the production processes of many industries including furniture, paper, packaging, aluminum window frame, fiber glass, and steel manufacturing. The lot-sizing problem determines production quantities for a number of items over a finite horizon, taking into account the fundamental trade-off between setup and inventory-holding costs. On the other hand, the cutting-stock problem determines patterns of cutting raw materials in stock to produce final products or sub-assemblies for assembling final products, where trim loss costs are often associated with these decisions. In the literature, these two problems are often addressed independently. However, in the current global manufacturing environment, manufacturers are under intense pressure to meet customers' demand in time with the lowest costs and hence integrating such problems for more "globally optimal" solutions gives a company a competitive edge. As such, optimizing production and cutting decisions simultaneously in a coupled lot-sizing and cutting-stock problem offers significant savings potential by capturing trade-offs among trim loss, inventory-holding, and setup costs.

Since the 1970's, researchers and practitioners experimented with integrating lot-sizing decisions into the cutting-stock problems by considering setup costs for pattern changes in the cutting process. The work of [29] proposed a heuristic procedure for scheduling cutting operations with an objective to minimize the total costs of setups and trim loss in the paper industry, and [30] presented a mathematical formulation for a one-dimensional cutting-stock problem with setup costs incurred for pattern changes. The heuristic procedure of [29] was further developed by [14], and [31] proposed some approaches for solving the one-dimensional roll trim problem with setup costs. The Linear Programming (LP)-based approach of [12] considered a two-stage cutting-stock problem involving a number of setups, and [35] evaluated four different approaches for solving the cutting-stock problem with setup costs for pattern changes. The generalized cutting-stock problem with setup costs for pattern changes was also more recently investigated by [20], and the minimization of the number of different patterns has also been used as an objective by [46].

The coupled cutting-stock and production planning problem has also been studied in the literature. [42] studied the process of importing tree trunks and cutting them into assortments and boards for various markets, and [34] proposed a two-step procedure for a coupled lot-sizing and cutting-stock problem in the paper industry. The coupled problem has been studied with different approaches in various industry settings, such as in the copper manufacturing industry (see, e.g., [32]), cutting of steel plates (see, e.g., [38]), off-road truck companies (see, e.g., [39]) and production of gear belts (see, e.g., [8]). Further work in this area can also be found in [37, 41, 44] and [45]. Finally, we note [10] and [22] as general references on heuristics and meta-heuristics for the interested reader, since these methods provide the backbone of many of the approaches discussed, as well as part of the approach we propose in this paper.

Early research (e.g., [8], [39], and [41]) is valuable for understanding underlying structures of the coupled problem. However, these studies were limited on a number of aspects, such as disregarding capacity constraints, inventory-holding costs, and the assembly process of final items in formulating these problems, and using a two-phase procedure in solving them. Some of these concerns have been partly addressed in the more recent literature. [26] analyzed the trade-off among trim loss, inventory-holding and setup costs that arises when solving the cutting-stock problem by taking into account production planning for various periods. The authors formulated a mathematical model and a solution approach for the coupled lot-sizing and cutting-stock problem. [27] proposed a Lagrangian relaxation-based heuristic for the coupled lot-sizing and cutting-stock problem. However, the difficulty faced by this heuristic is that the resulting subproblems are $\mathcal{NP}$-hard capacitated lot-sizing problems. [28] relax setups for the problem but include the storage of parts in order to solve the problem more effectively. We also note the very recent work of [4] proposing robust models for coupled lot-sizing and cutting-stock in the furniture industry.

The problem studied in this paper, the coupled **L**ot-**S**izing and **C**utting-**S**tock (LS-CS) problem, is similar to the one analyzed by [27]. Its goal is to plan the production of a number of items over a horizon of finite periods, while these items are produced by cutting rectangular sheets into standard items with different sizes. These items are assembled to make a number of different final products with external demand to be met on time. The objective of the problem is to find a minimum cost production plan without violating capacity limitations, where the total cost includes setup cost, inventory-holding cost, cutting cost, and raw material cost. The effective solution of this problem is an important determinant of cost performance in related production and inventory control systems, which include the well-known material requirements planning systems prevalent in manufacturing practice. Research on finding effective models and solution methods thus has the potential to provide tangible benefits in the form of lower total production-related costs for problems in this class.

This paper proposes a progressive selection (PS) method which combines extended reformulation with heuristics in order to leverage the strengths of both methods. The proposed PS approach is designed to first generate an initial population of bounding solutions using extended reformulation and heuristics. Then, the approach utilizes domain knowledge derived from these solutions to iteratively generate and evaluate subproblems through a progressive selection strategy. These subproblems have a number of integer variables fixed and therefore can be efficiently solved. The goal of the PS approach is that, by solving these subproblems, objective values of the original problem can be iteratively improved and eventually converge to a solution of good quality. The application of the PS method to the LS-CS problem shows that the method is able to obtain competitive results while being computationally tractable. In addition, this paper proposes per-item and per-period Dantzig–Wolfe decompositions to improve lower bounds for the LS-CS problem and presents theoretical comparisons. The master problem of the per-period Dantzig–Wolfe decomposition is often degenerate, which results in a tailing-off effect for column generation. A hybridization of Lagrangian relaxation and stabilization techniques is applied to improve the convergence.

The remainder of this paper is organized as follows. In Section 2, we present several mathematical formulations for the LS-CS problem. In Section 3, we discuss Dantzig–Wolfe decomposition and column generation for the LS-CS problem. In Sections 4 and 5, we describe in detail the PS method. Section 6 presents statistical analyses of the PS method for its application to the LS-CS problem. Section 7 presents extensive computational test results and comparisons with other approaches from the literature. Finally, we conclude with future directions in Section 8.

2. Alternative formulations for the LS-CS problem

This paper considers a problem that is similar to the one discussed in [27], with the additional complexity of integrality constraints enforced on the number of plates. Another difference is the capacity constraints: this paper refers to time availability whereas [27] refer to the amount of material area that can be cut by a saw machine. These two factors are convertible if we know the cutting speed for the saw machine. However, we decided to use time capacity due to the benefit of allowing setup times in the capacity constraints as is the practice in the area. Furthermore, we assume that setup time and costs are not dependent on the sequence, all initial inventories are zero, all production costs are linear, all holding costs are time-invariant, lead times are zero, and demands need to be satisfied on time (i.e., no backlogging allowed). The cutting pattern is assumed to be rectangular, and no inventory of sub-assemblies or plates is considered. Before presenting different formulations, we define our notation:

Sets:

T	Number of periods in the planning horizon, indexed by q, t, f and ℓ.
I	Number of items (final products), indexed by i.
J	Number of sub-items (sub-assemblies), indexed by j.
P	Number of possible cutting patterns, indexed by p.

Parameters:

uc	Cost of a piece of plate.
r_{ji}	Number of sub-items of type j needed to produce one unit of item i.
pc_{it}	Unit production cost of item i in period t.
sc_{it}	Setup cost for producing item i in period t.
st_{it}	Setup time for producing item i in period t.
hc_i	Inventory-holding cost for one unit of item i per period.
a_{jp}	Number of sub-items of type j made by cutting pattern p.
v_{it}	Unit production time of item i in period t.
d_{it}	Gross demand for item i in period t.
C_t	Assembly capacity for producing final items in period t.
BM_{it}	A big number for each product i and period t.

Decision variables:

x_{it}	Number of units of item i produced in period t.
s_{it}	Inventory of item i at the end of period t.
z_{pt}	Number of plates used with cutting pattern p in period t.
y_{it}	Setup decision variables: $y_{it} = 1$ if $x_{it} > 0$, zero, otherwise.

We first present the most basic and intuitive problem formulation, i.e., the **L**ot-**S**izing and **C**utting-**S**tock formulation (LSCS), as follows:

LSCS:

$$\min \sum_{i=1}^{I}\sum_{t=1}^{T}(pc_{it}\cdot x_{it} + hc_i \cdot s_{it} + sc_{it}\cdot y_{it}) + \sum_{p=1}^{P}\sum_{t=1}^{T} uc\cdot z_{pt} \tag{1}$$

Subject to:

$$x_{it} + s_{i,t-1} - s_{it} = d_{it} \qquad \forall\, i \in \{1,...,I\},\ t \in \{1,...,T\} \tag{2}$$

$$\sum_{p=1}^{P} a_{jp}\cdot z_{pt} \geq \sum_{i=1}^{I} r_{ji}\cdot x_{it} \qquad \forall\, j \in \{1,...,J\},\ t \in \{1,...,T\} \tag{3}$$

$$\sum_{i=1}^{I} v_{it}\cdot x_{it} + \sum_{i=1}^{I} st_{it}\cdot y_{it} \leq C_t \qquad \forall\, t \in \{1,...,T\} \tag{4}$$

$$x_{it} \leq BM_{it}\cdot y_{it} \qquad \forall\, i \in \{1,...,I\},\ t \in \{1,...,T\} \tag{5}$$

$$x,\ s,\ z \geq 0,\ y \in \{0,1\}^{I\times T} \tag{6}$$

The objective function (1) minimizes the total cost over the planning horizon. Constraints (2) ensure on-time demand satisfaction for all items (i.e., final products). Constraints (3) enforce that the total number of sub-items (i.e., sub-assemblies) obtained by cutting are sufficient to satisfy the production of final products constrained by the bill-of-material (BOM) relations. Constraints (4) enforce big-bucket capacity restrictions for final products (i.e., multiple items sharing the same resource, each item taking up both variable processing time and setup time) and constraints (5) ensure that no production occurs for item i in period t unless a set up takes place (i.e., $y_{it} = 1$), in which case the amount of production is limited by a large value $BM_{it} = \min\{\sum_{q\in\{t,\ldots,T\}} d_{iq}, (C_t - st_{it})/v_{it}\}$. Constraints (6) enforce the integrality and non-negativity requirements for various variables. Note that in the paper by [27], where this model was introduced, the z variables were defined as continuous. We keep the same definition of the problem. However, in our computational experiments in Section 6, we also consider the version of this problem imposing integrality constraints on these z variables.

The time required for proving the optimality of a given solution is often prohibitive for this model, because it provides only a weak LP relaxation, causing a large integrality gap and also poor guidance in the search for good feasible solutions in the branch-and-bound method. Therefore, we propose a new formulation that is able to provide better lower bounds. This formulation is similar to those proposed for the lot-sizing problems by [19] and [47], which provide integer solutions for the uncapacitated single-item problem, and hence we use the same terminology. This formulation is called the **S**hortest **P**ath and **C**utting-**S**tock formulation (SPCS). We introduce a new set of variables, w_{itq}, which represent the percentage of production of item i in period t used to satisfy the accumulated demand for item i from period t to period q. In SPCS, the relationships among variables w_{itq}, x_{it} and s_{it} are established as follows:

$$x_{it} = \sum_{q=t}^{T}\sum_{\ell=q}^{T} d_{iq} \cdot w_{it\ell} \qquad \forall\, i \in \{1,\ldots,I\},\ t \in \{1,\ldots,T\}$$

$$s_{it} = \sum_{q=1}^{t}\sum_{f=t+1}^{T}\sum_{\ell=f}^{T} d_{if} \cdot w_{iq\ell} \qquad \forall\, i \in \{1,\ldots,I\},\ t \in \{1,\ldots,T\}$$

Using these relationships, we can substitute x and s with w into constraints (1)-(6). The resulting SPCS formulation is given as follows:

SPCS:

$$\min \sum_{i=1}^{I}\sum_{t=1}^{T}\sum_{q=t}^{T}\sum_{\ell=q}^{T} pc_{it} \cdot d_{iq} \cdot w_{it\ell} + \sum_{i=1}^{I}\sum_{t=1}^{T}\sum_{q=t}^{T} \left\{ \sum_{\ell=t}^{q} hc_i \cdot (\ell - t) \cdot d_{i\ell} \right\} \cdot w_{itq} + \sum_{i=1}^{I}\sum_{t=1}^{T} sc_{it} \cdot y_{it}$$
$$+ \sum_{p=1}^{P}\sum_{t=1}^{T} uc \cdot z_{pt} \qquad (7)$$

Subject to:

$$\sum_{q=1}^{T} w_{i1q} = 1 \qquad \forall\, i \in \{1, ..., I\} \quad (8)$$

$$\sum_{q=1}^{t-1} w_{iq(t-1)} = \sum_{q=t}^{T} w_{itq} \qquad \forall\, i \in \{1, ..., I\},\ t \in \{2, ..., T\} \quad (9)$$

$$\sum_{p=1}^{P} a_{jp} \cdot z_{pt} \geq \sum_{i=1}^{I} \sum_{q=t}^{T} \sum_{\ell=q}^{T} r_{ji} \cdot d_{iq} \cdot w_{it\ell} \qquad \forall\, j \in \{1, ..., J\},\ t \in \{1, ..., T\} \quad (10)$$

$$\sum_{i=1}^{I} \sum_{q=t}^{T} \sum_{\ell=q}^{T} v_{it} \cdot d_{iq} \cdot w_{it\ell} + \sum_{i=1}^{I} st_{it} \cdot y_{it} \leq C_t \qquad \forall\, t \in \{1, ..., T\} \quad (11)$$

$$\sum_{q=t}^{T} w_{itq} \leq y_{it} \qquad \forall\, i \in \{1, ..., I\},\ t \in \{1, ..., T\} \quad (12)$$

$$w,\ z \geq 0,\ y \in \{0, 1\}^{I \times T} \quad (13)$$

In this formulation, constraints (8) and (9) ensure demand satisfaction for all items over the entire horizon and correspond to the flow conservation for each node in the network. Constraints (10) ensure that the total number of sub-assemblies are sufficient to satisfy the production of final products. Constraints (11) enforce capacity limits and constraints (12) are setup forcing constraints. Constraints (13) enforce the integrality and non-negativity requirements for various variables.

3. Dantzig–Wolfe decomposition and column generation

In the coupled lot-sizing and cutting-stock literature, we are only aware that [39] proposed a column generation approach for the coupled problem. However, the problem presented is different from the one studied in this paper, since it does not consider capacity constraints. Moreover, their framework uses a two-phase solution procedure and it does not solve the coupled lot-sizing and cutting-stock problem considering in conjunction capacity constraints, setup, storage and production costs. Due to these differences, the column generation approach of [39] is not directly applicable for the LS-CS problem studied in this paper, and this is our main motivation to propose two new Dantzig–Wolfe decomposition and column generation approaches, i.e., per-item and per-period decompositions.

For the per-item decomposition, the problem is decomposed into single-item uncapacitated lot-sizing subproblems containing the demand (2), setup forcing (5) and integrality (6) constraints for the specific item, while the BOM (3) and capacity (4) constraints are the linking constraints in the master problem. For the per-period decomposition, the problem is decomposed into single-period subproblems containing the BOM (3), capacity (4), setup forcing (5) and integrality (6) constraints for the specific period, while the demand constraints (2) are the linking constraints in the master problem. An overview of lower bounds yielded by these decompositions is given in Table 1, where the notation indicates which formulation and decomposition have been considered, e.g., $\hat{\mathrm{D}}_{\mathrm{LSCS1}}$ means the lower bound obtained by the per-item decomposition of the LSCS formulation. We also define the notation $\hat{\mathrm{R}}$ to indicate LP relaxation.

Table 1: Summary of lower bounds yielded by the Dantzig–Wolfe decompositions

	No decomposition	Decomposition 1 (per-item)	Decomposition 2 (per-period)
The LSCS formulation	$\hat{\mathrm{R}}_{\mathrm{LSCS}}$	$\hat{\mathrm{D}}_{\mathrm{LSCS1}}$	$\hat{\mathrm{D}}_{\mathrm{LSCS2}}$
The SPCS formulation	$\hat{\mathrm{R}}_{\mathrm{SPCS}}$	$\hat{\mathrm{D}}_{\mathrm{SPCS1}}$	$\hat{\mathrm{D}}_{\mathrm{SPCS2}}$

We next describe the per-period Dantzig–Wolfe decomposition of SPCS and present the other decompositions in the Online Supplement. We define K^t as the set of all possible setup schedules for period t, $K^t = \{(y_{1t}, \ldots, y_{It}) | y_{it} \in \{0,1\}, \forall\, i \in \{1,...,I\}\}$. For a given possible setup schedule $k \in K^t$, we define the following parameters: y^k_{it} is the setup indicator for item i in period t; w^k_{itq} is the percentage of production of item i in period t used to satisfy the accumulated demand for item i from period t to period q; and z^k_{pt} is the number of plates that are cut with pattern p in period t. In addition, π_{tk} is the new variable and is defined as the fraction of production made using setup schedule k in period t; γ_{it} is the dual variable related with the demand-satisfaction constraints; and δ_t is the dual variable for the convexity constraints. The SPCS formulation can be separated into a master problem (MP-D$_{SPCS2}$) and T pricing problems (SP-D$_{SPCS2}$) as follows:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{I}\sum_{t=1}^{T}\sum_{q=t}^{T}\sum_{\ell=q}^{T}\sum_{k\in K^t} pc_{it}\cdot d_{iq}\cdot w^k_{it\ell}\cdot \pi_{tk} + \sum_{i=1}^{I}\sum_{t=1}^{T}\sum_{q=t}^{T}\sum_{k\in K^t}\left\{\sum_{\ell=t}^{q} hc_{it}\cdot(\ell-t)\cdot d_{i\ell}\right\}\cdot w^k_{itq}\cdot\pi_{tk} \\
& \quad + \sum_{i=1}^{I}\sum_{t=1}^{T}\sum_{k\in K^t} sc_{it}\cdot y^k_{it}\cdot \pi_{tk} + \sum_{p=1}^{P}\sum_{t=1}^{T}\sum_{k\in K^t} uc\cdot z^k_{pt}\cdot\pi_{tk} \\
\text{s.t.} \quad & \sum_{q=1}^{T}\sum_{k\in K^1} w^k_{i1q}\cdot\pi_{1k} = 1,\ \forall\, i\in\{1,...,I\} \\
& \sum_{q=1}^{t-1}\sum_{k\in K^q} w^k_{iq(t-1)}\cdot\pi_{qk} = \sum_{q=t}^{T}\sum_{k\in K^t} w^k_{itq}\cdot\pi_{tk},\ \forall\, i\in\{1,...,I\},\ t\in\{2,...,T\} \\
& \sum_{k\in K^t}\pi_{tk} = 1,\ \forall\, t\in\{1,...,T\} \qquad\qquad (\text{MP-D}_{SPCS2}) \\
& \pi_{tk}\geq 0,\ \forall\, t\in\{1,...,T\},\ k\in K^t
\end{aligned}$$

$$
\begin{aligned}
\min \quad & \zeta_t = \sum_{i=1}^{I}\sum_{q=t}^{T}\sum_{\ell=q}^{T} pc_{it} \cdot d_{iq} \cdot w_{it\ell} + \sum_{i=1}^{I}\sum_{q=t}^{T}\left\{\sum_{\ell=t}^{q} hc_{it} \cdot (\ell - t) \cdot d_{i\ell}\right\} \cdot w_{itq} + \sum_{i=1}^{I} sc_{it} \cdot y_{it} \\
& \qquad + \sum_{p=1}^{P} uc \cdot z_{pt} - \sum_{i=1}^{I}\sum_{q=t}^{T} \gamma_{it} \cdot w_{itq} + \sum_{i=1}^{I}\sum_{q=t}^{T-1} \gamma_{i(q+1)} \cdot w_{itq} - \delta_t \\
\text{s.t.} \quad & \sum_{i=1}^{I}\sum_{q=t}^{T}\sum_{\ell=q}^{T} v_{it} \cdot d_{iq} \cdot w_{it\ell} + \sum_{i=1}^{I} st_{it} \cdot y_{it} \leq C_t \\
& \sum_{p=1}^{P} a_{jp} \cdot z_{pt} \geq \sum_{i=1}^{I}\sum_{q=t}^{T}\sum_{\ell=q}^{T} r_{ji} \cdot d_{iq} \cdot w_{it\ell}, \ \forall\, j \in \{1, ..., J\} \\
& \sum_{q=t}^{T} w_{itq} \leq y_{it}, \ \forall\, i \in \{1, ..., I\} \\
& z_{pt}, w_{itq} \geq 0, \ y_{it} \in \{0, 1\}, \ \forall\, i \in \{1, ..., I\}, \ q \in \{t, ..., T\}, \ p \in \{1, ..., P\}
\end{aligned}
\tag{SP-D$_{SPCS2}$}
$$

To achieve lower bounds, we suggest the use of column generation, which stems from the idea that when a problem has significantly more variables (i.e., columns) than constraints (i.e., rows), since the number of basic variables will be not more than the number of its constraints and hence the majority of variables will end up being non-basic, one can identify these useful columns of the basic solution, rather than including all columns in the problem. Column generation with the Dantzig–Wolfe decomposition has been widely used for problems with complicating linking constraints and also proven to be useful for the lot-sizing problem (see, e.g., [33] and [15]).

This approach consists of a *restricted master problem*, which represents the linking constraints of the original problem and additional convexity constraints and contains a limited number of variables that are selected so far. When the restricted master problem is solved, it provides dual multiplier values to a *pricing problem*, which is a problem that decides whether there are any other useful columns remaining or not. When the pricing problem returns a column, this new variable is added to the master problem and that is resolved and the procedure terminates when no columns price out, i.e., when $\sum_{t\in\{1,\ldots,T\}} \min(\zeta_t, 0) = 0$. Note that a valid lower bound on the objective value of the original problem is available throughout column generation ([15] and [21]). If $\hat{\mathrm{D}}^{\epsilon}_{RMP}$ is the optimal objective value of the restricted master problem at iteration ϵ, then a valid lower bound is $\hat{\mathrm{D}}^{\epsilon}_{LB} = \hat{\mathrm{D}}^{\epsilon}_{RMP} + \sum_{t\in\{1,\ldots,T\}} \min(\zeta_t, 0)$.

As noted in the literature, the primal solutions of the restricted master problem are usually degenerate ([18] and [36]). When the dual restricted master problem has multiple optimal solutions and therefore the dual optimal solution at hand might not be an accurate representation of the optimal dual space, this may result in the pricing problems pricing out some bad-quality columns, which are not used in the optimal solution of the subsequent restricted master problem(s). In this case, column generation takes a degenerate step. This phenomenon has a severe impact on the algorithmic performance, and it is usually magnified as the final optimal solution is approached, thereby called the tailing-off effect ([17] and [21]).

We notice that the per-period Dantzig–Wolfe decomposition of SPCS is degenerate and influenced by the tailing-off effect. To deal with this issue, we restrict the dual space of the restricted master program by introducing artificial variables on the primal space, i.e., the stabilization technique described in [18]. This method reduces the number of degenerate iterations via reducing the feasible dual space. In addition, during early iterations, we employ a hybrid column generation and Lagrangian relaxation scheme to enhance the algorithmic performance, similar to those described in [16], [15] and [21]. We present the Lagrangian relaxation problem in the Online Supplement. In the Lagrangian problem, the demand satisfaction constraints (8) and (9) are dualized into the objective function (7) with non-negative dual multipliers η_i and μ_{it}.

There is a strong relationship between Dantzig–Wolfe decomposition and Lagrangian relaxation ([23]). Because both methods have the same subproblem, we can use both to generate columns. In addition, the optimal dual variables (γ_{it}) for the linking constraints in the Dantzig–Wolfe master correspond to the optimal multipliers (η_i and μ_{it}) for the complicating constraint in the Lagrangian objective function. Therefore, we can pass the dual values of the restricted master problem to the Lagrangian relaxation problem and generate a new set of dual values via subgradient optimization given in the Online Supplement. This updating process is deemed to lead to better quality dual prices, and it has the additional benefit that we can add several new columns to the restricted master problem through solving the Lagrangian problem without solving the restricted master problem. We call this procedure whenever column generation takes a degenerate step, i.e., when the optimal master objective does not improve in two consecutive iterations.

We now discuss some theoretical relationships between the bounds.

Theorem 3.1 $\hat{R}_{LSCS} \leq \hat{D}_{LSCS1} = \hat{D}_{SPCS1} = \hat{R}_{SPCS}$.

Proof. The first inequality is established by the fact that applying a Dantzig–Wolfe decomposition to a minimization problem cannot reduce the lower bound yielded by the LP relaxation ($\hat{\mathrm{R}}_{\mathrm{LSCS}}$). The next equality follows from the fact that both per-item Dantzig–Wolfe decompositions describe the convex hull of the single-item uncapacitated lot-sizing problems, see, e.g., [3]. If we let ($\bar{x}_{it}$, $\bar{s}_{it}$, $\bar{y}_{it}$, $\bar{z}_{pt}$) be a feasible solution of the decomposition model of the LSCS formulation, it means that a) the solution point is contained in the convex hull defined by constraints (2)-(3)

and (5)-(6) and b) it satisfies constraints (4). We can prove that this solution must correspond to a feasible solution point ($\bar{w}_{itq}$, $\bar{y}_{it}$, $\bar{z}_{pt}$) contained in the convex hull defined by constraints (8)-(10) and (12)-(13) and satisfy constraints (11) for the per-item decomposition of the SPCS formulation. The above logic follows reversely. Finally, the last equality is trivial due to the fact that the convex hulls of these relaxations have all integral extreme points. □

In other words, since the subproblem for the per-item decomposition of the SPCS formulation is the shortest path reformulation of the single-item uncapacitated lot-sizing problem (containing constraints (8)-(10) and (12)-(13)), these subproblems have the integrality property. Consequently, the decomposition bound $\hat{\mathrm{D}}_{\mathrm{SPCS1}}$ is equal to the original LP relaxation bound $\hat{\mathrm{R}}_{\mathrm{SPCS}}$.

Theorem 3.2 $\hat{D}_{SPCS1} \leq \hat{D}_{SPCS2}$.

Proof. Using the fact that the SPCS formulation contains constraints (8)-(9) and (11)-(13) that are the same as the constraints of the shortest path reformulation of the classical multi-item capacitated lot-sizing problem and the solution space of these constraints is not affected by constraints (10); and using the fact that the shortest path reformulation of the classical multi-item capacitated lot-sizing problem defines the convex hull of the single-item uncapacitated lot-sizing problem (e.g., [40]), we have that the SPCS formulation also defines the convex hull of the single-item uncapacitated lot-sizing problem. Applying the per-item decomposition to the SPCS reformulation therefore cannot improve the lower bound yielded by the LP relaxation. However, applying the per-period decomposition to the shortest path formulation can lead to at least equivalent or better lower bounds, as the pricing problems do not have the integrality property. The numerical examples showing that for some instances this inequality is satisfied strictly as inequality are provided in the Online Supplement, where we also show that the per-period decomposition improves lower bounds for almost all tested instances. □

Theorem 3.3 $\hat{D}_{LSCS2} \leq \hat{D}_{SPCS2}$

Proof. See the Online Supplement. □

4. The progressive selection method for the LS-CS problem

For the simplicity of the description, we use the following binary mixed integer programming (BMIP) formulation to represent the mathematical formulation of the LS-CS problem.

$$\begin{aligned} \min \quad & a \cdot x + b \cdot y \\ \text{Subject to:} \quad & C \cdot x + D \cdot y \geq e \\ & x \in \mathbb{R}^n, \;\; y \in \{0,1\}^m \end{aligned} \tag{$\mathcal{P}$}$$

For problem ($\mathcal{P}$), x is an n-dimensional vector of real variables, y is an m-dimensional vector of binary variables (each element represented as y_h, $\forall\, h \in \{1, \ldots, m\}$) and a, b, C, D and e are given parameter vectors/matrices. To solve problem ($\mathcal{P}$), the progressive selection (PS) method initially generates a population of solutions, which are not required to be feasible, to problem ($\mathcal{P}$) using an upper bounding technique such as heuristics within a limited amount of computational time, and generates a relaxation solution of problem ($\mathcal{P}$) using a lower bounding technique such as LP relaxation. To facilitate the explanation, we use the following definitions:

$\bar{y}$	The solution point of y that leads to an upper bound of problem ($\mathcal{P}$). We let the upper bound be ∞ if $\bar{y}$ is an infeasible solution point.
$\underline{y}$	The relaxation solution point of y that leads to a lower bound of problem ($\mathcal{P}$).
$\mathcal{G}_y$	The population of the corresponding solution points $\bar{y}$ in the population of solutions.
G	The size of population $\mathcal{G}_y$.
g	The index of $\bar{y}$ in population $\mathcal{G}_y$, where $\bar{y}^g = [\bar{y}^g_1, \bar{y}^g_2, \ldots, \bar{y}^g_m]^T$, $\forall\, g \in \{1, ..., G\}$. As such, $\mathcal{G}_y = \{\bar{y}^1, \bar{y}^2, \ldots, \bar{y}^g, \ldots, \bar{y}^G\}$. We can write $\mathcal{G}_y$ in a matrix form as follows:

$$\mathcal{G}_y = \begin{pmatrix} \bar{y}^1_1 & \bar{y}^2_1 & \cdots & \bar{y}^g_1 & \cdots & \bar{y}^G_1 \\ \bar{y}^1_2 & \bar{y}^2_2 & \cdots & \bar{y}^g_2 & \cdots & \bar{y}^G_2 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \bar{y}^1_m & \bar{y}^2_m & \cdots & \bar{y}^g_m & \cdots & \bar{y}^G_m \end{pmatrix}$$

$\hat{g}$	The index of the solution point of $\mathcal{G}_y$ that leads to the smallest upper bound of problem ($\mathcal{P}$).
$\bar{y}^{\hat{g}}$	The solution point of $\mathcal{G}_y$ that leads to the smallest upper bound of problem ($\mathcal{P}$). For simplicity, we also define $\bar{y}^{\hat{g}}$ as $\hat{y}$.

Given $\mathcal{G}_y$ and $\underline{y}$, the motivation behind the PS method is to find an efficient way to fix a subset of binary variables y_h, $h \in \{1, \ldots, m\}$, that leads to a restricted version of problem ($\mathcal{P}$) and hence is easier to solve. To define the restricted problem (called *subproblem*), we introduce three non-overlapping subsets of the index set $\{1, \ldots, m\}$: FV, SV and MV. FV defines an index set of binary variables that have been fixed based on results of the previous iteration(s); SV indicates an index set of binary variables that are individually selected and fixed to the corresponding values of

$\hat{y}$ for a specific subproblem; and MV defines the remaining index set of binary variables. With the above definitions, the subproblem ($\mathcal{P}^S$) can be defined as follows:

$$\begin{aligned} \min \quad & a \cdot x + b \cdot y \\ \text{s.t.} \quad & C \cdot x + D \cdot y \geq e \\ & y_h = \hat{y}_h, \ \ h \in FV \cup SV \\ & y_h \in \{0,1\}, \ \ h \in MV \\ & x \in \mathbb{R}^n \end{aligned} \tag{$\mathcal{P}^S$}$$

The PS method tries to find a solution for problem ($\mathcal{P}$) by iteratively solving subproblem ($\mathcal{P}^S$) using different subsets FV, SV and MV. At the beginning of each iteration, the subproblem is determined only by subsets FV and MV, since SV is empty. However, a number of different selections for SV creates a number of different subproblems. We name these subproblems *potential subproblems*, which are solved by using some upper bounding technique. At the current iteration, the subproblem that leads to the smallest objective value is considered the most interesting potential subproblem, where a subset of indices of its associated SV is inserted to FV. When the method moves to the next iteration, SV is reset to be empty. With the evolution of the method, the size of FV grows and hence the size of subproblems becomes smaller. Since fixing variables in FV might cause the method to converge locally, the method also generates a number of *random subproblems* in order to have a global perspective.

We next describe all six steps of the PS method in detail, from initialization to stop-criterion check. In the description, emphasis will be given on how subproblems are created, i.e., how the subsets FV, SV and MV are selected and evolved in the method. Before the description, we define some necessary notation as follows:

ν	The current number of iterations.
N_ν^1	Number of potential subproblems created at the ν-th iteration.
N_ν^2	Number of random subproblems created at the ν-th iteration.
ψ_ν	Index for subproblems at the ν-th iteration, $\psi_\nu \in \{1, ..., N_\nu^1 + N_\nu^2\}$.
SV_{ψ_ν}	Subset SV for the ψ_ν-th subproblem, $\psi_\nu \in \{1, ..., N_\nu^1 + N_\nu^2\}$.
MV_{ψ_ν}	Subset MV for the ψ_ν-th subproblem, $\psi_\nu \in \{1, ..., N_\nu^1 + N_\nu^2\}$.
n_{ψ_ν}	Number of binary variables selected to be fixed in the ψ_ν-th subproblem, $n_{\psi_\nu} = \lvert SV_{\psi_\nu} \rvert$, $\psi_\nu \in \{1, ..., N_\nu^1 + N_\nu^2\}$.
$\mathcal{P}_{\psi_\nu}^S$	The ψ_ν-th subproblem, $\psi_\nu \in \{1, ..., N_\nu^1 + N_\nu^2\}$.
UB	The current best upper bound of problem ($\mathcal{P}$).
$\hat{y}_\nu$	$\hat{y}$ at the ν-th iteration.
T_s	Computational time limit for solving subproblem ($\mathcal{P}^S$).
T_{lim}	Computational time limit for the whole method.

- **Step I: Initialization.** Set $\nu = 1$, $UB = \infty$, $FV = \emptyset$, $SV = \emptyset$ and $MV = \{1, \ldots, m\}$. Solve problem $(\mathcal{P})$ using an upper bounding technique (e.g., using different parameter settings) to obtain $\mathcal{G}_y$ with G solutions and $\hat{y}$, and solve problem $(\mathcal{P})$ using the LP relaxation of the SPCS formulation to obtain $\underline{y}$, which will be used in Step II for the progressive selection procedure. Go to Step II.

- **Step II: Progressive selection.** The main idea of progressive selection is to use a domain-knowledge-guided selection procedure to determine index subsets SV and MV that are used for building the subproblems. The details of the progressive selection procedure are described in Section 5.1. Go to Step III.

- **Step III: Subproblem creation.** With the selection of SV_{ψ_ν} and MV_{ψ_ν}, the ψ_ν-th potential subproblem ($\mathcal{P}^S_{\psi_\nu}$, $\forall\ \psi_\nu \in \{1, ..., N^1_\nu\}$) can be easily created by fixing the variables in SV_{ψ_ν} to their corresponding values of $\hat{y}$. Besides potential subproblems, N^2_ν random subproblems are also created using a random sampling scheme, where the size of the set of fixed variables is defined as $|FV| + n_{\psi_\nu}$. Different from the strategy of fixing variables to $\hat{y}$ for potential subproblems, the binary values, which the selected variables are fixed to, are randomly sampled with a constant probability of being 0 or 1. The constant probability can be varied dependent on the problem's properties to enhance the solution quality of random subproblems, e.g., according to the proportion observed in the best solution. Go to Step IV.

- **Step IV: Subproblem evaluation.** A total number of $N^1_\nu + N^2_\nu$ subproblems are created in Step III. These subproblems are solved by the upper bounding technique to derive an upper bound, potentially improving the current best upper bound UB. To provide better guidance for subsequent iterations, UB and $\hat{y}$ are therefore updated whenever a better objective value is found; population $\mathcal{G}_y$ is also updated with the solution points of y of the G current best subproblems. Go to Step V.

- **Step V: Subset update.** The subproblem with the smallest upper bound is selected as the best subproblem at the current iteration. If the best subproblem is one of the potential subproblems, we select for the best subproblem the variable with index h from the set SV that has the highest value for ω_h (defined in Section 5.1), i.e., $h = \arg\max_{h' \in SV}(\omega_{h'})$. Next we insert the index h into FV. In case there is more than one variable in SV having the maximum value of ω_h, we randomly select one of them to insert into FV. We set $MV = MV - FV$ and $SV = \emptyset$. Otherwise, we update $FV = \emptyset$, $MV = \{1, 2, \ldots, m\}$ and $SV = \emptyset$. Go to Step VI.

- **Step VI: Stop-criterion check.** We define two stop-criteria. The method stops either when the computational time limit (T_{lim}) is met or when $FV = \{1, 2, \ldots, m\}$. Otherwise, the method sets $\nu = \nu + 1$ and goes to Step II. $\square$

The PS method has a potential to be applied to other BMIP problems, in addition to the LS-CS problem investigated in this paper. However, we acknowledge that the effectiveness of the PS method for general BMIP problems should be verified by an extensive computational analysis, comparing with effective algorithms from the literature. However, such an extensive computational comparison is outside of the scope of this paper, since we focus on the application that combines lot-sizing and cutting-stock. Furthermore, when this method is applied to a specific BMIP problem, some extra effort in selecting upper bounding and lower bounding techniques is needed to enhance the method performance.

5. Further discussion of the PS method for the LS-CS problem

In this section, we will discuss the progressive selection procedure in detail, and present the upper bounding technique we select for the PS method.

5.1 Progressive selection

The details of progressive selection are given in Algorithm 1 that determines index subsets SV and MV. To describe the selection procedure, we define, for each $h \in MV$, φ_h as the distance between $\hat{y}_h$ and $\underline{y}_h$ (that is, $\varphi_h = |\hat{y}_h - \underline{y}_h|$), and we define ω_h as the number of $\bar{y}_h^g$ ($\forall\ g \in \{1, ..., G\} \backslash \hat{g}$) that take the same value as $\hat{y}_h$. We have $\omega_h = \sum\limits_{g \in \{1,...,G\} \backslash \hat{g}} \sigma_h^g$, $\forall\ h \in MV$ where

$$\sigma_h^g = \begin{cases} = 0, & \text{if } \bar{y}_h^g \neq \hat{y}_h, \\ = 1, & \text{if } \bar{y}_h^g = \hat{y}_h, \end{cases} \quad \forall\ g \in \{1, ..., G\} \backslash \hat{g},\ h \in MV.$$

An example showing how to calculate φ_h and ω_h is given in Table 2. With these definitions, in Algorithm 1, the probability ($Prob_h$) of selecting variable y_h to be fixed in a potential subproblem is defined as follows:

$$Prob_h = \frac{\rho_1^{\rho_2 \cdot (\omega_h + 1 - \varphi_h)}}{\sum_{h' \in MV} \rho_1^{\rho_2 \cdot (\omega_{h'} + 1 - \varphi_{h'})}} \qquad \forall\ h \in MV. \tag{14}$$

Here, parameters ρ_1 and ρ_2 ($\rho_1 \geq 1$ and $\rho_2 \geq 1$) are used to adjust sampling weights. A larger value of ρ_1 and/or ρ_2 leads to a higher likelihood of selecting a variable that has high similarities with solution values of population $\mathcal{G}_y$.

```
for $\psi_\nu \leftarrow 1$ to $N_\nu^1$ do
    $SV_{\psi_\nu} \leftarrow \emptyset$ ;
    $MV_{\psi_\nu} \leftarrow MV$ ;
    for $h \leftarrow 1$ to $n_{\psi_\nu}$ do
        Select an index from $MV_{\psi_\nu}$ using the likelihood function defined by (14) ;
        Remove the index out of $MV_{\psi_\nu}$ ;
        Insert the index to $SV_{\psi_\nu}$ ;
    end
end
Return $SV_{\psi_\nu}$, $MV_{\psi_\nu}$, $\forall\ \psi_\nu \in \{1, ..., N_\nu^1\}$
```

Algorithm 1: Progressive selection

One can select the index of a binary variable directly using a uniform distribution, i.e., all $h \in MV$ have an equal probability to be picked. However, defining individual probabilities based on similarities among solution points in population $\mathcal{G}_y$ leads to better subproblems. Our computational experience reveals that the more similar other values $\bar{y}_h^g$ ($\forall\ g \in \{1, ..., G\} \backslash \hat{g}$, $h \in \{1, 2, ..., m\}$) with $\hat{y}_h$, the higher the probability that $\hat{y}_h$ is the same as the corresponding optimal value of y_h of problem ($\mathcal{P}$). The analysis of its statistical significance will be given in Section 6. Meanwhile, our computational tests also show that the smaller the 1-norm distance between the variable's values $\hat{y}_h$ and $\underline{y}_h$, the higher the probability that the variable's value $\hat{y}_h$ is the same as its corresponding optimal value of y_h of problem ($\mathcal{P}$). The analysis of its statistical significance on the two PS applications will be also discussed in Section 6. We therefore define the selection probability function (14) for selecting SV_{ψ_ν} that favors the variables that have $\hat{y}_h$ most similar with other $\bar{y}_h^g$ in population $\mathcal{G}_y$ and the variables that have smaller 1-norm distances between variable's values $\hat{y}_h$ and $\underline{y}_h$. To illustrate these concepts with a numerical example, we present Table 2 where the size of population $\mathcal{G}_y$ is 3 and $\hat{g}$ is 2. For the sake of simplicity of probability calculations, let the set MV consist of only the variables presented in the table. If three variables were to be chosen, the variables y_1, y_5 and y_{h+1} would have a higher likelihood to be the choices. We also note that if the initialization step turned only one feasible solution, i.e., $G = 1$, then the selection of the variables is only based on the 1-norm distance.

5.2 The upper bounding technique

For the upper bounding technique of the PS approach, we have considered a number of candidate methods such as various heuristics. From our experience (in particular in the area of lot-sizing) as well as from some preliminary testing on the coupled problem, we have observed that relax-and-fix works well as a general upper bounding technique. Relax-and-fix has been applied successfully to

Table 2: A numerical example for index selection

Variables	y_1	y_2	y_3	y_4	y_5	$\dots$	y_h	y_{h+1}	y_{h+2}	$\dots$
Feasible solution $\bar{y}^1$	1	0	1	0	1	$\dots$	1	0	1	$\dots$
Feasible solution $\bar{y}^2$ (also $\hat{y}$)	1	1	0	1	1	$\dots$	0	0	0	$\dots$
Feasible solution $\bar{y}^3$	1	1	0	0	1	$\dots$	1	0	1	$\dots$
Relaxation solution $\underline{y}$	1	0.7	0.3	0.4	0.9	$\dots$	0.6	0.1	0.5	$\dots$
ω_h	2	1	1	0	2	$\dots$	0	2	0	$\dots$
φ_h	0	0.3	0.3	0.6	0.1	$\dots$	0.6	0.1	0.5	$\dots$
$Prob_h$ ($\rho_1 = 2 = \rho_2$)	0.317	0.052	0.052	0.009	0.276	$\dots$	0.009	0.276	0.01	$\dots$

different classes of lot-sizing problems, see, e.g., [2, 43, 48]. The relax-and-fix algorithm implemented in this paper partitions the set of all periods into three subsets: The first subset is a period window that contains α periods where all setup decision variables are kept as binary variables; the second subset contains all periods preceding the period window where all setup decision variables have been fixed based on solution values of the previous iteration(s); and the third subset consists of all periods following the period window where all setup decision variables are relaxed as continuous variables. At the first iteration of the algorithm, a MIP solver is applied to solve this smaller restricted problem with a limited amount of computational time T_{rf} and the resulting solution value is used to fix the binary variables of the first β periods of the period window ($\beta \leq \alpha$). The period window rolls forward to the next few periods containing period $\beta + 1$ to period $\beta + \alpha$ at the next iteration. The same step is iteratively performed until the period window rolls to the end and a binary solution value has been achieved for all binary setup decision variables. In the PS approach, parameters α and β are set to different values to generate a variety of solutions of y in the initial population $\mathcal{G}_y$.

6. Statistical analyses of the PS method for the $LS-CS$ problem

In Section 5, we gave three observations without verification:

- Observation 1: Potential subproblems yield better upper bounds than random subproblems.
- Observation 2: The more similar other values $\bar{y}_h^g$ ($\forall\ g \in \{1, ..., G\} \backslash \hat{g},\ h \in \{1, ..., m\}$) with $\hat{y}_h$ ($\forall\ h \in \{1, ..., m\}$), the higher the probability that $\hat{y}_h$ is the same as the corresponding optimal value of problem ($\mathcal{P}$).
- Observation 3: The smaller the distance between $\hat{y}_h$ and $\underline{y}_h$ ($\forall\ h \in \{1, ..., m\}$), the higher the probability that $\hat{y}_h$ is the same as the corresponding optimal value of problem ($\mathcal{P}$).

In this section, we will perform statistical analyses to show that the three observations are statistically significant for the application of the PS method to the LS-CS problem. We generated two groups of test instances (SetA and SetB) using different parameter settings in order to perform statistical analyses. The settings of these test instances are described in Section 7.

To test Observation 1, we considered all generated potential and random subproblems for each test instance and performed statistical analyses using different statistical techniques, such as boxplot, Welch's test and finite mixture model. The computational results showed that all potential subproblems are feasible but only about 52%, on average, random subproblems are feasible.

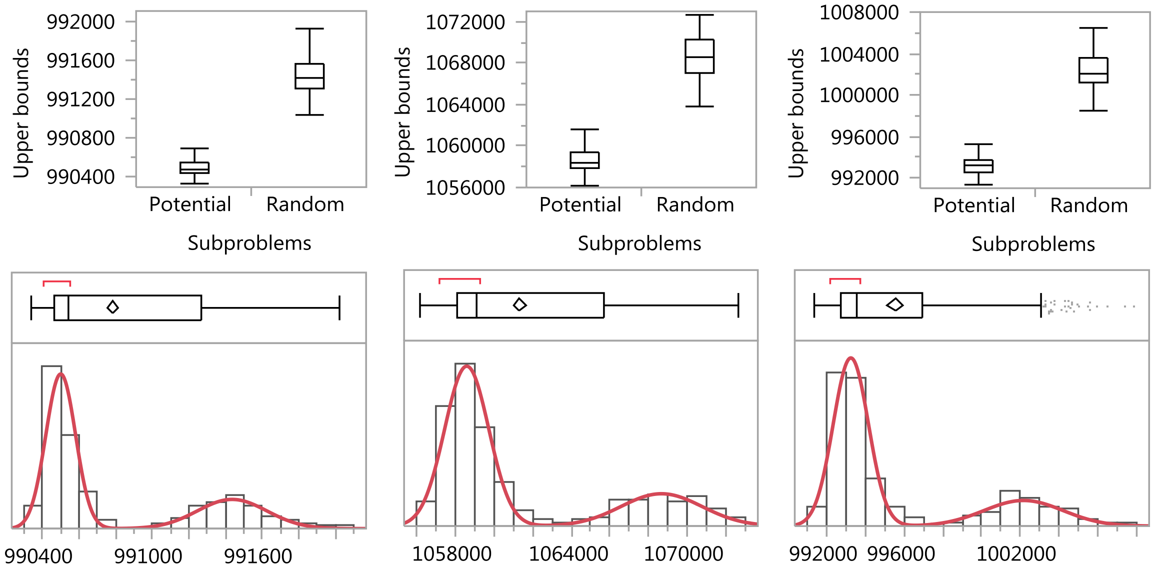

Figure 1: Analytical results of three randomly chosen instances to test Observation 1.

Figure 1 shows analytical results of three instances randomly chosen from all test instances to test Observation 1, where the results of the boxplots and the fitting curves of finite mixture models for three representative examples are presented. The three top figures give boxplots showing that potential subproblems have significantly better upper bounds when compared with random subproblems. The bottom three figures show the histograms of upper bounds of both potential and random subproblems. The grey curves were derived by the finite mixture model method for which the distribution of the upper bounds is assumed to be a mixture of two normal distributions. From the results, we have that the distribution of the upper bounds can be well fitted as a mixture of two normal distributions with significantly different means; and that the distinguishing accuracy is about 95% on average for these three representative examples if the upper bounds, generated by both potential and random subproblems, are separated into two clusters. Moreover, Welch's test showed that upper bounds derived from potential subproblems are better with statistical significance compared to those derived from random subproblems with a p-value less than 0.0001.

Finally, the test results for Observations 2 and 3 are shown in Figures 2 and 3, respectively. In Figure 2, boxplots and linear regression results of three groups of test instances are presented to test Observation 2, where the population group is built using the result at the end of the algorithm. To ensure the optimal solution be found for all test instances, we generated the instances with the same parameter settings described in Section 7 but with smaller sizes. In addition, we generated 4 repeats for each parameter setting such that there are a total number of 320 test instances within each group. The top two sub-figures show results for the group with 5 final products, 5 types of sub-assemblies and 16 periods; the middle two sub-figures are for the one with 10 final products, 5 types of sub-assemblies and 16 periods; and the bottom two sub-figures are for the one with 10 final products, 10 types of sub-assemblies and 16 periods. In the figure, MG represents, for each setup decision variable y_{it} ($\forall\ i \in \{1, ..., I\},\ t \in \{1, ..., T\}$), the number of solution values $\bar{y}_{it}^{g}$ ($\forall\ g \in \{1, ..., 6\} \backslash \hat{g}$) that are the same as $\hat{y}_{it}$. The range of MG is between 0 and 5, as the population size was set to 6 in the experimental tests. MR denotes the probability of the value $\hat{y}_{it}$ to be the same as the corresponding optimal solution value for each test instance, which is calculated for each MG and distance bin, e.g., if there are 100 variables and 80 of them coincide with the optimal solution, then $MR = 0.8$. MR-AVG indicates the MR on average for all 320 test instances within each group. From the experimental results, we can see that both statistical methods show a strong statistical relationship between MR and MG. That is, the more similar other values $\bar{y}_{it}^{g}$ ($\forall\ g \in \{1, ..., 6\} \backslash \hat{g},\ i \in \{1, ..., I\},\ t \in \{1, ..., T\}$) with $\hat{y}_{it}$ ($\forall\ i \in \{1, ..., I\},\ t \in \{1, ..., T\}$), the higher the probability that $\hat{y}_{it}$ is the same as the corresponding optimal value of the LS-CS problem.

Figure 3, similar to the previous figure, shows boxplots of the same three groups of test instances to test Observation 3. MR has a similar interpretation as in the previous analysis. For each setup decision variable y_{it} ($\forall\ i \in \{1, ..., I\},\ t \in \{1, ..., T\}$), the distance is calculated as $|\hat{y}_{it} - \underline{y}_{it}|$. The distance value is separated into two different bins in order to perform statistical comparisons shown by the boxplots. From the experimental results, we can see that this statistical analysis shows a strong statistical relationship between MR and distance. That is, the smaller the distance between $\hat{y}_{it}$ and $\underline{y}_{it}$ ($\forall\ i \in \{1, ..., I\},\ t \in \{1, ..., T\}$), the higher the probability that $\hat{y}_{it}$ is the same as the corresponding optimal value of the LS-CS problem.

To further analyze the Observations 2 and 3, we also performed Welch's test, as presented in Table 3. We note that since the variances across groups are not equal, the common ANOVA F test is not valid in this setting. In Table 3, F Ratio indicates the F test statistic for the equal variance test; DFDen records the degrees of freedom in the denominator of the test; and Prob >

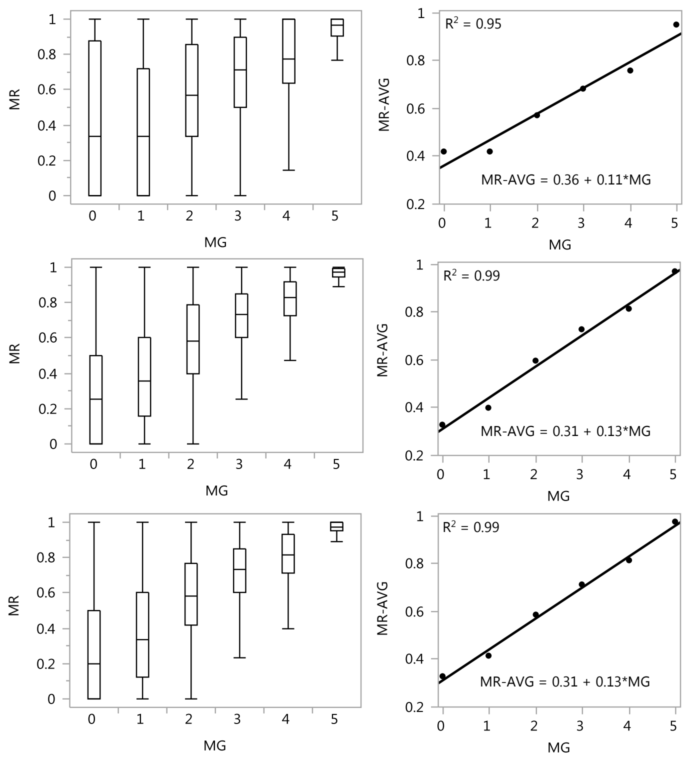

Figure 2: Boxplots and linear regression results of three groups of test instances to test Observation 2.

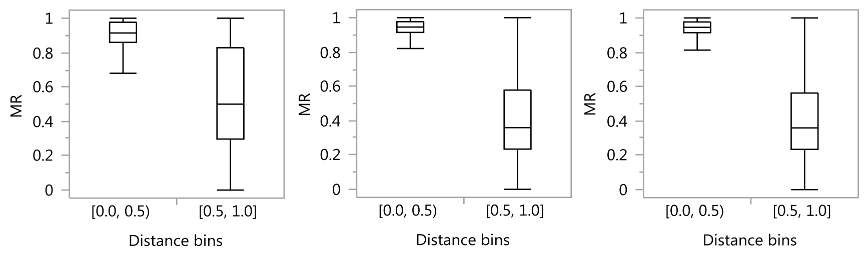

Figure 3: Boxplots of the same three groups of test instances to test Observation 3

Table 3: Welch's test for the effects of MG and distance bins on MR

		16-5-5			16-10-5			16-10-10		
	(Level) - (Level)	F Ratio	DFDen	Prob > F	F Ratio	DFDen	Prob > F	F Ratio	DFDen	Prob > F
MG	(0) - (1)	0.25	289.76	0.6169	6.92	488.28	0.0088	6.36	491.85	<.0120
	(1) - (2)	24.69	443.55	<.0001	64.24	562.48	<.0001	62.79	553.42	<.0001
	(2) - (3)	16.01	513.85	<.0001	32.75	560.69	<.0001	36.05	554.82	<.0001
	(3) - (4)	13.30	507.32	0.0003	26.96	543.01	<.0001	23.37	569.2	<.0001
	(4) - (5)	158.79	297.51	<.0001	230.81	290.63	<.0001	227.57	297.23	<.0001
	(Level) - (Level)	F Ratio	DFDen	Prob > F	F Ratio	DFDen	Prob > F	F Ratio	DFDen	Prob > F
Distance bins	[0.0, 0.5) - [0.5, 1.0]	424.11	358.58	<.0001	1075.74	333.03	<.0001	1073.31	330.72	<.0001

F means the probability of obtaining, by chance alone, an F value larger than the one calculated if in reality the means are equal across all levels. We considered three groups of problems, where the name of each group follows the syntax "number of periods-number of final products-number of sub-assemblies" as indicated on the top row of the table. Observed significance probabilities of 0.05 or less are considered evidence of unequal means across the levels. Obviously, the Welch's test clearly shows that both MG and distance bins have significant effects on MR as the p-value is much smaller than 0.05 for all tests.

7. Computational tests

We generated six groups of test instances (SetA, SetB, SetC, SetD, SetE, and SetF) using different parameter settings in order to test the relative performance of the PS method. The test instances are available at DOI: 10.15129/5e486422-31b0-4cbf-b1c1-584602dd7e3d, http://dx.doi.org/10.15129/5e486422-31b0-4cbf-b1c1-584602dd7e3d. SetA has 20 final products, 20 types of sub-assemblies and 16 periods, whereas SetB has 25 final products, 15 types of sub-assemblies and 16 periods. Two levels were applied to six parameters (demand, production cost, plate cost, capacity usage, setup time, and the ratio (sc_{it}/hc_i) between setup and inventory-holding costs (TBO). The experimental design is full factorial such that there are 64 test instances in each group. The detailed design of the parameter settings for SetA and SetB is given as follows:

$w \times l$	Plate width $\times$ length is set to 10 $\times$ 20.
l_j	Length of sub-item is uniformly distributed in [2.5, 5].
uc	There are two settings, denoted by low and high, indicating plate cost is set to 0.3 and 3, respectively.
r_{ji}	Number of sub-assemblies is randomly selected from set {1, 2, 3, 4, 5}.
pc_{it}	There are two settings, denoted by low and high, that indicate the value of pc_{it} is set to a coefficient of $\{0.3,\ 3\} \times \sum_{j=1}^{J} 0.1 \cdot r_{ji} \cdot l_j \cdot w$, respectively. The value is independent from period t.
sc_{it}	There are two settings, denoted by low and high, that indicate setup cost sc_{it} is set to $1.5 \cdot pc_{it}$ and $15 \cdot pc_{it}$, respectively.
hc_i	Inventory-holding cost hc_i is set to $\frac{\sum_{t \in \{1,\dots,T\}} 0.01 \cdot pc_{it}}{T}$.
a_{jp}	Cutting patterns that were generated by a priori column generation procedure.
v_{it}	Unit production time v_{it} is set to $\sum_{j=1}^{J} r_{ji} \cdot w$.
st_{it}	There are two settings, denoted by low and high, that indicate setup time st_{it} is set to $0.6 \cdot v_{it}$ and $6 \cdot v_{it}$, respectively.
d_{it}	There are two settings, denoted by low and high, that indicate gross demand amount is uniformly distributed in [0, 15] and [0, 150] in the first period and [0, 60] and [0, 600] for other periods, respectively.
C_t	There are two settings, denoted by low and high, that indicate a coefficient of $\{1.05,\ 0.95\} \times \frac{1.1 \cdot \sum_{i=1}^{I} \sum_{t=1}^{T} (st_{it} + v_{it} \cdot d_{it})}{T}$, respectively.

Additionally, we create SetC and SetD by changing some parameter settings of SetA and SetB, respectively. We keep all parameter settings the same as SetA and SetB except that the production cost pc_{it} is divided by $\sum_{j=1}^{J} 0.1 \cdot r_{ji} \cdot l_j \cdot w$, while the setup costs sc_{it} is multiplied by 30. The purpose of modifying the parameters is to generate test instances with higher TBOs, which are more challenging to solve according to the computational tests. Furthermore, we create SetE and SetF by doubling the size of planning periods and tripling the size of sub-assemblies of SetC and SetD, respectively.

The PS method was implemented using the SPCS formulation in GAMS v24.3, a high-level algebraic modeling language, where IBM Ilog CPLEX 12.6 is set as the LP/IP solver. All computations were run on a PC with Intel(R) Core(TM) 2.8 GHz processor and 16.0 GB RAM. The parameter settings for the PS method was given as follows: Parameter T_s was set to 5 seconds. Parameters (α, β) were respectively set to (4, 2), (3, 2), (2, 1), (3, 1), (4, 3) and (4, 1) to derive the initial population $\mathcal{G}_y$ and to solve subproblems. The size of $\mathcal{G}_y$ was set to 6. Parameters ρ_1 and ρ_2 are set to (2, 2). The number of potential and random subproblems at each iteration were set to 4 and 1, and the size of subset SV was set to $0.30 \cdot |MV|$. We note that we used the column generation algorithm proposed by [24] and [25] to generate cutting patterns a_{jp}, mainly because this was sufficient for test problems with a small number of patterns. We acknowledge that this procedure of generating cutting patterns can be improved by implementing more efficient column generation algorithms ([6]).

7.1 Lower bounds

Table 4: Comparisons of lower bounds for SetA

	LSCS		SPCS		LSCS2		SPCS2	
	LB	time	LB	time	LB	(iter, time, col)	LB	(iter, time, col)
Demand								
Low	3,700,913	0.02	3,842,560	0.02	3,704,009	(623, 212, 4546)	3,843,762	(495, 146, 4361)
High	35,235,210	0.01	35,547,112	0.02	35,240,017	(437, 133, 3559)	35,547,695	(774, 294, 8567)
Production Cost								
Low	4,327,473	0.01	4,369,379	0.02	4,328,208	(553, 182, 4067)	4,369,538	(631, 218, 6399)
High	34,608,650	0.01	35,020,293	0.02	34,615,818	(506, 163, 4038)	35,021,920	(638, 223, 6528)
Plate Cost								
Low	18,660,712	0.02	18,889,104	0.02	18,664,500	(533, 181, 4014)	18,890,084	(645, 220, 6488)
High	20,275,410	0.01	20,500,568	0.02	20,279,526	(526, 164, 4091)	20,501,373	(624, 221, 6440)
TBO								
Low	19,177,144	0.01	19,256,681	0.03	19,177,875	(494, 160, 3635)	19,256,812	(744, 281, 8185)
High	19,758,979	0.01	20,132,990	0.02	19,766,151	(565, 185, 4470)	20,134,646	(525, 160, 4742)
Setup Time								
Low	19,662,931	0.01	19,895,243	0.02	19,667,724	(485, 158, 3950)	19,896,217	(678, 235, 6878)
High	19,273,192	0.01	19,494,428	0.03	19,276,302	(574, 187, 4155)	19,495,240	(591, 206, 6049)
Utilization								
Low	19,437,085	0.01	19,665,485	0.02	19,439,432	(559, 178, 4050)	19,666,133	(611, 212, 6327)
High	19,499,038	0.01	19,724,187	0.03	19,504,594	(500, 167, 4055)	19,725,325	(658, 228, 6600)
Average	19,468,061	0.01	19,694,836	0.02	19,472,013	(530, 172, 4053)	19,695,729	(634, 220, 6464)

The relative computational effectiveness of the lower bound techniques was tested on SetA and SetB. The detailed results are summarized in Tables 4 and 5, where the columns associated with

LSCS and SPCS give the respective lower bounds achieved by the LP relaxation of LSCS and SPCS; and the columns associated with LSCS2 and SPCS2 give respective lower bounds achieved by the column generation procedure of the per-period Dantzig–Wolfe decomposition of LSCS and SPCS. Symbol (iter, time, col) represents the number of iterations, time (minute), and number of columns associated with the relative column generation procedure. The lower bounds associated with the per-item Dantzig–Wolfe decomposition are not listed in these two tables because they are equivalent to the lower bounds achieved by the LP relaxation of the SPCS formulation.

Table 5: Comparisons of lower bounds for SetB

	LSCS		SPCS		LSCS2		SPCS2	
	LB	time	LB	time	LB	(iter, time, col)	LB	(iter, time, col)
Demand								
Low	3,464,744	0.01	3,594,300	0.02	3,465,888	(1008, 408, 7199)	3,595,031	(678, 235, 6158)
High	33,602,414	0.01	33,900,820	0.02	33,603,828	(668, 259, 5353)	33,901,108	(1007, 623, 12270)
Production Cost								
Low	4,115,648	0.01	4,155,195	0.02	4,115,962	(847, 334, 6289)	4,155,277	(835, 454, 9176)
High	32,951,509	0.01	33,339,924	0.02	32,953,755	(830, 333, 6263)	33,340,862	(850, 404, 9251)
Plate Cost								
Low	17,561,790	0.01	17,774,258	0.02	17,562,953	(835, 337, 6267)	17,774,754	(851, 446, 9290)
High	19,505,367	0.01	19,720,862	0.02	19,506,763	(842, 330, 6285)	19,721,386	(834, 412, 9137)
TBO								
Low	18,321,092	0.01	18,397,033	0.02	18,321,393	(740, 288, 5491)	18,397,057	(972, 605, 11709)
High	18,746,066	0.01	19,098,087	0.02	18,748,324	(937, 379, 7061)	19,099,082	(714, 253, 6719)
Setup Time								
Low	18,763,980	0.01	18,985,868	0.02	18,765,487	(824, 337, 6180)	18,986,401	(876, 456, 9733)
High	18,303,177	0.01	18,509,252	0.02	18,304,229	(852, 330, 6372)	18,509,739	(809, 402, 8695)
Utilization								
Low	18,384,044	0.01	18,598,761	0.02	18,384,677	(856, 333, 6254)	18,599,171	(782, 377, 8496)
High	18,683,114	0.01	18,896,359	0.02	18,685,039	(821, 334, 6298)	18,896,969	(903, 481, 9932)
Average	18,533,579	0.01	18,747,560	0.02	18,534,858	(838, 334, 6276)	18,748,070	(843, 429, 9214)

From the results, it can be seen that the LP relaxation of SPCS and Dantzig–Wolfe decompositions improve lower bounds over the LP relaxation of LSCS. We note that the per-period decomposition of LSCS and SPCS consumes a larger amount of computational time. To have a good balance between computational time and lower bound quality, we select the LP relaxation of SPCS as the lower bound technique for the PS method. However, if the computational time is not of concern, the per-period decomposition of SPCS is recommended in order to derive the best lower bounds.

We note that the calculations of the linear relaxation ($\hat{\mathrm{R}}_{\mathrm{LSCS}}$) and all the per-item decompositions ($\hat{\mathrm{D}}_{\mathrm{LSCS1}}$ and $\hat{\mathrm{D}}_{\mathrm{SPCS1}}$) remain the same when integrality on z variables is enforced, since in the per-item decomposition, the BOM constraint (which is the only one containing the z variables) will be in the master problem, in which the variables are relaxed (including the z variables). On the other hand, for the per-period decompositions, we tested the problems with integer z variables, where the sub-problems became very difficult to solve and made it impractical to use these decompositions.

7.2 Preliminary tests for parameter settings

The relative performance of the PS method was tested under different settings of parameters, including G (the size of population $\mathcal{G}_y$), ρ_1, ρ_2, the size of subset SV, and the numbers of potential and random subproblems (N_ν^1 and N_ν^2). For their default setting, we let $\mathcal{G}_y$ be 6; both ρ_1 and ρ_2 be 2; the size of subset SV be $0.30 \cdot |MV|$; and N_ν^1 and N_ν^2 be 4 and 1. The number of cases for these parameters is given as follows:

- G: there are 3 cases including 2, 4, and 6.
- The size of subset SV: there are 7 cases including 0.05, 0.15, 0.30, 0.45, 0.60, 0.75, and 0.90 of $|MV|$.
- ρ_1 and ρ_2: there are 4 cases including (1, 1), (2, 2), (3, 3), and (4, 4).
- N_ν^1 and N_ν^2: there are 3 cases including (2, 1), (4, 1), and (5, 3).

When testing each of the above parameters, the other parameters are set to the default values. Figure 4 compares the performance of the parameter settings by the number of the best solutions and Table 6 shows their average optimality gaps for the 64 test instances in SetA. From the figure and table, we see that the PS method performs the best when each parameter is set to the default values and the performance of the PS method changes but is relatively stable under different parameter settings.

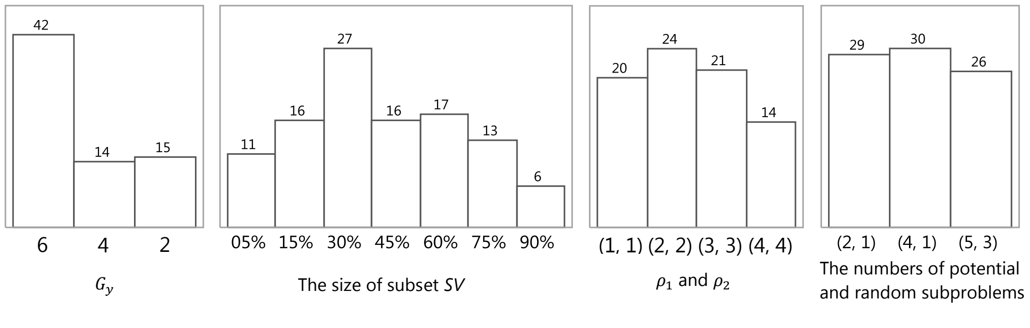

Figure 4: The number of the best solutions achieved under each scenario

The PS method performs better when parameter G is set to 6 compared to 2 and 4. This means that increasing the variety of the population of $\mathcal{G}_y$ may enhance the performance of the PS method. In addition, the PS method performs better when the size of subset SV is set to 0.3 $\cdot|MV|$. Increasing (decreasing) the size makes the subproblems $\mathcal{P}^S$ more similar to the current best

Table 6: Optimality gaps under each scenario

G	PS Gap	$\|SV\|/\|MV\|$	PS Gap	(ρ_1, ρ_2)	PS Gap	(N_ν^1, N_ν^2)	PS Gap
2	0.1871%	0.05	0.1711%	(1, 1)	0.1711%	(2, 1)	0.1603%
4	0.1792%	0.15	0.1686%	(2, 2)	0.1589%	(4, 1)	0.1589%
6	0.1589%	0.30	0.1589%	(3, 3)	0.1650%	(5, 3)	0.1629%
		0.45	0.1709%	(4, 4)	0.1717%		
		0.60	0.1668%				
		0.75	0.1759%				
		0.90	0.1769%				

The lower bounds yielded by the LP relaxation of the SPCS formulation are applied to calculate the PS optimality gaps.

subproblems (to the original problem), which reduces the variety of subproblems and negatively impacts the performance of the PS method. Furthermore, the PS method performs better when ρ_1 and ρ_2 are set to 2. The selection procedure of SV is too little (much) focused on the variables that have high similarities on the solution values of population $\mathcal{G}_y$ when ρ_1 and ρ_2 are set to be too small (high), which negatively impacts the performance of the PS method. Based on our computational experience, the performance of the PS method is not sensitive to the numbers of potential and random subproblems. However, we recommend setting these numbers not too big, in order to avoid slowing down the process of updating subset MV.

7.3 Comparison with other methods for SetA and SetB

We compared the PS approach with the Lagrangian relaxation-based heuristic (denoted as LRH) proposed by [27] and IBM Ilog CPLEX 12.6 (denoted as CPLEX). The PS and CPLEX approaches were implemented in GAMS v24.3 using the SPCS formulation. Detailed parameter settings for the PS approach were described above. For the CPLEX method, IBM Ilog CPLEX 12.6 was set as an IP solver using the default setting. For the LRH approach, using GAMS v24.3, we applied the same formulation (LSCS) to implement the algorithm described in Section 4 of [27]. As shown in the algorithm, a sub-gradient method is employed to update multipliers for the Lagrangian relaxation problems that are then solved to optimality. If the optimal solution of the Lagrangian relaxation problem is not feasible to the original problem, a smoothing heuristic is applied to obtain a feasible solution. The incumbent solution is continuously updated to the current best solution until a stopping criterion is satisfied. We note that there is a slight difference in the LSCS formulation compared to the one proposed in [27], i.e., the LSCS formulation additionally considers setup times in the capacity constraints. It creates a small difference in the objective of the Lagrangian relaxation problems. However, it does not influence the core implementation of the algorithm. The total computational time of 60 seconds was allocated for each test instance. We note that for every instance and every solution method used, we have imposed this maximum time in each computation.

Table 7: Comparisons of solution characteristics for SetA

				LRH			CPLEX			PS	
										Average	
	Pattern #	Time (s)	Inv $	Set #	Plate #	Inv $	Set #	Plate #	Inv $	Set #	Plate #
Demand											
Low	62	48	45,211	118	103,028	45,621	91	103,029	45,394	92	103,030
High	62	42	172,723	184	1,012,526	169,164	170	1,012,527	167,679	172	1,012,526
Production Cost											
Low	62	47	19,869	161	559,246	19,610	130	559,245	19,535	131	559,246
High	62	43	198,065	141	556,307	195,174	131	556,310	193,538	132	556,310
Plate Cost											
Low	62	45	111,857	144	550,200	111,116	130	550,201	109,749	131	550,201
High	62	45	106,078	158	565,353	103,669	131	565,354	103,324	132	565,355
TBO											
Low	62	45	38,578	180	552,046	36,026	169	552,044	35,963	170	552,043
High	62	45	179,357	123	563,508	178,759	92	563,512	177,110	94	563,512
Setup Time											
Low	62	45	103,965	159	559,754	102,812	134	559,757	101,066	136	559,758
High	62	45	113,969	143	555,799	111,973	127	555,799	112,007	128	555,798
Utilization											
Low	62	45	111,934	145	558,175	108,885	131	558,177	110,298	132	558,178
High	62	45	106,001	157	557,378	105,900	130	557,379	102,775	132	557,378
Average	**62**	**45**	**108,967**	**151**	**557,777**	**107,392**	**131**	**557,778**	**106,536**	**132**	**557,778**

The standard deviations of Inv $, Set #, and Plate # for the PS method are 2,276, 0.86, and 3.34 on average, respectively.

Table 8: Comparisons of solution characteristics for SetB

				LRH			CPLEX			PS	
										Average	
	Pattern #	Time (s)	Inv $	Set #	Plate #	Inv $	Set #	Plate #	Inv $	Set #	Plate #
Demand											
Low	46	36	42,368	147	95,770	43,330	111	95,771	43,197	114	95,772
High	47	35	167,202	228	965,755	161,960	213	965,758	158,570	215	965,761
Production Cost											
Low	46	36	18,708	199	531,792	18,286	163	531,794	18,194	165	531,799
High	46	35	190,863	176	529,734	187,004	161	529,736	183,573	164	529,733
Plate Cost											
Low	46	36	104,689	176	519,997	104,148	161	519,995	101,875	163	519,995
High	47	35	104,882	199	541,528	101,143	163	541,535	99,892	165	541,537
TBO											
Low	47	37	37,379	224	528,531	35,329	211	528,530	35,082	212	528,530
High	45	34	172,191	151	532,995	169,961	114	532,999	166,685	116	533,002
Setup Time											
Low	46	36	101,606	197	535,117	100,767	165	535,121	98,529	167	535,125
High	46	35	107,965	178	526,408	104,523	159	526,408	103,238	161	526,408
Utilization											
Low	46	35	105,136	184	522,327	103,085	163	522,326	101,533	164	522,327
High	47	36	104,435	192	539,199	102,205	161	539,203	100,233	164	539,206
Average	**46**	**36**	**104,785**	**188**	**530,763**	**102,645**	**162**	**530,765**	**100,883**	**164**	**530,766**

The standard deviations of Inv $, Set #, and Plate # for the PS method are 1,704, 0.87, and 4.65 on average, respectively.

First, we discuss some important solution characteristics as presented in Tables 7 and 8, which indicate the number of cutting patterns (Pattern #), computational time of generating the cutting patterns (Time (s)), and comparisons of LRH, CPLEX, and PS solutions in terms of total inventory-holding costs (Inv $), total number of setups (Set #) and total number of plates (Plate #), where the presentation is classified according to problem characteristics such as demand and production cost. Since PS is an algorithm with randomness, we executed it four times for each test instance and therefore, the tables present the "average" and "standard deviation" values obtained from these four tests. An interesting aspect of the results is that the difference among LRH, CPLEX, and PS solutions is quite minimal for the number of plates used, but it seems to vary significantly more with respect to the number of setups and inventory-holding costs. The PS approach has an obvious advantage of reducing inventory-holding costs and number of setups compared to the LRH method, while the PS approach has a close performance compared to CPLEX.

Next, we present details on upper bounds as well as optimality gaps in Tables 9 and 10. The second, third, and fourth columns list upper bounds obtained by the LRH, CPLEX, and PS methods, respectively. The remaining columns indicate optimality gaps calculated in the fashion $(UB - LB)/UB$ for the LRH, CPLEX, and PS methods, where two different lower bounds are used, including those obtained by the LP relaxation of the SPCS formulation (SPCS) and the per-period Dantzig–Wolfe decomposition of the SPCS formulation (SPCS2). Similar to the previous two tables, the figures for PS indicate both averages and standard deviations for four runs accomplished for each instance.

Table 9: Comparisons of bounds and optimality gaps for SetA

	Upper bounds			LRH Gaps		CPLEX Gaps		PS Gaps	
								Average	
	LRH	CPLEX	PS	SPCS	SPCS2	SPCS	SPCS2	SPCS	SPCS2
Demand									
Low	3,929,753	3,851,266	3,856,781	2.76%	2.73%	0.19%	0.17%	0.32%	0.29%
High	35,584,998	35,551,227	35,554,078	0.14%	0.14%	0.01%	0.01%	0.02%	0.02%
Production Cost									
Low	4,393,407	4,370,408	4,371,130	1.95%	1.94%	0.09%	0.07%	0.15%	0.14%
High	35,121,345	35,032,086	35,039,729	0.95%	0.93%	0.12%	0.10%	0.19%	0.17%
Plate Cost									
Low	18,913,143	18,895,964	18,900,143	1.21%	1.19%	0.12%	0.10%	0.19%	0.17%
High	20,601,608	20,506,530	20,510,716	1.70%	1.68%	0.09%	0.07%	0.15%	0.14%
TBO									
Low	19,262,975	19,257,327	19,257,619	0.10%	0.10%	0.01%	0.01%	0.01%	0.01%
High	20,251,777	20,145,166	20,153,240	2.80%	2.77%	0.20%	0.17%	0.33%	0.30%
Setup Time									
Low	19,975,922	19,903,192	19,906,936	1.78%	1.76%	0.13%	0.11%	0.18%	0.16%
High	19,538,829	19,499,302	19,503,923	1.12%	1.11%	0.08%	0.06%	0.16%	0.15%
Utilization									
Low	19,709,921	19,669,585	19,674,221	1.09%	1.08%	0.08%	0.06%	0.16%	0.15%
High	19,804,830	19,732,909	19,736,638	1.81%	1.79%	0.13%	0.11%	0.18%	0.16%
Average	**19,757,376**	**19,701,247**	**19,705,429**	**1.45%**	**1.44%**	**0.10%**	**0.09%**	**0.17%**	**0.16%**

The standard deviations of the optimality gaps SPCS and SPCS2 for the PS method are 0.02% and 0.02% on average, respectively.

From the results, we observe that the PS approach offers solutions superior to the LRH method across various parameter settings with respect to the upper bounds. We also note that the PS method showed, in general, a stable performance over the four runs for each instance as indicated by small standard deviations. In addition, when comparing the PS approach with the LRH method, there are big differences in the total inventory-holding costs and the total number of setups but smaller differences in the upper bounds. This is impacted by the other components in the problem objective: the production and plate costs, for which the PS approach has a relative smaller improvement over the LRH method. In order to know whether the difference in the upper bounds would be more highlighted if the parameters of production and plate costs (pc_{it} and uc) are smaller,

Table 10: Comparisons of bounds and optimality gaps for SetB

	Upper bounds			LRH Gaps		CPLEX Gaps		PS Gaps	
								Average	
	LRH	CPLEX	PS	SPCS	SPCS2	SPCS	SPCS2	SPCS	SPCS2
Demand									
Low	3,687,177	3,599,505	3,605,795	2.88%	2.86%	0.13%	0.11%	0.27%	0.26%
High	33,937,166	33,903,388	33,906,535	0.12%	0.12%	0.01%	0.01%	0.02%	0.02%
Production Cost									
Low	4,176,120	4,155,891	4,156,687	1.91%	1.90%	0.06%	0.05%	0.12%	0.12%
High	33,448,223	33,347,002	33,355,643	1.09%	1.08%	0.07%	0.06%	0.17%	0.16%
Plate Cost									
Low	17,791,612	17,778,074	17,783,081	1.14%	1.13%	0.07%	0.06%	0.16%	0.15%
High	19,832,731	19,724,820	19,729,249	1.86%	1.85%	0.07%	0.06%	0.13%	0.13%
TBO									
Low	18,401,873	18,397,429	18,397,782	0.09%	0.09%	0.00%	0.00%	0.01%	0.01%
High	19,222,469	19,105,465	19,114,548	2.90%	2.89%	0.13%	0.11%	0.28%	0.26%
Setup Time									
Low	19,056,783	18,990,480	18,995,328	1.73%	1.72%	0.08%	0.07%	0.15%	0.14%
High	18,567,560	18,512,413	18,517,002	1.27%	1.26%	0.06%	0.05%	0.14%	0.13%
Utilization									
Low	18,645,452	18,601,340	18,605,980	1.17%	1.16%	0.05%	0.04%	0.13%	0.12%
High	18,978,890	18,901,553	18,906,350	1.82%	1.81%	0.08%	0.07%	0.16%	0.15%
Average	**18,812,171**	**18,751,447**	**18,756,165**	**1.50%**	**1.49%**	**0.07%**	**0.06%**	**0.15%**	**0.14%**

The standard deviations of the optimality gaps SPCS and SPCS2 for the PS method are 0.01% and 0.01% on average, respectively.

we performed two extra tests for all instances with high demand: the first test reduces pc_{it} by 80% and keeps all other parameters unchanged; and the second test further reduces uc by 80%. The differences in upper bounds between LRH and PS for these two tests are respectively 1.3% and 1.7%, relative to 0.5% for the original test.

We performed additional tests for SetA by setting the computational time to 1,600 seconds. The optimality gaps based on the LP relaxation of SPCS, on average, drops from 0.17% to 0.15%. With additional computational time allowed, the optimality gaps will be reduced further but the reduction rate is deteriorating. However, the lower bound has a gap relative to the optimal objective. As such, we will not achieve an optimality gap of zero even when the PS method finds the optimal solution. Furthermore, the PS method is a heuristic that cannot guarantee an optimal solution. It improves the solution quality faster in the early iterations, and then it slows down when the incumbent upper bound is getting closer to the optimal objective. At this point, it is difficult to find a better solution because most points in the neighborhood have a worse solution compared to the incumbent.

When comparing the PS method with the CPLEX method, there are small differences in the upper bounds, the total inventory-holding costs, and the total number of setups. However, the CPLEX method achieves slightly better upper bounds that are reflected by a smaller number of setups. The CPLEX method thoroughly searches all possible solutions using a branch-and-bound technique and aims to find an optimal solution, and it may have advantages over heuristics for problems that are not complicated to solve. However, when the problems become complicated or the problem sizes become large, the CPLEX method may be limited on finding a good solution under a reasonable amount of computational time. In the next subsection, we will test four other sets that contain more complicated test problems, for which the PS method can achieve better solutions than the CPLEX method.

Table 11: Computational results of PS-Det and PS-Integer

	SetA				SetB			
	Upper bounds		Gaps		Upper bounds		Gaps	
	Det	Int	Det	Int	Det	Int	Det	Int
Demand								
Low	3,856,533	3,878,526	0.330%	0.870%	3,606,637	3,622,015	0.291%	0.724%
High	35,554,490	35,576,754	0.019%	0.082%	33,906,604	33,925,559	0.016%	0.073%
Production Cost								
Low	4,371,355	4,376,287	0.165%	0.467%	4,156,736	4,160,933	0.130%	0.394%
High	35,039,668	35,078,993	0.185%	0.485%	33,356,505	33,386,642	0.177%	0.403%
Plate Cost								
Low	18,899,885	18,918,103	0.184%	0.471%	17,783,426	17,796,808	0.161%	0.378%
High	20,511,138	20,537,177	0.165%	0.481%	19,729,815	19,750,766	0.146%	0.420%
TBO								
Low	19,257,721	19,259,881	0.015%	0.089%	18,397,862	18,399,805	0.013%	0.077%
High	20,153,302	20,195,400	0.335%	0.863%	19,115,379	19,147,769	0.294%	0.720%
Setup Time								
Low	19,907,464	19,930,785	0.180%	0.499%	18,995,788	19,015,145	0.157%	0.432%
High	19,503,559	19,524,496	0.170%	0.453%	18,517,453	18,532,430	0.150%	0.365%
Utilization								
Low	19,674,493	19,692,916	0.164%	0.425%	18,606,860	18,620,149	0.151%	0.370%
High	19,736,530	19,762,365	0.186%	0.527%	18,906,380	18,927,425	0.156%	0.427%
Average	**19,705,512**	**19,727,640**	**0.175%**	**0.476%**	**18,756,620**	**18,773,787**	**0.153%**	**0.399%**

Table 11 shows the results of the deterministic version of the PS method (*Det*), and the results of the PS method when solving the problem with integer z variables (*Int*), where the same computational time limits were used as in previous tests. In the deterministic version, we generated subproblems by selecting the variables with the highest P_q values. Iteratively, when some variables in FV are fixed, we also just fix the variables with the highest P_q. The main differences of the deterministic version compared to the original PS algorithm are: i) SV is selected deterministically using the highest P_q, and ii) variables with the highest P_q in FV are fixed, instead of randomly picking variable to fix according to P_q values. On the other hand, for the PS method considering integer z variables, we have explicitly incorporated the integrality on the z variables within the heuristic, where the calculation of upper bounds becomes more difficult compared to the relaxed case. More specifically, as described in Section 5.2, the relax-and-fix heuristic partitions the set of all periods $t \in \{1, ..., T\}$ into three subsets of setup decision variables y_{it} in order to create restricted subproblems, and then solves the subproblems iteratively. To incorporate the integral z variables, the heuristic still partitions the set of all periods $t \in \{1, ..., T\}$ into the same three subsets but both for setup decision variables y_{it} and for plate number z_{pt} to create subproblems, and then solves the subproblems iteratively. In the heuristic, y_{it} and z_{pt} are treated similarly in terms of their index t

being used for the partitioning into three subsets.

As the results indicate, the deterministic version of the PS method results in only slightly worse solutions on average, where the difference is minimal. On the other hand, enforcing integrality on z variables results in higher duality gaps as expected. However, the average gaps remain reasonably low.

7.4 Comparison with other methods for harder problems

We additionally tested the relative computational effectiveness of the LRH, CPLEX, and PS methods on SetC, SetD, SetE, and SetF and summarized the detailed results in Tables 12 and 13. In the tables, the columns associated with CPLEX and PS give respective upper bounds and optimality gaps achieved by the CPLEX and PS methods, and the optimality gaps were calculated using the lower bounds obtained by the LP relaxation of the SPCS formulation. We omitted the results of LRH because they are consistently worse than the results of CPLEX and PS. We allocated a total computational time of 60 seconds for each test instance in SetC and SetD and 600 seconds for instances in SetE and SetF (because of the complexity). Similar to the previous tables, the figures for PS indicate both average and standard deviation for four runs accomplished for each instance.

Table 12: Comparisons of bounds and optimality gaps for SetC and SetD

	SetC				SetD			
	Upper bounds		Optimality gaps		Upper bounds		Optimality gaps	
	CPLEX	PS	CPLEX	PS	CPLEX	PS	CPLEX	PS
Demand								
Low	7,420,684	7,358,302	3.80%	2.89%	6,727,147	6,692,440	2.81%	2.09%
High	10,637,069	10,514,561	2.14%	1.52%	9,969,074	9,892,513	1.56%	1.07%
Production Cost								
Low	2,401,762	2,386,645	2.52%	1.93%	2,250,873	2,239,698	1.86%	1.32%
High	15,655,991	15,486,218	3.41%	2.48%	14,445,348	14,345,254	2.51%	1.84%
Plate Cost								
Low	8,386,180	8,284,487	3.45%	2.53%	7,587,848	7,522,343	2.42%	1.71%
High	9,671,573	9,588,375	2.49%	1.88%	9,108,373	9,062,610	1.95%	1.45%
TBO								
Low	2,631,776	2,619,000	2.14%	1.60%	2,511,511	2,499,383	1.61%	1.06%
High	15,425,977	15,253,863	3.80%	2.81%	14,184,710	14,085,569	2.76%	2.10%
Setup Time								
Low	9,650,242	9,547,234	2.88%	2.16%	8,938,273	8,877,412	2.15%	1.51%
High	8,407,511	8,325,629	3.06%	2.25%	7,757,948	7,707,540	2.22%	1.65%
Utilization								
Low	7,981,115	7,889,451	2.86%	2.16%	7,298,835	7,255,250	2.08%	1.60%
High	10,076,638	9,983,412	3.08%	2.25%	9,397,386	9,329,702	2.29%	1.56%
Average	**9,028,877**	**8,936,431**	**2.97%**	**2.20%**	**8,348,111**	**8,292,476**	**2.18%**	**1.58%**

The standard deviations of the optimality gap for the PS method are 0.19% and 0.17% on average for SetC and SetD, respectively.

From the results, we observe that the PS approach offers upper bounds and optimality gaps superior to the CPLEX method across various parameter settings. We also note that the PS method showed, in general, a stable performance over the four runs for instances as indicated by relatively small standard deviations. Additionally, as indicated by the optimization gaps, the instances in SetC and SetD are more complicated to solve than the instances in SetA and SetB.

Table 13: Comparisons of bounds and optimality gaps for SetE and SetF

	SetE					SetF				
	Upper bounds		Optimality gaps			Upper bounds		Optimality gaps		
	CPLEX	PS	CPLEX	CPLEX2*	PS	CPLEX	PS	CPLEX	CPLEX2*	PS
Demand										
Low	33,297,188	31,519,414	7.93%	6.00%	4.13%	29,583,292	28,793,473	5.72%	4.91%	3.58%
High	52,323,886	51,092,881	3.90%	2.94%	2.19%	48,616,735	47,630,377	2.99%	2.40%	1.77%
Production Cost										
Low	12,570,244	12,268,587	5.14%	3.79%	2.50%	11,671,038	11,511,985	3.55%	3.11%	2.25%
High	73,050,830	70,343,708	6.69%	5.15%	3.82%	66,528,989	64,911,864	5.15%	4.20%	3.10%
Plate Cost										
Low	37,826,473	36,875,607	6.47%	5.16%	3.81%	34,896,214	34,194,592	4.82%	4.09%	3.13%
High	47,794,601	45,736,689	5.36%	3.79%	2.51%	43,303,813	42,229,257	3.88%	3.22%	2.23%
TBO										
Low	14,187,262	14,028,370	3.72%	2.88%	2.26%	13,368,893	13,234,388	2.83%	2.37%	1.71%
High	71,433,812	68,583,925	8.11%	6.07%	4.06%	64,831,134	63,189,461	5.88%	4.94%	3.64%
Setup Time										
Low	45,291,880	43,679,730	5.75%	4.33%	2.90%	41,701,139	40,814,920	4.35%	3.64%	2.67%
High	40,329,194	38,932,565	6.08%	4.62%	3.42%	36,498,888	35,608,930	4.36%	3.67%	2.69%
Utilization										
Low	37,318,912	36,001,992	5.94%	4.47%	3.03%	33,198,300	32,619,723	4.07%	3.51%	2.74%
High	48,302,162	46,610,303	5.89%	4.48%	3.29%	45,001,727	43,804,126	4.63%	3.80%	2.62%
Average	**42,810,537**	**41,306,148**	**5.92%**	**4.47%**	**3.16%**	**39,100,014**	**38,211,925**	**4.35%**	**3.65%**	**2.68%**

* CPLEX2 indicates the optimality gaps achieved by the CPLEX method when the time limit is set to 3,600 seconds. The standard deviations of the optimality gap for the PS method are 0.34% and 0.28% on average for SetE and SetF, respectively.

Even more challenging problems are the instances in SetE and SetF, which have more planning periods, more types of sub-assemblies, and more cutting patterns. We observe that the instances in SetE and SetF have 182 and 137 cutting patterns on average, respectively. The PS method has more advantages over the CPLEX method for the larger instances in SetE and SetF. We note that the PS method still achieved better solutions even when the CPLEX method is allowed 3,600 seconds on the computational time limits.

Regarding the solutions of the total inventory-holding costs and the total number of setups and plates, the PS method can reduce the number of setups but make a less noticeable improvement on the inventory-holding costs and the total number of plates for instances in SetC and SetD when compared to the CPLEX method. However, the PS method can both reduce the number of setups from 142 to 133 and inventory-holding costs from 2,216,181 to 2,183,260 on average for instances in SetE and SetF, although the improvement on the total number of plates is still very small for these instances.

8. Conclusions and future research

This paper proposes a progressive selection method and a number of new Dantzig–Wolfe decomposition and column generation methods for the LS-CS problem. All of these Dantzig–Wolfe decomposition methods are capable of deriving better lower bounds than the LP relaxation of the original formulation. This paper theoretically and computationally evaluates their relative effectiveness such that the one with the best performance is selected and combined into the PS approach. Extensive computational tests show that the PS approach can achieve better solutions than the Lagrangian relaxation-based heuristic proposed by [27] and perform better for complicated problems when compared to the CPLEX method. The current per-period Dantzig–Wolfe decomposition consumes relatively long computational time. It is interesting to explore if replacing the exact pricing with dual-feasible functions ([13]) can reduce computational time while preserving the bound quality. It is also interesting to apply the two-period convex hull closures ([1]) and the similar horizon decomposition approach ([21]) to improve lower bounds for the LS-CS problem.

The performance of the PS approach is extensively analyzed by statistical methodologies, such as boxplots, Welch's test, linear regression and finite mixture models. The statistical analyses show that solution values of the combined exact methods and heuristics can provide statistically significant domain knowledge on the optimal solution value. Meanwhile, the statistical analyses show that the domain-knowledge-guided subproblems can derive much better solution qualities than those subproblems generated without using the domain knowledge. Besides its applications on the PS approach, we believe that such domain knowledge has the potential to be applied to other heuristics, such as simulated annealing, tabu search, neighborhood search and genetic algorithm, to guide their search procedure. This will be an interesting area for future research.

Besides the application to the LS-CS problem, the PS method is a generic framework that has the potential to solve various classes of BMIP problems. The framework possesses flexibility in selecting the combined exact methods and heuristics and does not force any restrictions on inside-technologies of these methods. One generic application of this framework is that any existing exact methods and heuristics, that have been applied to a BMIP problem, can be adapted into the framework. The potential benefit is that, unlike some methods applied to the complete large problem consuming a large amount of computational time and still possibly achieving an undesired solution quality, in our framework, different methods are combined to solve smaller subproblems that consume a much smaller amount of computational time and potentially improve the solution quality when each subproblem is solved.

Future work along this line of research should focus on implementing the method to other MIP problems, and exploring theoretical results on how lower and upper bounds can be used to define intelligent partitioning approaches that improve the efficiency of the PS method. In addition, the LS-CS problem can be integrated with cut-and-scheduling problems, similar to the research in [7], [9], and [11]. There are other LS-CS problems in furniture and other practical settings that involve relevant setup time and costs in the cutting and drilling processes, which are different from the one studied in this paper. More details on some of these problems can be found in [5]. These problems are also very difficult to solve and could be interesting topics for future research, for instance, by appropriately modifying and applying the present proposed approach.

References

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

[2] K. Akartunalı and A. J. Miller. A heuristic approach for big bucket multi-level production planning problems. *European Journal of Operational Research*, 193(2):396–411, 2009.

[3] K. Akartunalı and A. J. Miller. A computational analysis of lower bounds for big bucket production planning problems. *Computational Optimization and Applications*, 53(3):729–753, 2012.

[4] D. Alem and R. Morabito. Production planning in furniture settings via robust optimization. *Computers and Operations Research*, 39(2):139–150, 2012.

[5] D. Alem and R. Morabito. Risk-averse two-stage stochastic programs in furniture plants. *OR Spectrum*, 35(4):773–806, 2013.

[6] H. B. Amor and J. V. de Carvalho. *Cutting stock problems*, pages 131–161. In Column Generation [17]. 2005.

[7] C. Arbib, F. di Iorio, F. Marinelli, and F. Rossi. Cutting and reuse: an application from automotive component manufacturing. *Operations Research*, 50(6):923–934, 2002.

[8] C. Arbib and F. Marinelli. Integrating process optimization and inventory planning in cutting-stock with skiving option: an optimization model and its application. *European Journal of Operational Research*, 163(3):617–630, 2005.

[9] C. Arbib and F. Marinelli. On cutting stock with due dates. *Omega*, 46:11–20, 2014.

[10] M.O. Ball. Heuristics based on mathematical programming. *Surveys in Operations Research and Management Science*, 16(1):21 – 38, 2011.

[11] N. Braga and C. Alves. A model-based heuristic for the combined cutting stock and scheduling problem. *Computational Science and its Applications*, in: O. Gervasi et al. (eds.)(LNCS 9156):490–505, 2015.

[12] J. M. V. D. Carvalho and A. J. G. Rodriguez. An LP-based approach to a two-stage cutting stock problem. *European Journal of Operational Research*, 84(3):580–589, 1995.

[13] F. Clautiaux, C. Alves, and J. V. de Carvalho. A survey of dual-feasible and superadditive functions. *Annals of Operations Research*, 179(1):317–342, 2010.

[14] I. Coverdale and F. Wharton. An improved heuristic procedure for a nonlinear cutting stock problem. *Management Science*, 23(1):78–86, 1976.

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

[16] Z. Degraeve and M. Peeters. Optimal integer solutions to industrial cutting-stock problems: Part 2, benchmark results. *INFORMS Journal on Computing*, 15(1):58–81, 2003.

[17] G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors. *Column Generation*. Springer, 2005.

[18] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1-3):229–237, 1999.

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

[20] H. Foerster and G. Wäscher. Pattern reduction in one dimensional cutting stock problems. *International Journal of Production Research*, 38(7):1657–1676, 2000.

[21] I. Fragkos, Z. Degraeve, and B. De Reyck. A horizon decomposition approach for the capacitated lot-sizing problem with setup times. *INFORMS Journal on Computing*, 28(3):465 – 482, 2016.

[22] M. Gendreau and J.-Y. Potvin, editors. *Handbook of Metaheuristics*. Springer, 2010.

[23] A. M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[24] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem. *Operations Research*, 9:849âĂŞ–859, 1961.

[25] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problem - part ii. *Operations Research*, 13:94–âĂŞ120, 1963.

[26] M.C.N. Gramani and P.M. França. The combined cutting stock and lot-sizing problem in industrial processes. *European Journal of Operational Research*, 174(1):509–521, 2006.

[27] M.C.N. Gramani, P.M. França, and M.N. Arenales. A Lagrangian relaxation approach to a coupled lot-sizing and cutting stock problem. *International Journal of Production Economics*, 119(2):219–227, 2009.

[28] M.C.N. Gramani, P.M. França, and M.N. Arenales. A linear optimization approach to the combined production planning model. *Journal of the Franklin Institute*, 348(7):1523–1536, 2011.

[29] R. W. Haessler. A heuristic programming solution to a nonlinear cutting stock problem. *Management Science*, 17(12):793–802, 1971.

[30] R. W. Haessler. Controlling cutting pattern changes in one-dimensional trim problems. *Operations Research*, 23(3):483–493, 1975.

[31] R. W. Haessler. Selection and design of heuristic procedures for solving roll trim loss problems. *Management Science*, 34(12):1460–1471, 1988.

[32] L. C. Hendry, K. K. Fok, and K. W. Shek. A cutting stock scheduling problem in the copper industry. *Journal of the Operational Research Society*, 47(1):38–47, 1996.

[33] R. Jans and Z. Degraeve. Improved lower bounds for the capacitated lot sizing problem with setup times. *Operations Research Letters*, 32:185–195, 2004.

[34] E. V. Krichagina, R. Rubio, M. I. Taksar, and L. M. Wein. A dynamic stochastic stock-cutting problem. *Operations Research*, 46(5):690–701, 1998.

[35] P. Lefrançois and A. Gascon. Solving a one-dimensional cutting-stock problem in a small manufacturing firm: A case study. *IIE Transactions*, 27(4):483–496, 1995.

[36] Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[37] R. Morabito and M. Arenales. Optimizing the cutting of stock plates in a furniture company. *International Journal of Production Research*, 38(12):2725–2742, 2000.

[38] S. L. Nonås and A. Thorstenson. A combined cutting-stock and lot-sizing problem. *European Journal of Operational Research*, 120(2):327–342, 2000.

[39] S. L. Nonås and A. Thorstenson. Solving a combined cutting-stock and lot-sizing problem with a column generating procedure. *Computers and Operations Research*, 35(10):3371–3392, 2008.

[40] Y. Pochet and L.A. Wolsey, editors. *Production Planning by Mixed Integer Programming.* Springer, 2006.

[41] S. C. Poltroniere, K. C. Poldi, F. M. B. Toledo, and M. N. Arenales. A coupling cutting stock-lot sizing problem in the paper industry. *Annals of Operational Research*, 157(1):91–104, 2008.

[42] M.P. Reinders. Cutting stock optimization and integral production planning for centralized wood processing. *Mathematical and Computer Modeling*, 16(1):37–55, 1992.

[43] H. Stadtler. Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research*, 51(3):487–502, 2003.

[44] C. L. Stowers and U. S. Palekar. Lot sizing problems with strong set-up interactions. *IIE Transactions*, 29(2):167–179, 1997.

[45] R. Vahrenkamp. Random search in the one-dimensional cutting stock problem. *European Journal of Operational Research*, 95(1):191–200, 1996.

[46] F. Vanderbeck. Exact algorithm for minimising the number of setups in the one-dimensional cutting stock problem. *Operations Research*, 48(6):915–926, 2000.

[47] T. Wu, L. Shi, J. Geunes, and K. Akartunalı. An optimization framework for solving capacitated multi-level lot-sizing problems with backlogging. *European Journal of Operational Research*, 214(2):428–441, 2011.

[48] T. Wu, L. Shi, J. Geunes, and K. Akartunalı. On the equivalence of strong formulations for capacitated multi-level lot sizing problems with setup times. *Journal of Global Optimization*, 53(4):615–639, 2012.