Algorithms for the One-Dimensional Two-Stage Cutting Stock Problem

İbrahim Muter*

School of Management, University of Bath, Claverton Down, Bath BA2 7AY, UK,

Zeynep Sezer

Department of Industrial Engineering, Bahçeşehir University, Çırağan Cad. No:4 Beşiktaş, Istanbul 34353, Turkey

Abstract

In this paper, we consider the two-stage extension of the one-dimensional cutting stock problem which arises when technical requirements inhibit the cutting of large stock rolls to demanded widths of finished rolls directly. Therefore, the demands on finished rolls are fulfilled through two subsequent cutting processes, in which the rolls produced in the former are used as an input for the latter, while the number of stock rolls used is minimized.We tackle the pattern-based formulation of this problem which has exponentially many variables and constraints. The special structure of this formulation induces both a column-wise and a row-wise increase when solved by column generation. We design an exact simultaneous column-and-row generation algorithm whose novel element is a row-generating subproblem that generates a set of columns and rows. For this subproblem, which is modeled as an unbounded knapsack problem, we propose three algorithms: implicit enumeration, column generation which renders the overall methodology nested column generation, and a hybrid algorithm. The latter two are integrated in a well-known knapsack algorithm which forges a novel branch-and-price algorithm for the row-generating subproblem. Extensive computational experiments are conducted, and performances of the three algorithms are compared.

Keywords: cutting; two-stage cutting stock problem; column-and-row generation; problems with column-dependent-rows.

1. Introduction.

The cutting stock problem is one of the oldest and most inspiring problems of operations research that has many applications in practice. A vast literature is dedicated to this problem and its extensions (See Dyckhoff (1990) and Wäscher et al. (2007) for the unifying classification of the extensions of this problem). An important class of cutting stock problems arise when stock rolls are cut into demanded finished rolls in more than one stage due to technical

*Corresponding author.

E-mail Addresses: i.muter@bath.ac.uk (İbrahim Muter), zeynep.sezer@bahcesehir.edu.tr (Zeynep Sezer)

restrictions. These restrictions are usually caused by the processing capacity of the equipment available, e.g. the number of knives and the maximum allowable width that can be processed, or by the nature of the production process, e.g. when additional processing is required for orders that have different characteristics. These problems are referred to as the one-dimensional multi-stage cutting stock (MSCS) problems and have reportedly occurred in paper, film and steel industries. Note that, in the literature, some multi-dimensional cutting stock problems (Gilmore and Gomory (1965), Vanderbeck (2001)) are also referred to as MSCS problems. In our case, however, the rolls are defined only by their widths and the demand on finished rolls are fulfilled through successive stages of cutting operations over the same dimension. In each stage of the MSCS problem, larger rolls are cut into smaller items, described by a set of cutting patterns, and are used either as an input for the next stage or to fulfill orders at the final stage. Hence, the number of rolls available in any intermediate stage is limited to its production in a previous stage. In this paper, we particularly focus on the one-dimensional two-stage cutting stock (TSCS) problem, in which stock rolls are first cut into intermediate rolls whose widths are not known a priori but are restricted to lie within a specified interval, and then, in the second stage, finished rolls of demanded widths are produced from these intermediate rolls. Therefore, the fundamental sets of this problem are the two types of cutting patterns, namely the first- and the second stage cutting patterns which are cut from stock rolls and intermediate rolls, respectively. Overall, the TSCS problem determines the number of cutting patterns used in each stage to ensure that the demand on finished rolls are fulfilled at the end of the second stage cutting process, while minimizing the total number of stock rolls used.

The concept of TSCS problem first appeared in a study by Haessler (1971), in which he considered the reprocessing costs of stock rolls that could not be processed in the first run due to the slitting capacity of the equipment and the low usage levels of patterns. However, the cutting patterns are manually generated and the emphasis is on scheduling of these cutting patterns to minimize the associated costs. The TSCS problem that involves generation of the cutting patterns in both stages of the cutting operation was introduced first in Haessler (1979) and later in Ferreira et al. (1990). In both these studies ordered items have different specifications in addition to their width, which requires them to be grouped together into intermediate rolls to reduce the additional processing costs. This grouping allows orders having different processing requirements to be combined across the width of a stock roll. Furthermore, the widths of intermediate rolls used are dependent on the cutter positions in the first stage which causes a setup cost to be incurred when changed. To solve the resulting nonlinear programming problem, which minimizes both trim loss and setup, Haessler (1979) and Ferreira et al. (1990) proposed heuristic procedures. The TSCS problem described in this paper does not take into account any setups, the number of different intermediate rolls used nor does it define finished roll set with something other than its width. Valério de Carvalho and Rodrigues (1995), on the other hand, simplified the TSCS problem by using only homogeneous intermediate rolls, i.e. composed of identical finished rolls of the same width. The authors solved the linear programming (LP) model of this problem using column generation and formulated the auxiliary knapsack problem in terms of the number of intermediate rolls which are generated by a dynamic programming procedure using pre-processing.

Several researchers (Marques and Arenales (2007), Hoto et al. (2007), Leão et al. (2011))

investigated a special version of the TSCS problem which considers the loading of classes of items into a single knapsack with compartments. Each compartment is designated to items from the same class, and incurs a fixed cost in the objective that maximizes the total value of the loaded items. The analogy between this problem and the TSCS problem is that compartments, as intermediate rolls, are bounded by a minimum and a maximum value, and when merged, form a loading cutting pattern. In majority of these studies heuristic algorithms are developed to solve the constrained version of the problem, in which the number of items loaded in the knapsack are restricted. The only exact solution algorithm, proposed by Hoto et al. (2007), is for the unconstrained version and is based on the pre-enumeration of compartments which results in solving a large number of knapsack problems. In a related study presented by Johnston and Khan (1995), a priori bounds on knapsack capacities at each stage of a nested knapsack problem are investigated.

Cutting stock problems are, in general, defined by column-based formulations in which variables correspond to cutting patterns. To overcome the difficulty caused by the large number of columns, Gilmore and Gomory (1961) proposed a column generation algorithm to solve the LP relaxation of this problem, which minimizes the number of stock rolls cut to satisfy the demand on finished rolls. Since then, it has become one of the most prominent methods to solve large-scale problems and has prevailed as the main tool for cutting stock problems, including the TSCS problem. This method initializes a restricted master problem (RMP) with a small subset of columns. The initial basic solution is then iteratively updated by columns with negative reduced cost which are identified by solving a suitable pricing subproblem (PSP). The LP problem is solved to optimality when the solution of the PSP no longer provides a negative reduced cost column. In order to obtain the integer optimal solution, column generation is integrated in a branch-and-bound framework, which is referred to as branch-and-price. The interested reader is referred to Lübbecke and Desrosiers (2005) and Barnhart et al. (1998) for the details of column generation and branch-and-price, respectively. The difficulty of the direct application of column generation to the TSCS problem is caused by the unknown widths of the intermediate rolls. Depending on the interval defined for intermediate roll widths and the number of different finished roll widths, the cardinality of the intermediate roll set can be prohibitively large. Hence, applying column generation based on pre-enumeration of intermediate rolls is not viable for solving practical problems.

The pattern-based formulation of the TSCS problem was first presented by Zak (2002a,b). This formulation has a special structure that qualifies it as a problem with column-dependent-rows, a generic class of problems introduced by Muter et al. (2013a) which possesses distinct characteristics when solved by column generation. The curious structure of these problems stems from the structural linking constraints which are dependent on the set of variables. Therefore, when the RMP is formed with a subset of variables, only the linking constraints induced by these variables exist in the RMP. This problem which lacks both columns and rows is referred to as the short RMP (SRMP). These problems grow both horizontally and vertically through the iterations of the column generation algorithm, leading to simultaneous column-and-row generation. The main challenge in solving these problems is to price columns that induce new rows as the dual variables associated with these new rows are unknown. To solve these problems, Muter et al. (2013a) presented a generic simultaneous column-and-row generation algorithm whose novelty lies in a PSP, namely the row-generating PSP, that cor-

rectly calculates the reduced costs of the variables under the absence of some of the linking constraints. Besides the TSCS problem, some examples of problems with column-dependent-rows include P-median facility location (Avella et al. 2007), multi-commodity capacitated network design (Katayama et al. 2009, Frangioni and Gendron 2009), two-stage batch scheduling (Wang and Tang 2010), robust crew pairing (Muter et al. 2013b) and time-constrained routing (Avella et al. 2006, Muter et al. 2012).

In TSCS, the linking constraints defined for column-dependent-rows problems tie the two stages of the cutting process by enforcing the availability of intermediate rolls cut in the first stage for the production of finished rolls in the second stage. In applying simultaneous column-and-row generation to the LP relaxation of the TSCS problem, the SRMP excludes some of the linking constraints corresponding to the intermediate rolls that have not been cut from any stock roll as part of a first stage cutting pattern. When a new column containing currently missing intermediate rolls is generated by the row-generating PSP, it also introduces new constraints to the SRMP. Zak (2002a,b) has developed a heuristic simultaneous column-and-row generation algorithm in which the number of new intermediate rolls that are added to the SRMP after the solution of the row-generating PSP is restricted to one. Such a restriction discards first stage cutting patterns that include more than one new intermediate roll which may possibly be part of an optimal solution. On the other hand, the application of the generic simultaneous column-and-row generation algorithm to the TSCS problem presented in Muter et al. (2013a) led to a large-scale integer programming formulation of the row-generating PSP, which does not limit the number of new intermediate rolls, unlike the heuristic approach proposed in Zak (2002a,b). However, no exact algorithm has been proposed to solve this large-scale PSP formulation so far.

In this paper, we solve the pattern-based formulation of the TSCS problem by an exact simultaneous column-and-row generation algorithm. The vital element of this algorithm is the row-generating PSP, which is a knapsack problem that generates a set of columns and rows simultaneously. We propose three exact algorithms to solve this PSP, which involve a combination of different strategies. The first one, an enumeration-based algorithm referred to as the two-phase approach, generates variables that may be part of an optimal solution in the first phase and then solves the resulting unbounded knapsack problem in the second. In the second and the third proposed approaches, the row-generating PSP is tackled by branch-and-price, which leads to nested column generation for the solution of the TSCS problem. We embed column generation within a well-known branch-and-bound based knapsack algorithm, and the PSP of this procedure becomes a knapsack problem with extra constraints. While the second approach generates the variables of the row-generating PSP one-by-one through the iterations of the column generation algorithm, the third approach generates the variables through a partial enumeration algorithm that is reminiscent of the first phase of the two-phase method. We test the performance of the proposed approaches by conducting computational experiments. Therefore, the contributions of this paper are

- an exact simultaneous column-and-row generation algorithm to solve the LP relaxation of the TSCS problem,
- three exact methods developed for the row-generating PSP, a large-scale knapsack problem, two of which are based on a novel column generation implementation embedded in

a branch-and-bound based knapsack algorithm,

- comprehensive computational experiments to evaluate the performance of the proposed methodologies, which demonstrates the superior performance of the combination of partial enumeration and column generation for the solution of the row-generating PSP.

In the computational experiments, we have also observed that the upper bound obtained by solving the SRMP using a mixed integer programming (MIP) solver coincides with the optimal objective value of the original problem in most of the instances.

This paper is organized as follows: in Section 2, the mathematical model of the TSCS problem and the overview of the simultaneous column-and-row generation algorithm to solve its LP relaxation are explained. The novel component of this algorithm, which is the row-generating PSP, is analyzed, and three exact methods are presented in Section 3. The results obtained from the computational experiments are reported in Section 4 which is followed by the conclusions in Section 5.

2. Simultaneous Column-and-Row Generation

In this section, we first give the pattern-based formulation of the TSCS problem and then present an overview of the simultaneous column-and-row generation algorithm to solve this formulation. For the generation of the cutting patterns, a PSP will be constructed for each variable set. The details of the row-generating PSP and the proposed algorithms for its solution, which stand as the major contribution of this paper, are deferred to the next section.

Mathematical Model

In the TSCS, a set of finished rolls, indexed by $m \in M$, defined by their width α_m and demand b_m, are cut from identical stock rolls, with widths of W, in two stages. In the first stage, stock rolls are cut into intermediate rolls, indexed by $i \in I$, through first stage cutting patterns, $k \in K$. The widths of intermediate rolls are not known in advance, but have to be in a specified interval, $[s^{min}, s^{max}]$. The intermediate rolls produced in the first stage are cut into finished rolls in the second stage through second stage cutting patterns, $n \in N$, to fulfill demand. The number of the first stage cutting pattern $k \in K$ and the second stage cutting pattern $n \in N$ used in the solution are represented by integer variables y_k and x_n, respectively. The mathematical model of this TSCS problem presented by Zak (2002a,b), which is referred to as the master problem (MP) in this paper, is

$$\text{(MP)} \qquad \text{minimize} \quad \sum_{k \in K} y_k, \tag{1}$$

$$\text{subject to} \quad \sum_{n \in N} B_{mn} x_n \geq b_m, \qquad m \in M, \tag{2}$$

$$\sum_{k \in K} C_{ik} y_k + \sum_{n \in N} D_{in} x_n \geq 0, \qquad i \in I, \tag{3}$$

$$y_k \geq 0, \text{ integer}, \quad k \in K, \tag{4}$$

$$x_n \geq 0, \text{ integer}, \quad n \in N. \tag{5}$$

In this formulation, the columns of the matrices C and B correspond to the first and the second stage cutting patterns, respectively. While constraints (2) impose that the demand on finished rolls is satisfied, the constraint set (3), referred to as the linking constraints, ensures that the number of second stage patterns cut from any intermediate roll $i \in I$, is smaller than or equal to the number of intermediate roll i being cut from stock rolls through first stage cutting patterns. Each second stage cutting pattern, $n \in N$, is linked to exactly one intermediate roll, $i \in I$, through a single non-zero entry of $D_{in} = -1$ in each column n of the matrix D. The objective function given in (1) minimizes the total number of stock rolls cut through the first stage cutting patterns.

The enumeration of set I is not possible as any roll having width in $[s^{min}, s^{max}]$ is a feasible intermediate roll. Neither is it essential in the solution of this problem since an optimal integer solution of (1)-(5) requires only a finite number of intermediate rolls and hence, a finite set of linking constraints of (3). In fact, it was shown in Zak (2002a,b) that an intermediate roll, which is characterized by its width, can exist in the optimal solution of this problem only if it corresponds to a second stage cutting pattern. Therefore, I will henceforth denote this finite subset of intermediate rolls. However, not only the generation of I may be cumbersome, but also the column generation method under the inclusion of all these rolls may perform poorly due to the large size of the subproblems. In this paper, our aim is to propose a simultaneous column-and-row generation algorithm to solve the LP relaxation of (1)-(5) to optimality without the complete enumeration of I. Hence, in the application of column generation, replacing K and N by $\bar{K}$ and $\bar{N}$, respectively, induces only a subset of the constraints corresponding to the intermediate rolls in which the existing columns reside. The resulting problem, which is referred to as the SRMP, is written as

$$\text{(SRMP)} \qquad \text{minimize} \quad \sum_{k \in \bar{K}} y_k, \tag{6}$$

$$\text{subject to} \quad \sum_{n \in \bar{N}} B_{mn} x_n \geq b_m, \quad m \in M, \tag{7}$$

$$\sum_{k \in \bar{K}} C_{ik} y_k + \sum_{n \in \bar{N}} D_{in} x_n \geq 0, \quad i \in \bar{I}, \tag{8}$$

$$y_k \geq 0, \quad k \in \bar{K}, \tag{9}$$

$$x_n \geq 0, \quad n \in \bar{N}. \tag{10}$$

where $\bar{I} \subset I$ refers to the set of existing intermediate rolls and the associated linking constraints in the SRMP that are induced by $\bar{K}$ and $\bar{N}$. Thus, generation of a first stage cutting pattern, which includes a set of intermediate rolls currently absent from the SRMP, adds new linking constraints corresponding to these intermediate rolls to the model. Consequently, the SRMP grows both column-wise and row-wise through the iterations of column generation, leading to simultaneous column-and-row generation. The main difficulty in solving these problems is to price columns that induce new rows as the dual variables associated with these new rows are unknown.

Before proceeding further in the explanation of the proposed algorithm to solve (1)-(5), it is essential to mention an alternative model that does not require simultaneous column-and-row

generation. Rather than utilizing two types of cutting patterns, a new matrix whose columns correspond to a set of cutting patterns, each consisting of finishes rolls cut from a stock roll, is introduced. This matrix can replace those in (2)-(5) if each column with entries representing number of finished rolls can be decomposed into second stage cutting patterns that form a set of feasible intermediate rolls. The resulting model does not have the linking constraints (3) and hence, can be solved by column generation without the simultaneous generation of rows. However, this model does not keep track of the intermediate rolls and associated second stage cutting patterns, and its solution mandates invoking the computationally burdensome row-generating PSP at each iteration, which generates a first stage cutting pattern that is composed of second stage cutting patterns.

Outline of Simultaneous Column-and-Row Generation

Finding the optimal solution for the LP relaxation of the MP using column generation amounts to ensuring that the respective values of the dual variables do not violate dual constraints. Letting v_m, $m \in M$ and w_i, $i \in I$ denote the dual variables associated with the sets of constraints (2) and (3), respectively, the dual of the MP is

$$\begin{aligned}
\text{(DMP)} \qquad & \text{maximize} && \sum_{m\in M} b_m v_m, && && (11)\\
& \text{subject to} && \sum_{i\in I} C_{ik} w_i \le 1, && k \in K, && (12)\\
& && \sum_{m\in M} B_{mn} v_m + \sum_{i\in I} D_{in} w_i \le 0, && n \in N, && (13)\\
& && v_m \ge 0, && m \in M, && (14)\\
& && w_i \ge 0, && i \in I. && (15)
\end{aligned}$$

If I, the set of intermediate rolls that can appear in the optimal solution, is given in advance, subproblems can be defined so as to generate only first stage and second stage cutting patterns with negative reduced cost. To check whether the optimal dual variable values v_m, $m \in M$ and w_i, $i \in I$ retrieved from the solution of the SRMP satisfy dual constraint sets (12) and (13), we identify two subproblems that generate the minimum reduced cost $y-$ and $x-$variables, respectively. The PSP associated with (12), referred to as $y-$PSP, that searches for a negative reduced cost first stage cutting pattern is

$$\begin{aligned}
\text{($y-$PSP)} \qquad \zeta_y = \text{maximize} \quad & \sum_{i\in I} w_i C_i, && (16)\\
\text{subject to} \quad & \sum_{i\in I} \gamma_i C_i \le W, && (16)\\
& C_i \in \mathbb{Z}^+ \cup \{0\}, \quad i \in I, && (16)
\end{aligned}$$

where γ_i is the width of intermediate roll $i \in I$, and C_i is the number of times this roll is cut in the first stage cutting pattern. It is an unbounded knapsack problem that may be solved efficiently by well-known methods from the literature (See Martello and Toth (1990) for the knapsack algorithms). If $\zeta_y > 1$, the resulting first stage cutting pattern, $k \in K \backslash \bar{K}$, with negative reduced cost is added to $\bar{K}$.

Recall that a single non-zero entry $D_{in} = -1$ in column n of D indicates that the second stage cutting pattern n is cut from intermediate roll i. Then, the dual constraint (13) for a second stage cutting pattern, $n \in N$, cut from $i \in I$ reduces to $\sum_{m \in M} B_{mn} v_m \leq w_i$. Hence, the following PSP, referred to as $x-$PSP, that seeks a negative reduced cost second stage cutting pattern is solved for each $i \in I$:

$$\begin{aligned} (x\text{−PSP}) \qquad \zeta_x^i = \text{maximize} \quad & \sum_{m \in M} v_m B_m, \\ \text{subject to} \quad & \sum_{m \in M} \alpha_m B_m \leq \gamma_i - e^{min}, \\ & B_m \in \mathbb{Z}^+ \cup \{0\}, \quad m \in M. \end{aligned} \tag{17}$$

where α_m is the width of $m \in M$, B_m is the number of times m is cut from i, and e^{min} is the mandatory minimum edge. The constraint imposes that the total width of the second stage cutting pattern merged with the minimum edge, e^{min}, does not exceed the width of i. Like $y-$PSP, the $x-$PSP is an unbounded knapsack problem. If $\zeta_x^i > w_i$, the resulting second stage cutting pattern cut from i with negative reduced cost is added to $\bar{N}$.

Consequently, for the given set of intermediate rolls I that can be cut in the optimal solution of (MP), the conventional column generation solves $y-$PSP and $x-$PSP repeatedly until neither of both generates a negative reduced cost column, in which case the optimal LP solution is found. On the other hand, for a given subset of intermediate rolls $\bar{I}$, the definitions of $y-$PSP and $x-$PSP are confined to the currently existing intermediate rolls in the SRMP, which produce the minimum reduced cost first stage cutting pattern containing only intermediate rolls in $\bar{I}$ and second stage cutting pattern cut from $\bar{I}$. Hence, the SRMP grows only column-wise when new negative reduced cost columns are generated by these subproblems. However, there may exist columns corresponding to the first stage cutting patterns which would have negative reduced cost under the existence of linking constraints currently missing from the SRMP. These linking constraints are associated with intermediate rolls that have not yet been generated. The generation of $y-$variables, which also introduce a set of new intermediate rolls in $I \backslash \bar{I}$ and enable new $x-$variables associated with the second stage cutting patterns cut from these intermediate rolls, is accomplished by the row-generating PSP. This PSP can be formulated as the $y-$PSP with the complete set of intermediate rolls I, even though $I \backslash \bar{I}$ are currently missing from the SRMP. We tackle this formulation in Section 3.

The overview of the simultaneous column-and-row generation algorithm, as proposed in Muter et al. (2013a), is given in Figure 1. In order to form the SRMP, we initialize the subsets of I, B, and C by using a simple procedure. Small sets of intermediate rolls and second stage cutting patterns are generated for each $m \in M$ by setting $B_{mm} = \lfloor s^{max}/\alpha_m \rfloor$ and $B_{m'm} = 0$ for $m' \neq m$ as long as $\lfloor s^{max}/\alpha_m \rfloor \alpha_m \geq s^{min}$. Using the initial set of intermediate rolls, we initialize C in a similar manner by cutting as many intermediate rolls of each type as possible from the stock roll. After the initialization of the SRMP with these sets, the $y-$PSP is invoked, whose objective is to identify a negative reduced cost first stage cutting pattern composed of intermediate rolls present in the current SRMP, $\bar{I}$. Therefore, the number of rows in the SRMP does not change with this subproblem. If $\zeta_y > 1$, a new first stage cutting pattern with a negative reduced cost, say k, is added to $\bar{K}$, and this PSP is called again with

new values of w_i, $i \in \bar{I}$ obtained from the solution of the SRMP. Otherwise, the algorithm moves on to the $x-$PSP, which aims to generate the minimum reduced cost second stage cutting pattern cut from one of the existing intermediate rolls $i \in \bar{I}$. If the sum of dual values of the finished rolls cut from $i \in \bar{I}$ is larger than the dual value associated with i, i.e., $\zeta_x^i > w_i$, a new second stage cutting pattern with a negative reduced cost, say n, is added to $\bar{N}$, and the algorithm returns to the $x-$PSP after retrieving the values of the dual variables from the optimal solution of the SRMP. Otherwise, the $y-$PSP is called if at least one $x-$variable has been added to the SRMP through the iterations since the last call of the $y-$PSP. If consecutive calls to $y-$ and $x-$PSP do not result in a negative reduced cost column, we move on to the row-generating PSP. If this PSP generates a first stage cutting pattern with negative reduced cost, it must contain intermediate rolls in $I \backslash \bar{I}$, and the SRMP grows both column-wise and row-wise. Otherwise, the solution of the SRMP is optimal for the LP relaxation of (1)-(5).

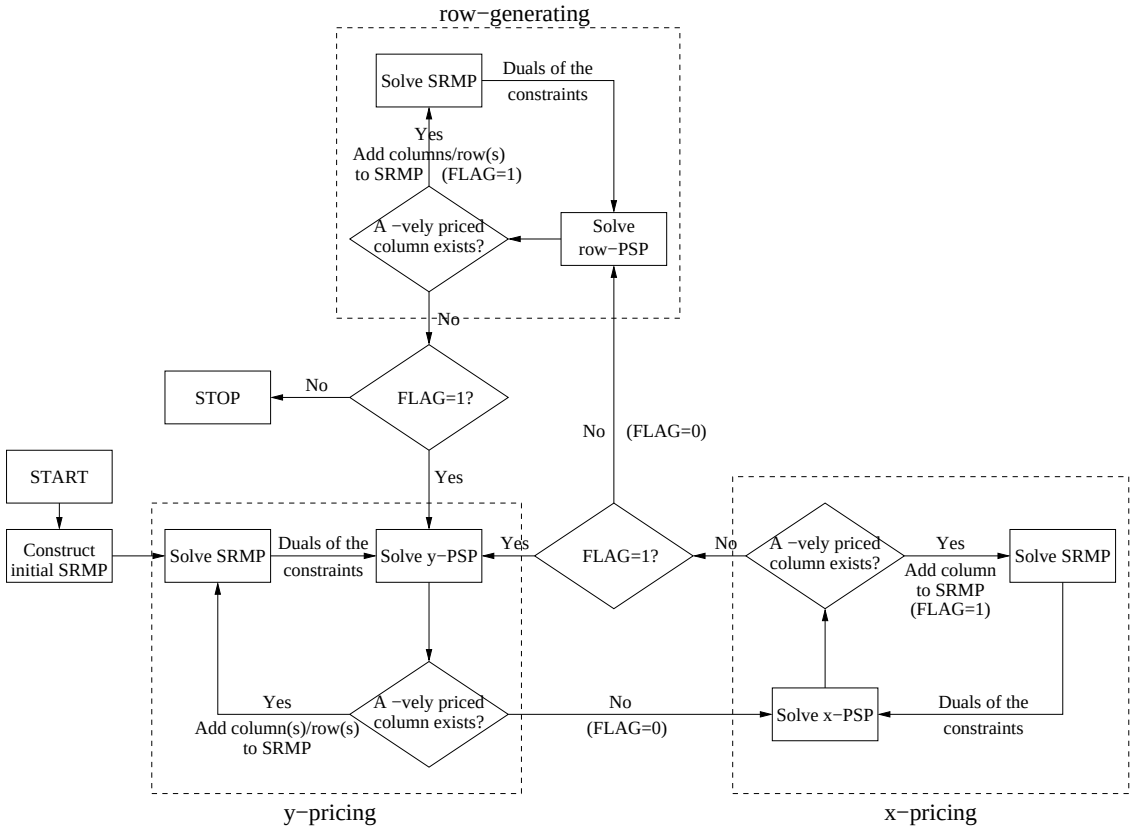

Figure 1: Simultaneous column-and-row-generation algorithm (Muter et al. (2013a)).

3. Row-Generating PSP

The row-generating PSP that was briefly explained previously is elaborated in this section. We first construct a large-scale mathematical model for this PSP using the theory existing in the literature, which allows us to prove the optimality of the simultaneous column-and-row

generation algorithm given in Figure 1. Then, we present three methods to solve this PSP, whose performances are compared in Section 4.

The hurdles related with the solution of the row-generating PSP, defined in (16) as a knapsack problem, are twofold: First, the widths of the currently missing intermediate rolls, γ_i for $i \in I\backslash\bar{I}$, are unknown, except for their allowable interval $[s^{min}, s^{max}]$, and many of these rolls may need to be generated simultaneously in the solution of the row-generating PSP. Second, the dual variables associated with the currently missing intermediate rolls, w_i for $i \in I\backslash\bar{I}$, are not known except that in a feasible dual solution they are non-negative, as imposed in (15). These dual variables must be estimated to be factored in the solution of the row-generating PSP that generates the intermediate rolls. The reformulation of (16) with these unknown components is as follows:

$$\zeta_{xy} = \text{maximize} \quad \sum_{i\in\bar{I}} w_i C_i + \sum_{i\in I\backslash\bar{I}} w_i C_i, \tag{18}$$

$$\text{subject to} \quad \sum_{i\in\bar{I}} \gamma_i C_i + \sum_{i\in I\backslash\bar{I}} \gamma_i C_i \leq W, \tag{19}$$

$$s^{min} \leq \gamma_i \leq s^{max}, \quad i \in I\backslash\bar{I}, \tag{20}$$

$$w_i \geq 0, \quad i \in I\backslash\bar{I}, \tag{21}$$

$$C_i \in \mathbb{Z}^+ \cup \{0\}, \quad i \in I, \tag{22}$$

which is similar to the $y-$PSP except for the second terms in (18) and (19) associated with the unknown intermediate rolls $I\backslash\bar{I}$ and the bounding constraints (20) and (21). This is a formidable nonlinear integer programming problem. By intuition, an intermediate roll is cut from a stock roll only if there exists a second stage cutting pattern which forges this intermediate roll. The implications of this conception, which have been proved respectively in Zak (2002a,b) and Muter et al. (2013a), are that

- the width of a new intermediate roll $i \in I\backslash\bar{I}$ is equal to the sum of widths of finished rolls cut from it plus the minimum edge, i.e.

$$\gamma_i = \sum_{m\in M} \alpha_m B_m + e^{min}, \tag{23}$$

- the dual value of a new intermediate roll $i \in I\backslash\bar{I}$ is equal to the sum of dual values of finished rolls cut from it, i.e.

$$w_i = \sum_{m\in M} v_m B_m, \tag{24}$$

  which is shown as part of the convergence proof of the proposed methodology.

Plugging in (23) and (24), we reach the following mathematical model of the row-generating PSP:

$$\zeta_{xy} = \text{maximize} \quad \sum_{i\in\bar{I}} w_i C_i + \sum_{i\in I\backslash\bar{I}} (\sum_{m\in M} v_m B_m^i) C_i, \tag{25}$$

$$\text{subject to} \quad \sum_{i\in\bar{I}} \gamma_i C_i + \sum_{i\in I\backslash\bar{I}} (\sum_{m\in M} \alpha_m B_m^i + e^{min}) C_i \leq W, \tag{26}$$

$$s^{min} \leq \sum_{m\in M} \alpha_m B_m^i + e^{min} \leq s^{max}, \quad i \in I\backslash\bar{I} \tag{27}$$

$$C_i \in \mathbb{Z}^+ \cup \{0\}, \quad i \in I, \tag{28}$$

$$B_m^i \in \mathbb{Z}^+ \cup \{0\}, \quad i \in I\backslash\bar{I}, \tag{29}$$

where B_m^i is the number of times finished roll m is cut in the new intermediate roll i. This model yields a first stage cutting pattern containing a set of new intermediate rolls, each of which introduce a linking constraint and a second stage cutting pattern. Note that due to (25) being a maximization objective, a second stage cutting pattern n formed by the solution of this problem yields the maximum value of $w_i = \sum_{m\in M} v_m B_m^i$ for a given i with width $\gamma_i = \sum_{m\in M} \alpha_m B_m^i + e^{min}$. This model is still intractable due to the non-linearity induced by terms associated with the unknown intermediate rolls $i \in I\backslash\bar{I}$, which also features exponentially many columns and constraints.

In order to alleviate these difficulties, Zak (2002a) restricts the number of new intermediate rolls to be added to the SRMP to one at each call of the row-generating PSP. For this new intermediate roll $i \in I\backslash\bar{I}$, (25)-(29) is solved for each value of C_i satisfying $0 \leq C_i \leq \frac{W-e^{min}}{s^{min}}$, and the solution that attains the largest objective function value is selected. This heuristic approach is shown to perform successfully in reaching the LP optimality for large instances, as observed in the computational experiments conducted in Zak (2002a) and those explained in Section 4.

Finally, we give the convergence proof of the proposed methodology given in Figure 1. This proof is a special version of that presented for general column-dependent-rows problems in Muter et al. (2013a). To that end, we show that if the solution of (25)-(29) attains $\zeta_{xy} \leq 1$, the algorithm stops at the optimal solution of the LP relaxation of the MP. Let SRMP($\bar{K}$, $\bar{N}$, $\bar{I}$) denote the current SRMP and $\mathbf{B}$ the optimal basis, a δ x δ matrix, of this problem. The solution of the row-generating subproblem is a first stage cutting pattern k that triggers the generation of a set of intermediate rolls indexed by $i \in \Delta_k$ and second stage cutting patterns indexed by $n \in S_N(k)$ each of which is tied to a single i with $D_{in} = -1$ and maximizes $\sum_{m\in M} v_m B_m^i$. Therefore, after y_k is added, the SRMP is augmented as SRMP($\bar{K}\cup k$, $\bar{N}\cup S_N(k)$, $\bar{I}\cup\Delta_k$). In the proof of the following theorem, we first demonstrate that the dual values prescribed for $i \in \Delta_k$ in (24) indeed result from an optimal basis for SRMP($\bar{K}$, $\bar{N} \cup S_N(k)$, $\bar{I} \cup \Delta_k$), which are then used to calculate the reduced cost of y_k correctly.

THEOREM 3.1 *If $\zeta_{xy} \leq 1$ after solving the row-generating PSP, the simultaneous column-and-row generation method terminates at an LP optimal solution of* (1)-(5).

PROOF. When the row-generating PSP is invoked, it is already guaranteed by $\zeta_y \leq 1$ and $\zeta_x^i \leq w_i$, $i \in \bar{I}$ after consecutive calls to $y-$PSP and $x-$PSP, respectively, that no $k \in K\backslash\bar{K}$ has a negative reduced cost using only $i \in \bar{I}$ and no $n \in N\backslash\bar{N}$ with a negative reduced cost can be cut from $i \in \bar{I}$. Those $k \in K\backslash\bar{K}$ that yield new intermediate rolls $i \in I\backslash\bar{I}$ and associated linking constraints can be priced with the dual values induced by an optimal

basis of SRMP($\bar{K}$, $\bar{N} \cup S_N(k)$, $\bar{I} \cup \Delta_k$). Therefore, new basic variables must be selected to account for the new linking constraints Δ_k in the augmentation of the optimal basis $\mathbf{B}$ with the addition of y_k, which constitutes the δ_kxδ_k matrix $\mathbf{B}_k$ where $\delta_k = \delta + |\Delta_k|$. There are two options for each $i \in \Delta_k$: the surplus variable associated with the new linking constraint or x_n, $n \in S_N(k)$ for which $D_{in} = -1$. The former renders $w_i = 0$, $i \in \Delta_k$ by complementary slackness, which leads to $\sum_{m \in M} B_{mn} v_m > w_i$ for a second stage cutting pattern n associated with i as $B_{mn} \neq 0$ for some $m \in M$ and $v_m \geq 0$, $m \in M$. Hence, the former violates some of the new dual constraints associated with x_n, $n \in S_N(k)$ and cannot form an optimal basis $\mathbf{B}_k$. Therefore, opting for the latter, we form the resulting augmented basis $\mathbf{B}_k$ and its inverse as

$$\mathbf{B}_k^{-1} = \begin{pmatrix} \mathbf{B} & \mathbf{F} \\ \mathbf{0} & -\mathbf{I} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{B}^{-1} & \mathbf{B}^{-1}\mathbf{F} \\ \mathbf{0} & -\mathbf{I} \end{pmatrix},$$

where the δx$|\Delta_k|$ matrix $\mathbf{F} = \begin{pmatrix} \mathbf{B_2} & \mathbf{0} \end{pmatrix}^T$ contains the coefficients of $n \in S_N(k)$ in the currently existing rows of SRMP, namely B_{mn}, $m \in M$ and $D_{in} = 0$, $i \in \bar{I}$, respectively. The $|\Delta_k|$x$|\Delta_k|$ submatrix $-\mathbf{I}$ is comprised of $D_{in} = -1$ for $i \in \Delta_k$ and $n \in S_N(k)$. Finally, the $|\Delta_k|$xδ submatrix $\mathbf{0}$ contains the entries of the basic variables in the constraint set Δ_k. The construction of this augmented basis with $S_N(k)$ is based on the fact that variables forming $\mathbf{B}$ do not exist in the new linking constraints Δ_k, and newly generated x_n, $n \in S_N(k)$ do not exist in the currently existing linking constraint $\bar{I}$. The values of the dual variables induced by $\mathbf{B}_k$ become

$$\begin{pmatrix} \mathbf{c_B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{B}^{-1} & \mathbf{B}^{-1}\mathbf{F} \\ \mathbf{0} & -\mathbf{I} \end{pmatrix} = \begin{pmatrix} \mathbf{c_B}\mathbf{B}^{-1} & \mathbf{c_B}\mathbf{B}^{-1}\mathbf{F} \end{pmatrix}, \tag{30}$$

where $\begin{pmatrix} \mathbf{c_B} & \mathbf{0} \end{pmatrix}$ is the vector of objective function coefficients of the basic variables forming $\mathbf{B}$ and x_n, $n \in S_N(k)$ which are selected to augment $\mathbf{B}$ to $\mathbf{B}_k$, respectively. As can be observed in (30), the values of the dual variables after the augmentation of the basis are preserved as $\mathbf{c_B}\mathbf{B}^{-1} = \begin{pmatrix} \mathbf{u} & \bar{\mathbf{w}} \end{pmatrix}$ where $\mathbf{u}$ and $\bar{\mathbf{w}}$ contain u_m, $m \in M$ and w_i, $i \in \bar{I}$. Moreover, we can show that the values of the dual variables w_i, $i \in \Delta_k$ are precisely those assigned by the row-generating PSP. The values assigned to w_i, $i \in \Delta_k$ in the row-generating PSP are calculated as

$$\mathbf{c_B}\mathbf{B}^{-1}\mathbf{F} = \begin{pmatrix} \mathbf{u} & \bar{\mathbf{w}} \end{pmatrix} \begin{pmatrix} \mathbf{B_2} & \mathbf{0} \end{pmatrix}^T = \mathbf{u}\mathbf{B_2}. \tag{31}$$

Therefore, the values of w_i, $i \in \Delta_k$ are as set in (24). Thus, the optimality of $\mathbf{B}_k$ for SRMP($\bar{K}$, $\bar{N} \cup S_N(k)$, $\bar{I} \cup \Delta_k$) can be argued as follows:

- Primal Feasibility: The linking constraints (3) indexed by $i \in \Delta_k$ in which no y_l, $l \in \bar{K}$ resides are satisfied since x_n, $n \in S_N(k)$ cannot take nonzero values as long as y_k is zero.
- Complementary Slackness: As shown above, after the augmentation of $\mathbf{B}$ to $\mathbf{B}_k$, the values of dual variables associated with the existing constraints do not change, and the newly selected basic variables x_n, $n \in S_N(k)$ prescribe the dual constraints (13) associated with them to be tight.
- Dual Feasibility: The reduced costs of the variables in SRMP($\bar{K}$, $\bar{N}$, $\bar{I}$) do not change

as these variables do not reside in Δ_k and the values of the existing dual variables are preserved after the augmentation of $\mathbf{B}$ to $\mathbf{B}_k$. Setting w_i, $i \in \Delta_k$ as prescribed in (25)-(29) ensures that the reduced costs of the slack variables associated with the new linking constraints are positive, $w_i > 0$, $i \in \Delta_k$ and that all x_n, $n \in N$ such that $D_{in} = -1$ have non-negative reduced costs.

The dual variable values induced by this optimal augmented basis $\mathbf{B}_k$ enables the correct calculation of the reduced cost of y_k, which is achieved by (25)-(29).

Using the same augmentation, an optimal basis can be formed for SRMP($\bar{K}$, N, I) by selecting a $x-$variable as basic for each $i \in I \backslash \bar{I}$. Therefore, when a call to (25)-(29) that considers the complete set of intermediate rolls, I, fails to yield a negative reduced cost $k \in K \backslash \bar{K}$, the simultaneous column-and-row generation algorithm stops with an LP optimal solution of (1)-(5). □

Outline of the Proposed Algorithms

In the rest of the section, we present three exact algorithms to solve the row-generating PSP. The first one, named two-phase method, generates the set of intermediate rolls that can be part of an optimal solution –which is a first stage cutting pattern– of the knapsack problem given in (16). This is accomplished by an implicit enumerative approach based on dynamic programming that generates the second stage cutting patterns forming a subset of the intermediate rolls. Given these rolls, (16) is solved to generate a first stage cutting pattern with the minimum reduced cost. The second algorithm tackles the reformulation (25)-(29), which contains large number of variables and constraints, by column generation. Therefore, instead of enumerating the intermediate rolls as in the first approach, the unknown intermediate rolls associated with variables C_i, $i \in I \backslash \bar{I}$ are generated via solving a PSP at a second level, leading to nested column generation. This column generation method to solve the row-generating PSP is integrated with a branch-and-bound algorithm designed specifically for the knapsack problem. The last approach is a combination of the first two. It devises the branch-and-price algorithm of the second method whereas the unknown intermediate rolls are generated by an enumerative scheme reminiscent of the first method.

Labeling Algorithm

Before we delve into the details of these proposed methodologies to solve the row-generating PSP, we point out that all of the methods are based on generation of the *best* second stage cutting pattern or a collection of second stage cutting patterns. The generation of intermediate rolls is directly related with that of feasible second stage cutting patterns through equations (23) and (24). For generation of cutting patterns, Valério de Carvalho (2002) developed an arc flow model that is based on shortest path formulation of the knapsack problem. However, we impose constraints, one of which is lower and upper limits $[s^{min}, s^{max}]$, on the generation of second stage cutting patterns, which prompted us to cast a shortest path problem with resource constraints. We apply labeling algorithms (Irnich and Desaulniers (2005)) that differ in the set of resources and the dominance relations used to eliminate unnecessary second stage cutting patterns. Below, we give the definition of the graph and the features common in all of the labeling algorithms employed in this paper. We defer the definition of dominance rules, which vary for each algorithm, later to the respective sections.

These labeling algorithms operate on graph $G = (V, A)$ where the set of nodes $V = M \cup \{s, t\}$ is comprised of the finished roll set augmented with a dummy source node $s = 0$ and sink node $t = |M| + 1$. Associated with each node $i \in V$, there is a width α_i and dual value v_i, and for s and t, both values are zero. $A = A_1 \cup A_2$ is the set of arcs which consists of $A_1 = \{(i, j) : i \in V, j \in V, i < j\}$ and $A_2 = \{(i, i) : i \in M\}$. The arcs in the second set are loops that allow inclusion of more than one finished roll in a second stage cutting pattern. The nodes in G are topologically sorted so that each node is treated only once in an ascending order. A path on this graph starts from s, and the arcs in A_1 can be traversed at most once and those in A_2 can be traversed as many times as the constraints allow. Let us define a label X_p^i for any partial path p on G starting from s and ending at some node $i \in V\backslash\{0\}$. We define R_C^{ip} and R_D^{ip} as the sum of dual values and the sum of widths of finished rolls in path p corresponding to a second stage cutting pattern, respectively, and B_m^p, $m \in M$ indicates the number of times node m is visited on path p. For path p, $R_C^{ip} = \sum_{m \in M} v_m B_m^p$ and $R_D^{ip} = \sum_{m \in M} \alpha_m B_m^p$. A label X_p^i corresponding to path p can be extended from node i to node $j \in V\backslash\{t\}$ for $j \geq i$ using arc (i, j) if $R_D^{ip} + \alpha_j \leq s^{max}$. In this case, a new label corresponding to a new path q, denoted by X_q^j, is formed with $R_D^{jq} = R_D^{ip} + \alpha_j$, $R_C^{jq} = R_C^{ip} + v_j$ and $B_m^{jq} = B_m^{ip}$ for $m \in M\backslash\{j\}$ and $B_j^{jq} = B_j^{iq} + 1$. On the other hand, when $j = t$, the feasibility condition is $R_D^{ip} \geq s^{min}$, which ensures that the completed second stage cutting pattern forms a feasible intermediate roll.

The labeling algorithm is initialized with a single label at node s, X_p^s, in which $R_C^{sp} = 0$, $R_D^{sp} = e^{min}$ and $B_m^{sp} = 0$, $m \in M$. When node $i \in M$ is selected to be treated at the ith iteration of the algorithm, the labels at this node are first extended with the loop arc (i, i). After no more feasible label can be created with the inclusion of the loop arc, all the labels on this node are extended to the reachable nodes $j > i$ where we apply the dominance rules to keep only the non-dominated labels.

3.1. Two-Phase Approach

The row-generating PSP can be formulated as an unbounded knapsack problem if we knew the entire set I. However, not all the intermediate rolls in I are essential in the solution of (16), and some of the rolls may be dominated by the others. These rolls can be fathomed from the problem without affecting the optimal solution. One basic dominance relation for unbounded knapsack is given in the following definition.

DEFINITION 3.1 *For a pair of rolls k and l, k dominates l if $w_k \geq w_l$ and $\gamma_k \leq \gamma_l$.*

In order to solve (16) without complete enumeration of I, we develop a two-phase algorithm that enumerates only the non-dominated intermediate rolls in the first phase, and solves the unbounded knapsack problem containing these rolls in the second phase. The labeling algorithm defined previously is devised in the first phase to find second stage cutting patterns which forge the intermediate rolls through (23) and (24). Since v_m, $m \in M$ that is part of (24) change after the solution of the SRMP, the first phase must be applied at each call of the row-generating subproblem. Such a two-phase algorithm has been applied to the two-stage cutting stock problem by Valério de Carvalho and Rodrigues (1995), in which the dominance rule given in Definition 3.1 is adapted for the knapsack subproblem, which includes only the homogeneous intermediate rolls. Other applications of the two-phase method can be seen in

the integrated location, scheduling and routing problems in Akca (2010) and in the multi-depot vehicle routing problem with inter-depot routes in Muter et al. (2014).

The only loose end in the labeling algorithm is the dominance step used to accelerate the labeling algorithm by eliminating the unnecessary labels. In defining the dominance rules, we need to take into account the feasibility conditions on intermediate rolls which set a lower and an upper bound on the width, namely s^{min} and s^{max}, respectively. Without a lower bound s^{min} on the intermediate roll widths, the dominance relation given in Definition 3.1 applied to a given pair of labels corresponding to paths p and q at node i would result in elimination of q if $R_C^{ip} \geq R_C^{iq}$ and $R_D^{ip} \leq R_D^{iq}$. However, with s^{min} imposed as a lower bound, this dominance relation applies only if both $R_D^{ip} \geq s^{min}$ and $R_D^{iq} \geq s^{min}$ or if $i = t$. On the other hand, if the total width of one or both of the paths are smaller than s^{min}, applying the above dominance relation would eliminate some of the essential non-dominated rolls. Paths p and q on some node i that satisfy $R_C^{ip} \geq R_C^{iq}$, $R_D^{ip} \leq R_D^{iq}$ and $R_D^{ip} < s^{min}$ are not comparable since not all extensions of these paths to t that is feasible for q, may constitute a feasible intermediate roll for p, due to the lower limit s^{min}. However, the dominance rule is still valid for these paths if their total widths are the same, i.e. $R_D^{ip} = R_D^{iq}$. Therefore, we use the following dominance relation:

DEFINITION 3.2 *At some node i, p dominates q if $R_C^{ip} \geq R_C^{iq}$ and one of the following holds:*

- $i \neq t$, $R_D^{ip} < s^{min}$ *and* $R_D^{ip} = R_D^{iq}$
- $i \neq t$, $R_D^{ip} \geq s^{min}$ *and* $R_D^{ip} \leq R_D^{iq}$
- $i = t$ *and* $R_D^{ip} \leq R_D^{iq}$

When all nodes $i \in M$ are treated, the labels at node t constitute the non-dominated intermediate roll set, denoted by $I^{'}$. The knapsack problem (16) composed of $I^{'}$ is then solved in the second phase of the two-phase approach. Before this unbounded knapsack problem is solved, we further eliminate some of the intermediate rolls using a well-known result of integer programming which was also employed in Muter et al. (2014) whose implication for our problem is: the intermediate rolls whose reduced costs are greater than (smaller than for a maximization problem) the difference between a lower and an upper bound can be fathomed (Nemhauser and Wolsey 1988, page 389). An easy-to-access upper bound of (16), which is an unbounded knapsack problem, is the objective function value of the LP relaxation that is readily available after putting the intermediate rolls in an nonincreasing order of (w_i/γ_i) for $i \in I^{'}$. Let the indices of intermediate rolls correspond to their position in this ordering. The optimal solution of the LP relaxation of (16), denoted by $\hat{C}$, is $\hat{C}_{i_1} = W/\gamma_{i_1}$, $\hat{C}_i = 0$, $i \in I^{'} \backslash \{i_1\}$. The objective function value attained by this solution, which is also an upper bound (UB) on ζ_{xy}, is $UB = W w_{i_1}/\gamma_{i_1}$. In the dual optimal solution, the dual value of the knapsack constraint (β) is $\beta = w_{i_1}/\gamma_{i_1}$. Hence, for this given dual optimal solution of (16), the reduced costs of variable C_i, $i \in I^{'}$ can be calculated as

$$\bar{c}_i = w_i - \beta\gamma_i = w_i - \frac{w_{i_1}}{\gamma_{i_1}}\gamma_i$$

The trivial lower bound (LB) on ζ_{xy} is one, $LB = 1$. The reason is that dual constraints (12) associated with $y-$variables is satisfied with equality for any basic first stage cutting pattern,

which attains $\zeta_{xy} = 1$. Therefore, any non-dominated intermediate roll $i \in I^{'}$ satisfying the following condition is eliminated.

$$\bar{c}_i < LB - UB = 1 - W\frac{w_{i_1}}{\gamma_{i_1}}$$

We will show in Section 4 that the first phase of the algorithm eliminates a large set of intermediate rolls, and the above property of integer programming helps to further eliminate a small number of intermediate rolls with a low computational effort. In the second phase, the unbounded knapsack problem is effectively solved by an existing algorithm, namely the MTU1 algorithm given in Martello and Toth (1990).

3.2. Branch-and-Price

In this section, we explain our proposed branch-and-price algorithm to solve the row-generating PSP. In the reformulation (25)-(29), the set of structural constraints that generates new intermediate rolls in $I\backslash\bar{I}$ is (27). Since the number of these rolls can be prohibitively large, we apply column generation to the LP relaxation of (25)-(29). To that end, we form a RMP using (25) and (26), which is initialized with C_i, $i \in \bar{I}$. Intermediate rolls in $I\backslash\bar{I}$ are then generated by solving a PSP that constructs the minimum reduced cost intermediate roll satisfying (27). We refer to this RMP as the *RMPSP*. In our nested column generation algorithm, the application of column generation to the row-generating PSP will be referred to as the second level column generation. The subproblem of the row-generating PSP, referred to as the *PSPSP*, can be written as

$$\begin{aligned} \text{(PSPSP)} \quad \tau_{2S} = \text{maximize} \quad & \sum_{m\in M} v_m B_m - \beta(\sum_{m\in M} \alpha_m B_m + e^{min}) \\ & = \sum_{m\in M} (v_m - \beta\alpha_m)B_m - \beta e^{min}, \\ \text{subject to} \quad & s^{min} \leq \sum_{m\in M} \alpha_m B_m + e^{min} \leq s^{max}, \\ & B_m \in \mathbb{Z}^+ \cup \{0\}, m \in M, \end{aligned} \tag{32}$$

where β denotes the dual variable corresponding to the knapsack constraint given in (26). This problem needs to be solved only once at each iteration of the second level column generation. since all intermediate rolls are characterized by the same width interval $[s^{min}, s^{max}]$. The PSPSP is an unbounded knapsack problem with a minimum filling constraint. Xu (2013) has introduced the binary version of the knapsack problem with a minimum filling constraint and have proposed an approximation algorithm for this problem. To solve this problem, we employ a modification of the labeling algorithm in which extra resources and dominance rules are defined. We first focus on the overall algorithm to solve the row-generating PSP, and then the solution of the PSPSP will be elaborated. If $v_m < \beta\alpha_m$ for all $m \in M$, τ_{2S} cannot be positive, which prompts the termination of the second level column generation. If the optimal solution $\hat{B}_m$, $m \in M$ attains $\tau_{2S} > 0$, an intermediate roll with $w = \sum_{m\in M} v_m\hat{B}_m$ and $\gamma = \sum_{m\in M} \alpha_m\hat{B}_m + e^{min}$ is added to $\bar{I}$, and the RMPSP is solved again. We point out that although $\tau_{2S} > 0$ guarantees that the ratio of this new roll is larger than β, i.e. $w/\gamma > \beta$, by

$$\sum_{m\in M} (v_m - \beta\alpha_m)B_m - \beta e^{min} = \quad w - \beta\gamma > 0,$$

Model (32) does not generate the intermediate roll with the largest ratio, which requires the solution of an integer linear fractional programming problem.

When (32) does not result in an intermediate roll with a positive reduced cost, the second level column generation terminates with the optimal solution to the LP relaxation of (25)-(29). If this solution is fractional, a branch-and-price algorithm which employs the second level column generation as the bounding procedure at each node of the branch-and-bound tree must be developed to find the optimal integer solution of the row-generating PSP. If $\zeta_{xy} > 1$ at the end of the branch-and-price algorithm, a first stage cutting pattern is added to the SRMP along with a set of linking constraints and $x-$variables. Otherwise, the column generation algorithm to solve the LP relaxation of the MP given in (1)-(5) terminates.

At the root node in the branch-and-price framework to solve (16), the solution of the RMPSP corresponds to the LP relaxation of the unbounded knapsack problem, denoted by $\hat{C}$, as explained previously. The primal and dual optimal solutions are $\hat{C}_{i_1} = w_{i_1}W/\gamma_{i_1}$, $\hat{C}_i = 0$, $i \in \bar{I}\backslash\{i_1\}$ and $\beta = w_{i_1}/\gamma_{i_1}$, respectively. Given the value of β, suppose that the optimal solution of the PSPSP, denoted by $\hat{B}$, renders $\tau_{2S} > 0$, and $i^* \in I\backslash\bar{I}$ is the newly generated intermediate roll satisfying $(w_{i^*}/\gamma_{i^*}) > \beta$ where $w_{i^*} = \sum_{m\in M} v_m\hat{B}_m$ and $\gamma_{i^*} = \sum_{m\in M} \alpha_m\hat{B}_m + e^{min}$. After i^* is added to the RMPSP, the new dual solution becomes $\beta = w_{i^*}/\gamma_{i^*}$. Suppose that the only nonzero variable in the solution, that is $\hat{C}_{i^*} = W/\gamma_{i^*}$, is fractional, and there exists no $i \in I$ satisfying $w_i/\gamma_i > \beta$. In this case, the conventional branching rule can be applied to generate the following two nodes: $C_{i*} \leq \lfloor W/\gamma_{i*}\rfloor$ and $C_{i*} \geq \lceil W/\gamma_{i*}\rceil$. At the node in which the former constraint is imposed, the optimal solution of the RMPSP becomes $\hat{C}_{i^*} = \lfloor W/\gamma_{i*}\rfloor$ and $\hat{C}_{i_1} = (W - \gamma_{i*}\lfloor W/\gamma_{i*}\rfloor)/\gamma_{i_1}$, and β is then equal to w_{i_1}/γ_{i_1}. Then, we check whether there exists $i \in I$ which satisfy $(w_i/\gamma_i) > \beta$ by solving the PSPSP. Previously generated intermediate roll i^*, whose reduced cost is nonpositive in the RMPSP, is generated again by the PSPSP with a positive reduced cost. This points to a crucial issue of integrating the dual variables associated with the branching constraints to the subproblem in the branch-and-price frameworks. Therefore, we have to add some constraints to the PSPSP to prevent the regeneration of intermediate roll i^*. Similar difficulties in branching were also faced by Vance (1998) in solving the conventional one-dimensional cutting stock problem. She transformed the PSP which is an unbounded knapsack problem to a 0-1 knapsack problem using the transformation given in Martello and Toth (1990) and solved it by the modified version of the algorithm developed in Horowitz and Sahni (1974) that handles the prevention constraints. Consequently, to solve the PSPSP given in (32) in the course of the branch-and-price algorithm, we need to develop a knapsack algorithm that avoids regeneration of existing intermediate rolls and also ensures that the width of the resulting intermediate roll is larger than or equal to s^{min}.

We observed in the preliminary experiments that the application of the branch-and-bound algorithm given above to the row-generating PSP is not efficient due to the explosion in the number of nodes. Moreover, as we go deeper in the branch-and-bound tree, the number of prevention constraints increases, which complicates the PSPSP. We circumvent these difficulties by integrating column generation within the lexicographic algorithm (Chvátal 1983, Gilmore and Gomory 1963), which is a branch-and-bound based algorithm designed for the unbounded knapsack problem. Since column generation has not been applied directly to the knapsack problem in the literature, we next explain how and when to call column generation within the

lexicographic algorithm to solve (25)-(29). The details of this algorithm, which is embedded column generation, is given in Algorithm 1, where z is the objective function value of the current solution and z^* is the best lower bound which is set to one initially. Only a subset of rolls $\bar{I}$ which is indexed in nonincreasing order of the ratios is available at the beginning of the row-generating PSP. We concatenate an artificial intermediate roll with $w = 0$ and $\gamma = s^{max}$ at the end of $\bar{I}$. The algorithm starts with the identification of the intermediate roll with the largest ratio that is achieved by a column generation procedure referred to as *ColGen* (Line 4 and Algorithm 2) so that an upper bound (UB) can be calculated using the ratio of this roll (Line 5). Whenever index i, which refers to an item at position i in Step 1 of the algorithm, is increased by one, the next roll in the list $\bar{I}$ is considered in the solution. Since the list of intermediate rolls is not complete, we check whether there exist any intermediate roll i^* between i and $i+1$ satisfying $w_{i^*}/\gamma_{i^*} > w_{i+1}/\gamma_{i+1}$. To that end, we call the ColGen procedure before the next item in the list is considered. This procedure solves the PSPSP with $\beta = w_{i+1}/\gamma_{i+1}$ in order to search for an intermediate roll whose ratio is larger than that of $i+1$. However, similar to the issue with that of branching mentioned previously, the PSPSP would result in an intermediate roll already existing in $\bar{I}$ with ratio larger than or equal to w_i/γ_i. Next lemma restricts the solution set of the PSPSP by ruling out the intermediate rolls whose ratios are larger than w_i/γ_i, which accelerates the solution of this problem.

LEMMA 3.1 *Suppose that ColGen is invoked with index i in Step 1 of Algorithm 1, and the PSPSP which is solved with $\beta = w_i/\gamma_i$ does not generate an intermediate roll with a positive reduced cost. Then, the PSPSP cannot generate an intermediate roll with a ratio larger than w_i/γ_i when solved with $\beta = w_{i+1}/\gamma_{i+1}$.*

PROOF. By contradiction. Suppose that when the PSPSP given in (32) is solved with $\beta = w_{i+1}/\gamma_{i+1}$, the resulting optimal solution denoted by $\hat{B}$ renders $\tau_{2S} > 0$, and the resulting intermediate roll denoted by i^* satisfies $w_{i^*}/\gamma_{i^*} > w_i/\gamma_i$. When the solution $\hat{B}$ associated with i^* defined above is plugged into the objective of the PSPSP with $\beta = w_i/\gamma_i \geq w_{i+1}/\gamma_{i+1}$, $\sum_{m \in M}(v_m - (w_i/\gamma_i)\alpha_m)\bar{B}_m - w_i/\gamma_i e^{min} > 0$, which contradicts the statement. □

Algorithm 1: An Algorithm to solve the Row-Generating PSP

Input: $z = 0$, $z^* = 1$, $j = 0$, $rest = W$, $sol_i = 0$, $i = 1, 2, ..., |\bar{I}|$;
Output: $bestsol$, z^* ;
$i = j + 1$;
ColGen ;
$UB = W \frac{w_1}{\gamma_1}$;
if $UB < z^*$ **then**
    Stop ;
Step 1 ;
while $rest \geq s^{min} \wedge i < |\bar{I}|$ **do**
    $sol_i = \lfloor rest/\gamma_i \rfloor$;
    $rest = rest - sol_i \gamma_i$;
    $z = z + sol_i w_i$;
    ColGen ;
    $i = i + 1$;
    $u = rest\, w_i/\gamma_i$;
    if $z^* \geq z + u$ **then**
        Go to Step 3 ;
    if $u = 0$ **then**
        Go to Step 2 ;
end
Step 2 ;
if $z^* < z$ **then**
    $bestsol = sol$;
    $z^* = z$;
if $z^* = UB$ **then**
    Stop ;
Go to Step 3 ;
Step 3 ;
$j = i - 1$;
while $j > 0 \wedge sol_j = 0$ **do**
    $j = j - 1$;
end
if $j = 0$ **then**
    Stop ;
$sol_j = sol_j - 1$;
$rest = rest + \gamma_j$;
$z = z - w_j$;
if $z^* \geq z + rest\, w_{j+1}/\gamma_{j+1}$ **then**
    Go to Step 3 ;
$j = i + 1$;
Go to Step 1 ;

Using the result of the above lemma, we can restrict the ratio of the intermediate roll generated by the PSPSP solved with $\beta = w_{i+1}/\gamma_{i+1}$ to be no larger than that of roll i by adding a constraint to the PSPSP as

$$\sum_{m \in M} (v_m - (w_i/\gamma_i)\alpha_m) B_m \leq (w_i/\gamma_i) e^{min}. \tag{33}$$

For the sake of clarity, we refer to the extension of the PSPSP which generates an intermediate roll satisfying (33) and which is not dominated by the rolls in $\bar{I}$ as the *PSPSP1*. If solving the PSPSP1 with $\beta = w_{i+1}/\gamma_{i+1}$ results in $\tau_{2S} > 0$, the generated intermediate roll, say i^*, is inserted into $\bar{I}$ at the $i+1$st position, and the PSPSP1 is solved again by setting $\beta = w_{i^*}/\gamma_{i^*}$ since this problem does not generate the intermediate roll with the largest ratio between i and $i+1$, as pointed out earlier. Otherwise, there does not exist any non-dominated roll between i and $i+1$, and the lexicographic algorithm continues by considering $(i+1)$st roll in the list. It is clear that solving the PSPSP1 with β which has already provided no new intermediate roll

would be futile. Hence, we call column generation only if w_{i+1}/γ_{i+1} is smaller than parameter Γ, and this parameter is set to β when the PSPSP1 does not generate a new intermediate roll. The outline of ColGen that is executed within Algorithm 1 is given in Algorithm 2.

Algorithm 2: ColGen

```
Input: i, Γ ;
Output: Γ ;
if w_{i+1}/γ_{i+1} < Γ then
    while true do
        β = w_{i+1}/γ_{i+1} ;
        Solve PSPSP1 ;
        if τ_{2S} > 0 then
            Insert {i*} into Ī at the i + 1st position ;
        else
            Γ = w_{i+1}/γ_{i+1} ;
            break out of while ;
    end
```

We model the PSPSP1 as a shortest path problem with resource constraints on graph G which was defined previously together with a labeling algorithm. This graph and the labeling algorithm will be modified accordingly to reflect the characteristics of the PSPSP1. First, we associate two more values $\bar{v}_m = v_m - \beta\alpha_m$ where $\beta = w_{i+1}/\gamma_{i+1}$ and $\hat{v}_m = v_m - \bar{\beta}\alpha_m$ where $\bar{\beta} = w_i/\gamma_i$ with each node $m \in M$, and again these values are zero for both s and t. The former parameter $\bar{v}_m$ is the objective function coefficient of $m \in M$ in the PSPSP1, and the latter parameter $\hat{v}_m$ is used to calculate the left-hand-side of the prevention constraint (33). In order to accelerate the enumeration process using a bounding procedure, we sort the nodes $m \in M$ in nonincreasing order of the ratios, $\bar{v}_m/\alpha_m$. In addition to previously defined resource R_D^{ip} associated with a label X_p^i, we employ two more information, R_O^{ip} and R_L^{ip}, to calculate the objective function value and the left-hand-side of (33), respectively, which are initialized as zero at node s. Observe that $\bar{v}_m$ and $\hat{v}_m$ can have nonpositive values for some $m \in M$. In the unbounded knapsack problem, items with a nonpositive objective function coefficient can be disregarded. However, in the PSPSP1, the finished rolls with $\bar{v}_m \leq 0$ must be considered due to the constraint imposing minimum total width s^{min} as well as constraint (33).

Since the PSPSP1 is an optimization problem, we aim at finding the second stage cutting pattern that maximizes the objective function. This pattern is a candidate to be inserted at the $(i+1)$st position only if $\sum_{m\in M} \bar{v}_m B_m > \beta e^{min}$. Hence, we initialize a parameter $\rho = \beta e^{min}$ which is updated when a second stage cutting pattern satisfying $\sum_{m\in M} \bar{v}_m B_m > \rho$ is found. A label X_p^i corresponding to path p can be extended from node i to node $j \in V\backslash\{t\}$ for $j \geq i$ using arc (i, j) if $R_D^{ip} + \alpha_j \leq s^{max}$ as imposed previously, and one of the following conditions is satisfied:

- $\bar{v}_j > 0$ and $R_O^{ip} + (s^{max} - R_D^{ip})\bar{v}_j/\alpha_m > \rho$
- $\bar{v}_j < 0$ and $R_O^{ip} > \rho$

In the above conditions, we check whether partial path p can attain the best objective function value in the PSPSP1 by calculating an upper bound which is the current value of R_O^{ip} if a roll with $\bar{v}_j < 0$ is reached. These upper bounds are valid since the finished rolls are sorted in nonincreasing order of the ratios. On the other hand, path p can be extended to $j = t$ under the following conditions:

- $R_D^{ip} \geq s^{min}$, i.e., it has an allowable width,
- $R_O^{ip} > \rho$, i.e., it attains the maximum objective function value,
- $R_L^{ip} \leq \bar{\beta} e^{min}$, i.e., its ratio is smaller than or equal to $\bar{\beta}$

Along the extension to $j \in V \backslash \{t\}$ from $i \leq j$, a new label corresponding to path q, denoted by X_q^j, is formed by setting $R_D^{jq} = R_D^{ip} + \alpha_j$, $R_O^{jq} = R_O^{ip} + \bar{v}_j$, $R_L^{jq} = R_L^{ip} + \hat{v}_m$ and $B_m^{jq} = B_m^{ip}$ for $m \in M \backslash \{j\}$ and $B_j^{jq} = B_j^{iq} + 1$.

In many calls to the PSPSP1 through the iterations of second level column generation, the optimal solution of this problem may be dominated by an intermediate roll in $\bar{I}$, which deteriorates the performance of Algorithm 1. Applying the dominance condition given in Definition 3.1 within the labeling algorithm developed for the PSPSP not only keeps the non-dominated rolls in $\bar{I}$ but also ensures that even if the optimal solution satisfies (33) with equality, the generated roll is not a duplicate of some $j \in \bar{I}$ with $w_j/\gamma_j = w_i/\gamma_i$. The dominance conditions given in Definition 3.2 are also applied here with the exception in the last condition that we compare path q at the terminal node t with each $i \in \bar{I}$ to ensure that this list contains only the non-dominated intermediate rolls. In Section 4, we observed in the computational experiments that this branch-and-price approach generates smaller number of intermediate rolls at the solution of the row-generating PSP than the other proposed methods.

3.3. Hybrid Algorithm

Unlike the two-phase method in which the non-dominated columns are generated in a single pass of the labeling algorithm, the branch-and-price algorithm given in Algorithm 1 fills the incomplete list $\bar{I}$ by calling the PSPSP1 through its course. The PSPSP1 is generally called large number of times, which has a detrimental effect on the performance of the overall lexicographic algorithm. Thus, in the hybrid approach, instead of generating the $i+1$st item when ColGen is called with index i in Step 1 of Algorithm 1, we enumerate the set of non-dominated items between rolls i and $i+1$. This enumeration-based approach to solve the row-generating PSP is similar to Algorithm 1 except that instead of the ColGen procedure, a modification of the first phase approach that we explained in Section 3.1 is applied. In this case, we aim to generate not only the non-dominated intermediate rolls but also the ones that induce a positive objective function value in the PSPSP1 with $\beta = w_{i+1}/\gamma_{i+1}$. In other words, we generate all possible solutions satisfying the constraints of the PSPSP1.

For the enumeration of the non-dominated rolls between a pair of adjacent rolls in $\bar{I}$ through the course of Algorithm 1, we integrate the algorithms developed for the PSPSP1 given in Section 3.2 with the first phase of the two-phase method. In this hybrid algorithm, graph G, in which node set V is sorted in nonincreasing order of the ratios, and the values associated with the node set are utilized as defined in the solution procedure of the PSPSP1. The extension conditions in the labeling algorithm designed for the PSPSP1 are adapted for the enumeration process. The parameter ρ that was used in the extension conditions to solve the optimization problem PSPSP1 is fixed to βe^{min} since R_O^{ip}, which records the objective function value of path p on i, is required to exceed βe^{min} at termination to enter the list $\bar{I}$. The dominance conditions defined in Definition 3.2 are also employed here to rule out the useless labels. Moreover, in order to ensure that $\bar{I}$ contains only the non-dominated

Table 1: Instance generators

Parameter	I1	I2
Stock size (W)	5000	10000
(s^{min}, s^{max})	(1200,1900)	(1250,2500)
Order width (α_m)	U(300,500)	U(300,500)
Order amount (b_m)	U(10,100)	U(10,100)

intermediate rolls, we apply the dominance conditions given in Definition 3.1 while adding the feasible rolls at t to $\bar{I}$ at the end of the enumeration. After the non-dominated rolls are inserted in $\bar{I}$, Γ is set to β. Following the lines of Algorithm 2, unless $w_{i+1}/\gamma_{i+1} < \Gamma$, the enumeration procedure is not invoked.

In the preliminary experiments, we have observed that the hybrid algorithm calls the enumeration scheme large number of times, and in most of them, the difference between w_i/γ_i and w_{i+1}/γ_{i+1} is very small. In order to boost the performance of this algorithm, unless $w_i/\gamma_i - w_j/\gamma_j > \epsilon$ where initially $j = i+1$, we increment j by one. Enlarging the gap between $\bar{\beta}$ and β increases the number of the intermediate rolls created by the enumeration procedure. However, the experiments reveal that the saving from the number of calls to the enumeration procedure compensates for this increase.

4. Computational Experiments

In this section, we conduct extensive computational tests to evaluate the performance of the proposed simultaneous column-and-row generation algorithm for the TSCS problem and to compare the three algorithms that were designed to solve the row-generating PSP. The computational experiments are run on a computer with 3.6 GHz Intel Xeon E5-1620 processor and 16 GB of RAM, and the algorithms are implemented in Visual C++ employing IBM ILOG CPLEX 12.6 and Concert Technology. Since the proposed methodology solves the LP relaxation of the TSCS problem, the duality gap is evaluated by solving the final SRMP as an integer program using the MIP solver of IBM ILOG CPLEX 12.6. A gap smaller than 1 indicates that an optimal integer solution has been reached. All results on times are reported in seconds.

We generate instances using a strategy similar to that devised in Zak (2002a). We select ten values for the cardinality $|M|$ of the finished roll set from 10 to 100 with increment of 10, and then fix W, s^{min}, s^{max} and e^{min} to integer values. While enlarging W allows a wider variety of first stage cutting patterns, the difference between s^{min} and s^{max} is the determinant on the cardinality of the intermediate roll set and also the set of second stage cutting patterns. The width and demand of finished rolls are generated randomly from integer uniform distribution. Table 1 shows two instance generators which are dubbed as "I1" and "I2" where the latter generates harder problems due to the larger cardinality of the intermediate roll and cutting pattern sets compared to those generated by I1. For each value selected for $|M|$, five replications are performed, and the results are reported in averages.

The average results of the proposed simultaneous column-and-row generation algorithm on I1 and I2 instances are reported in Tables 2 and 3, respectively. The results associated with the algorithms proposed for the row-generating PSP in Sections 3.1, 3.2 and 3.3 are

given under the columns "2-Phase", "BP" and "Hybrid", respectively. For each algorithm, the number of intermediate rolls at the end of the algorithm (# int. rolls), the solution time (Time), the average number of intermediate rolls generated in the calls to the row-generating PSP (Av. int. rolls) and the number of calls to the row-generating PSP (# calls) averaged over the five replications are reported. The instances for which the average time is larger than one hour are indicated by "-". Each table contains two set of rows, one for $e^{min} = 0$ and the other for $e^{min} = 50$. In the Hybrid algorithm, we have devised a sufficiently small parameter ϵ which affects the number of times the enumeration procedure is called. In the preliminary experiments, we observed the best performance of this algorithm when $\epsilon = 10^{-10}$ for $e^{min} = 0$ and $\epsilon = 10^{-5}$ for $e^{min} = 50$.

Tables 2 and 3 convey similar comparative results of the proposed algorithms. In terms of computational time, Hybrid outperforms BP and 2-Phase, and BP is by far the worst algorithm. In both tables, when e^{min} increases to 50, the performance of all algorithms, especially that of BP whose average solution time exceeds the time limit in many instances, deteriorates. This can be attributed to an evident increase in the number of calls to the row-generating PSP, which is the most time-consuming component of the overall algorithm. On the other hand, the average computational time of Hybrid and 2-Phase is within the time limit. Although BP generates the fewest number of intermediate rolls in the solution of the row-generating PSP, as indicated under the "Av. int. rolls" column, the inferior performance is due to the large number of calls to the PSPSP1. Being based on the complete enumeration of the non-dominated intermediate rolls, 2-phase generates the largest number of intermediate rolls in the solution of the row-generating PSP. The numbers of intermediate rolls in the SRMP and calls to the row-generating PSP reported under "int. rolls" and "# calls", respectively, are close for all algorithms. Regardless of the algorithm employed, the computation times and the numbers of intermediate rolls in the SRMP increase with few exceptions as $|M|$ increases. Since there is a declining trend in the number of calls to the row-generating PSP, the increase in the number of intermediate rolls in the SRMP together with $|M|$ can be partly attributed to the initialization procedure in which an intermediate roll is formed for each finished roll width.

Additionally, we solve the instance sets I1 and I2 by the heuristic simultaneous column-and-row generation algorithm proposed by Zak (2002a), which allows only a single intermediate roll to be generated by the row-generating PSP. We report "Time" and the optimality gap (%Gap) averaged over the five replications. The row-generating PSP of this algorithm is solved by CPLEX 12.6, which resulted in shorter average computation times than the original branch-and-bound algorithm employed by Zak (2002a). On the instance set I1, this algorithm achieves the optimal LP relaxation values except for a single instance with 10 finished rolls with a minor gap. The experiments on I2 instance set yield similar results, where the heuristic algorithm fails to reach the optimal LP bound in 4 instances with 10 finished rolls, three for $e^{min} = 0$ and one for $e^{min} = 50$, with maximum gap of 6.8%. These results are in line with those reported by Zak (2002a) in which large gaps are observed only in small instances. Despite being a heuristic, this algorithm does not provide a computational advantage over the exact algorithm proposed in this paper due to the fact that (25)-(29) is solved several times for possible values C_i of the new intermediate roll i. In fact, when $e^{min} = 50$, the average time for the I2 instances with 100 finished rolls is larger than the time limit.

Table 2: Comparison of the algorithms on the instances generated by I1

$e^{min}=0$	2-Phase				BP				Hybrid				Zak (2002a)	
$\|M\|$	# int. rolls	Time	Av. int. rolls	# calls	# int. rolls	Time	Av. int. rolls	# calls	# int. rolls	Time	Av. int. rolls	# calls	Time	% Gap
10	29.6	0.5	195.5	12.2	29.0	6.1	173.5	14.2	29.0	0.7	176.0	14.0	5.9	0.0
20	30.2	2.0	582.3	8.4	30.0	58.3	459.8	8.0	30.0	2.7	477.0	8.8	7.6	0.0
30	33.8	3.4	561.3	4.8	33.8	107.2	456.6	4.8	33.6	3.8	512.6	4.6	13.2	0.0
40	42.6	7.5	735.0	4.6	42.4	127.5	508.5	4.4	42.6	7.2	732.4	4.6	14.1	0.0
50	51.4	16.5	745.9	2.8	51.8	180.1	524.7	3.0	51.8	15.6	712.1	3.0	31.6	0.0
60	61.4	37.0	603.8	2.8	62.0	281.0	477.3	2.8	62.0	33.3	602.3	2.8	41.9	0.0
70	69.0	48.7	611.2	2.2	69.0	260.2	431.3	2.2	69.0	40.1	610.9	2.2	113.8	0.0
80	79.6	67.5	773.1	2.8	80.2	740.4	648.8	2.8	79.6	56.9	772.2	2.8	62.2	0.0
90	86.4	117.1	469.2	1.6	86.4	359.9	355.6	1.6	86.4	102.5	468.8	1.6	139.8	0.0
100	95.0	100.6	314.0	1.4	95.4	547.5	280.0	1.4	95.0	84.6	313.6	1.4	125.4	0.0
Average	57.9	40.1	559.1	4.4	58.0	266.8	431.6	4.5	57.9	34.8	537.8	4.6	55.6	0.0
$e^{min}=50$	2-Phase				BP				Hybrid				Zak (2002a)	
$\|M\|$	# int. rolls	Time	Av. int. rolls	# calls	# int. rolls	Time	Av. int. rolls	# calls	# int. rolls	Time	Av. int. rolls	# calls	Time	% Gap
10	42.6	1.0	182.3	19.6	39.6	11.0	172.8	17.2	37.6	1.1	173.9	16.4	35.8	<0.1
20	35.8	3.6	537.4	10.8	37.6	254.0	518.7	11.8	37.4	6.4	525.6	11.8	23.8	0.0
30	37.2	6.4	690.2	7.6	36.6	846.8	661.7	7.4	36.4	8.2	668.5	7.2	19.1	0.0
40	46.6	13.8	731.5	8.2	47.0	5094.8	695.6	8.2	46.6	17.0	697.7	7.8	22.9	0.0
50	53.8	22.0	742.0	5.0	-	-	-	-	53.4	23.7	699.7	4.6	30.1	0.0
60	63.4	47.0	746.5	4.6	-	-	-	-	63.6	48.5	699.7	4.6	48.8	0.0
70	70.2	57.0	743.1	2.8	-	-	-	-	69.6	52.7	700.6	2.8	70.6	0.0
80	80.4	82.6	756.4	3.2	-	-	-	-	80.6	76.9	700.6	3.0	79.0	0.0
90	85.8	145.0	769.7	1.4	-	-	-	-	86.0	109.0	701.0	1.4	163.0	0.0
100	96.2	140.3	777.4	2.2	-	-	-	-	96.8	135.8	700.6	2.6	147.7	0.0
Average	61.2	51.9	667.6	6.5					60.8	47.9	626.8	6.2	64.1	0.0

Table 3: Comparison of the algorithms on the instances generated by I2

$e^{min} = 0$	2-Phase				BP				Hybrid				Zak (2002a)	
$\|M\|$	# int. rolls	Time	Av. int. rolls	# calls	# int. rolls	Time	Av. int. rolls	# calls	# int. rolls	Time	Av. int. rolls	# calls	Time	% Gap
10	18.0	0.8	407.3	4.8	18.8	12.4	347.3	5.0	18.6	0.6	332.5	5.0	11.0	2.2
20	23.2	4.3	1087.1	3.0	23.4	198.9	832.9	3.0	23.4	2.8	907.2	3.0	13.9	0.0
30	31.4	9.1	1178.2	2.2	31.8	297.6	863.3	2.4	31.6	5.6	1023.3	2.4	13.4	0.0
40	41.4	26.0	1243.2	2.4	41.4	492.7	934.3	2.4	41.4	13.0	1108.1	2.4	16.4	0.0
50	49.4	47.7	1273.0	2.0	49.4	496.4	827.0	2.0	49.4	17.0	1055.2	2.0	16.3	0.0
60	58.6	81.9	1283.0	2.0	58.6	414.5	768.6	2.0	58.6	47.4	1168.6	2.0	28.0	0.0
70	68.4	143.8	1281.6	2.0	68.4	796.2	903.4	2.0	68.4	76.7	1154.0	2.0	54.5	0.0
80	77.0	199.1	1290.4	2.0	77.0	1459.5	982.6	2.0	77.0	76.3	1140.4	2.0	56.6	0.0
90	84.6	311.4	1296.4	2.0	84.6	1401.8	855.4	2.0	84.6	188.6	1218.8	2.0	94.2	0.0
100	93.2	394.5	1305.8	2.0	93.2	1850.2	967.4	2.0	93.2	182.9	1248.0	2.0	80.2	0.0
Average	54.5	121.9	1164.6	2.4	54.7	742.0	828.2	2.5	54.6	61.1	1035.6	2.5	38.5	0.2
$e^{min} = 50$	2-Phase				BP				Hybrid				Zak (2002a)	
$\|M\|$	# int. rolls	Time	Av. int. rolls	# calls	# int. rolls	Time	Av. int. rolls	# calls	# int. rolls	Time	Av. int. rolls	# calls	Time	% Gap
10	30.8	54.0	264.3	12.2	28.4	148.8	234.4	11.6	28.2	124.0	246.2	11.4	99.1	<0.1
20	32.0	94.8	825.6	11.6	31.8	2559.4	677.4	11.4	31.6	164.5	763.1	11.2	178.5	0.0
30	40.8	61.4	1197.5	12.0	-	-	-	-	39.8	66.0	1035.9	10.4	102.1	0.0
40	48.6	123.9	1265.0	9.6	-	-	-	-	48.8	108.0	1050.8	9.6	61.9	0.0
50	54.6	200.4	1266.3	7.0	-	-	-	-	53.4	178.1	1056.0	5.8	544.8	0.0
60	66.6	413.4	1271.0	10.2	-	-	-	-	65.0	320.4	1033.9	8.2	127.7	0.0
70	75.4	627.2	1277.0	8.2	-	-	-	-	73.6	495.8	1018.0	6.6	515.0	0.0
80	86.6	1063.5	1278.7	11.0	-	-	-	-	85.8	855.2	959.0	10.0	845.2	0.0
90	95.0	1699.5	1286.0	11.8	-	-	-	-	92.0	1224.0	974.9	9.4	2772.1	0.0
100	99.8	2554.9	1292.5	12.8	-	-	-	-	99.8	1635.8	999.0	8.2	-	-
Average	63.0	689.3	1122.4	10.6					61.8	517.2	913.7	9.1	582.9	0.0

Table 4: Evaluation of the integer solutions of TSCS

	I1			I2		
$\|M\|$	# Solved	# Opt.	Av. Diff.	# Solved	# Opt.	Av. Diff.
10	5	5	0	5	5	0
20	5	5	0	5	5	0
30	5	5	0	5	5	0
40	5	5	0	5	5	0
50	5	5	0	5	5	0
60	5	5	0	5	5	0
70	5	5	0	5	5	0
80	5	5	0	5	5	0
90	4	4	0	5	2	1
100	4	4	0	4	1	1

Lastly, in order to evaluate the distance between the optimal LP solution, denoted by Z^{LP}, and the integer optimal solution of TSCS, denoted by Z^*, we solve the SRMP as an integer program whose optimal objective function value is denoted by z^{MIP}. The SRMP that is solved by the MIP solver is the one obtained at termination of the Hybrid algorithm. The results are presented in Table 4 for the I1 and I2 instances. A one-hour time limit is imposed on the MIP solver, and the number of instances out of five solved within the time limit is given under "# Solved". If the distance between the LP optimal solution and the solution of the MIP solver is smaller than one, i.e. $Z^{MIP} - Z^{LP} < 1$, the latter is the proven optimal integer solution of the instance. The number of optimal solutions obtained out of five instances and the average distance $(Z^{MIP} - \lceil Z^{LP} \rceil)$ when integer optimality cannot be proven are reported under "# Opt." and "Av. Diff.", respectively. Until $|M| = 80$, all the MIPs are solved within the time limit, and the optimal integer solutions are reached. In the I1 instances, only one instance for both $|M| = 90$ and $|M| = 100$ could not be solved within the time limit while the others are solved to integer optimality. In the I2 instances, only one instance with 100 orders could not be solved. However, in six instances, $(Z^{MIP} - \lceil Z^{LP} \rceil) > 0$, and particularly, the difference is one in all of them.

5. Conclusions

In this paper, we have tackled the TSCS problem and presented an exact simultaneous column-and-row generation algorithm for the TSCS problem. The vital piece of this algorithm is the row-generating PSP, which is formulated as an unbounded knapsack problem containing items missing from the model. We have proposed three methods to solve this PSP. The first method is based on the intelligent enumeration of the intermediate rolls by devising a simple dominance relation defined for the knapsack problem. The other one is a branch-and-price algorithm that devises lexicographic method for the branch-and-bound and a novel procedure for column generation. The last algorithm for the row-generating PSP, which is also the best performing one, is a hybrid algorithm that integrates the first two. We have also observed that the upper bound obtained by solving the SRMP using a mixed integer programming (MIP) solver coincides with the optimal objective value of the original problem in most of the instances.

As a future research direction, we intend to design a branch-and-price algorithm for problems with column-dependent-rows which also encompass the TSCS problem. The features specific to this class of problems will necessitate novel branching decisions related with each

variable set. Additionally, in the literature, one prominent algorithm for problems with two-level decomposable structure is nested column generation, which performed poorly for the TSCS problem. Effective enumeration-based schemes, namely the two-phase and hybrid approaches, have been devised in this paper to solve the row-generating PSP. One line of research is generalizing and analyzing the approaches for problems with two-level decomposable structure and explaining when and why enumeration-based approaches perform better than column generation in the solution of the PSP.

Acknowledgement

This work has been partially completed while the author was a member in the Faculty of Engineering at Bahçeşehir University. This study is supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) under grant 113M480.

Akca, Z., 2010. Integrated location, routing and scheduling problems: Models and algorithms. Ph.D. thesis, Lehigh University.

Avella, P., D'Auria, B., Salerno, S., 2006. A LP-based heuristic for a time-constrained routing problem. European Journal of Operational Research 173 (1), 120–124.

Avella, P., Sassano, A., Vasilev, I., 2007. Computational study of large-scale p-median problems. Mathematical Programming 109 (1), 89–114.

Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., Vance, P., 1998. Branch-and-price: column generation for solving huge integer programs. Operations Research 46 (3), 316–329.

Chvátal, V., 1983. Linear Programming. Series of books in the mathematical sciences. W. H. Freeman.

Dyckhoff, H., 1990. A typology of cutting and packing problems. European Journal of Operational Research 44 (2), 145 – 159.

Ferreira, J., Neves, M., Castro, P., 1990. A two-phase roll cutting problem. European Journal of Operational Research 44 (2), 185–196.

Frangioni, A., Gendron, B., 2009. 0-1 reformulations of the multicommodity capacitated network design problem. Discrete Applied Mathematics 157 (6), 1229–1241.

Gilmore, P., Gomory, R. E., 1965. Multistage cutting stock problems of two and more dimensions. Operations Research 13 (1), 94–120.

Gilmore, P. C., Gomory, R. E., 1961. A linear programming approach to the cutting-stock problem. Operations Research 9, 849–859.

Gilmore, P. C., Gomory, R. E., 1963. A linear programming approach to the cutting-stock problem - part ii. Operations Research 11, 863–888.

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

Haessler, R. W., 1979. Solving the two-stage cutting stock problem. Omega 7 (2), 145–151.

Horowitz, E., Sahni, S., 1974. Computing partitions with applications to the knapsack problem. Journal of ACM 21, 277–292.

Hoto, R., Arenales, M., Maculan, N., 2007. The one dimensional compartmentalised knapsack problem: a case study. European Journal of Operational Research 183 (3), 1183–1195.

Irnich, S., Desaulniers, G., 2005. Shortest Path Problems with Resource Constraints. Springer US, Boston, MA, pp. 33–65.

Johnston, R., Khan, L., 1995. Bounds for nested knapsack problems. European Journal of Operational Research 81 (1), 154–165.

Katayama, N., Chen, M., Kubo, M., 2009. A capacity scaling heuristic for the multicommodity capacitated network design problem. Journal of Computational and Applied Mathematics 232 (1), 90–101.

Leão, A. A. S., Santos, M. O., Hoto, R., Arenales, M. N., 2011. The constrained compartmentalized knapsack problem: mathematical models and solution methods. European Journal of Operational Research 212 (3), 455–463.

Lübbecke, M. E., Desrosiers, J., 2005. Selected topics in column generation. Operations Research 53 (6), 1007–1023.

Marques, F. P., Arenales, M. N., 2007. The constrained compartmentalised knapsack problem. Computers & Operations Research 34 (7), 2109–2129.

Martello, S., Toth, P., 1990. Knapsack problems: algorithms and computer implementations. John Wiley & Sons, Inc., New York, NY, USA.

Muter, İ., Birbil, Ş. İ., Bülbül, K., Şahin, G., 2012. A note on "A LP-based heuristic for a time-constrained routing problem". European Journal of Operational Research 221, 306–307.

Muter, İ., Birbil, Ş. İ., Bülbül, K., Şahin, G., 2013a. Simultaneous column-and-row generation for large-scale linear programs with column-dependent-rows. Mathematical Programming 142 (1-2), 47–82.

Muter, İ., Birbil, Ş. İ., Bülbül, K., Şahin, G., Taş, D., Tüzün, D., Yenigün, H., 2013b. Solving a robust airline crew pairing problem with column generation. Computers & Operations Research 40 (3), 815–830.

Muter, İ., Cordeau, J.-F., Laporte, G., 2014. A branch-and-price algorithm for the multidepot vehicle routing problem with interdepot routes. Transportation Science 48 (3), 425–441.

Valério de Carvalho, J. M., 2002. LP models for bin packing and cutting stock problems. European Journal of Operational Research 141 (2), 253–273.

Valério de Carvalho, J. M., Rodrigues, A. G., 1995. An LP-based approach to a two-stage cutting stock problem. European Journal of Operational Research 84 (3), 580–589.

Vance, P., 1998. Branch-and-price algorithms for the one-dimensional cutting stock problem. Computational Optimization and Applications 9, 211–228.

Vanderbeck, F., 2001. A nested decomposition approach to a three-stage, two-dimensional cutting-stock problem. Management Science 47 (6), 864–879.

Wang, G., Tang, L., 2010. A row-and-column generation method to a batch machine scheduling problem. In: Proceedings of the Ninth International Symposium on Operations Research and Its Applications (ISORA-10). Chengdu-Jiuzhaigou, China, pp. 301–308.

Wäscher, G., Haußner, H., Schumann, H., 2007. An improved typology of cutting and packing problems. European Journal of Operational Research 183 (3), 1109–1130.

Xu, Z., 2013. The knapsack problem with a minimum filling constraint. Naval Research Logistics 60 (1), 56–63.

Zak, E. J., 2002a. Modeling multistage cutting stock problems. European Journal of Operational Research 141 (2), 313–327.

Zak, E. J., 2002b. Row and column generation technique for a multistage cutting stock problem. Computers & Operations Research 29 (9), 1143–1156.