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 a two-stage extension of one-dimensional cutting stock problem which arises when technical requirements inhibit cutting large stock rolls to demanded widths of finished rolls directly. Therefore, demands on finished rolls are fulfilled through two subsequent cutting processes, in which rolls produced in the former are used as input for the latter, while the number of stock rolls used is minimized. We tackle the pattern-based formulation of this problem which typically has a very large number of columns 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), zeysezer.81@gmail.com (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, some multi-dimensional cutting stock problems (Gilmore and Gomory (1965), Vanderbeck (2001)) are also referred to as MSCS problems in the literature. 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 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 times these cutting patterns are 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 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 any characteristic 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 an auxiliary knapsack problem in terms of the number of intermediate rolls that are generated by dynamic programming 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 loading 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 loaded items. The analogy between this problem and the TSCS problem is that compartments, as intermediate rolls, are bounded from below and from above, 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 it is based on the pre-enumeration of compartments, which requires solving a knapsack problem a large number of times. 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.

In general, cutting stock problems are 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 it 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). The peculiar structure of these problems stems from structural linking constraints that incapacitate the use of column generation: when a RMP is formed with a subset of variables, only the linking constraints induced by these variables exist in the RMP. This problem, which is restricted in terms of both columns and rows, is referred to as the short RMP (SRMP), and necessitates generating rows simultaneously with columns. 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 row-generating PSP that correctly 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), integrated airline recovery problem (Maher 2015) and time-constrained routing (Avella et al. 2006, Muter et al. 2012). Simultaneous generation of rows in column generation has also been devised for formulations that do not follow the assumptions of problems with column-dependent-rows ((Valério de Carvalho 1999) and (Mamer and McBride 2000)). However, these problems are not within the scope of this paper.

In TSCS, the two stages of the cutting process are tied by the linking constraints that enforce 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 at each iteration 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. However, no exact algorithm has yet been proposed in the literature to solve this large-scale PSP formulation.

In this paper, we solve the LP relaxation of 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, each involving 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 algorithms, 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 additional 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. 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 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. In Section 3, the novel component of this algorithm, which is the row-generating PSP, is constructed and the exact algorithms proposed for its solution are presented. 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, the pattern-based formulation of the TSCS problem, and an overview of the simultaneous column-and-row generation algorithm to solve this formulation are presented. Two subproblems are then constructed for each variable set to generate cutting patterns in their respective stages. 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 problem, 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 of width W in two successive 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 times k and n are performed in the solution are represented by integer decision 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

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

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

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

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

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

where columns of C and B correspond to the first and the second stage cutting patterns, respectively. In this formulation, (1) minimizes the total number of stock rolls cut through the first stage cutting patterns, while constraints (2) impose that the demands on finished rolls are satisfied. Each second stage cutting pattern n is linked to exactly one intermediate roll i through a single non-zero entry of $D_{in} = -1$ in column n of D, and thus constraints (3), referred to as the linking constraints, restricts the number of i processed in the second stage to the number of i produced in the first stage.

Note that, the enumeration of set I is not possible, i.e., any intermediate roll whose width is within $[s^{min}, s^{max}]$ is a feasible intermediate roll, nor is it essential in the solution of this problem since an optimal solution of (1)-(5) requires only a finite number of intermediate rolls, and hence, a finite set of linking constraints (3). In fact, it was shown in Zak (2002a,b) that an intermediate roll can exist in the optimal solution of this problem only if it corresponds to a second stage cutting pattern. Even then, 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 large cardinalities of the subproblems (A discussion on *a priori* enumeration of I is presented in Section 4.) 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.

By following a similar logic as in column generation, an SRMP is formed with subsets of $\bar{K} \subset K$ and $\bar{N} \subset N$, which induce only a subset of linking constraints. These constraints are associated with intermediate rolls that tie $\bar{N}$ to $\bar{K}$. This 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 intermediate rolls and the associated linking constraints that currently exist in SRMP. Thus, generation of a first stage cutting pattern, which includes a set of intermediate rolls currently absent from SRMP adds new linking constraints to the model. Consequently, SRMP grows both column-wise and row-wise through the iterations of a column generation algorithm, 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.

Outline of Simultaneous Column-and-Row Generation

Finding an optimal solution for the LP relaxation of MP 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

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

To check whether the optimal dual variable values v_m, $m \in M$ and w_i, $i \in \bar{I}$ retrieved from the solution of SRMP satisfy dual constraint sets (12) and (13), two pricing subproblems are defined to generate minimum reduced cost $y-$ and $x-$variables, respectively. These PSP search for new k and n that can be generated with $\bar{I}$. The PSP associated with (12), referred to as $y-$PSP, that searches for a negative reduced cost first stage cutting pattern is

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

where γ_i is the width of intermediate roll $i \in \bar{I}$, and the entries of C_i represent the number of times i appear in a newly generated first stage cutting pattern. (16) 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 a second stage cutting pattern n is cut from intermediate roll i. Hence, the dual constraints (13) reduce to $\sum_{m \in M} B_{mn} v_m \leq w_i$, and an $x-$PSP, that seeks a negative reduced cost second stage cutting pattern is solved for each $i \in \bar{I}$:

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

where α_m is the width of $m \in M$, and B_m^i is the new second stage cutting pattern represented by the number of times m is cut from i. In TSCS problem, it is assumed that a minimum trim loss of e^{min} is incurred from cutting each intermediate roll, defined as the mandatory minimum edge. The constraint in (17) imposes that the sum of widths of finished rolls that could be included in n plus e^{min} can not exceed the width of the intermediate roll it is cut from. Like $y-$PSP, $x-$PSP is an unbounded knapsack problem. If $\zeta_x^i > w_i$, the resulting

second stage cutting pattern cut from i is added to $\bar{N}$.

If the set of intermediate rolls I that can appear in an optimal solution of MP was known and included in SRMP, the optimal LP solution of the MP would have been found only by applying column generation, and solving $y-$PSP and $x-$PSP repeatedly until neither of both generate a negative reduced cost column. On the other hand, for a given subset of intermediate rolls $\bar{I}$, the definitions of $y-$PSP and $x-$PSP are confined to intermediate rolls currently existing in 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, SRMP grows only column-wise when new 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 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 SRMP. This formulation is tackled 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. The initial SRMP is constructed with subsets of I, B, and C using a simple procedure: first, for each $m \in M$ satisfying $\lfloor (s^{max} - e^{min})/\alpha_m \rfloor \, \alpha_m + e^{min} \geq s^{min}$, an intermediate roll is generated and is added to $\bar{I}$. Then, to account for those n that can appear in an optimal solution even though their second stage trim is larger than e^{min}, a single intermediate roll whose width is equal to s^{min} is added to $\bar{I}$. $\bar{C}$ and $\bar{B}$ are then initialized using $\bar{I}$. The initial $\bar{C}$ is a diagonal matrix with $C_{ii} = \lfloor W/\gamma_i \rfloor$, $\forall i \in \bar{I}$, whereas, $\bar{B}$ is a horizontal concatenation of diagonal matrices B^i, $\forall i \in \bar{I}$, with $B^i_{mm} = \lfloor (\gamma_i - e^{min})/\alpha_m \rfloor$, $\forall m \in M$. After the initialization of SRMP, $y-$PSP is invoked to identify a negative reduced cost first stage cutting pattern composed of intermediate rolls currently present in SRMP. If $\zeta_y > 1$, a new $k \in K\backslash\bar{K}$ is added to $\bar{K}$, and $y-$PSP is called again with new values of w_i, $i \in \bar{I}$ obtained from the solution of SRMP. Otherwise, the algorithm moves on to $x-$PSP, which searches for negative reduced cost second stage cutting patterns cut from each $i \in \bar{I}$. If the sum of dual values of finished rolls cut from i is larger than the dual value associated with i, i.e., $\zeta^i_x > w_i$, a new $n \in N\backslash\bar{N}$ is added to $\bar{N}$, and the algorithm returns to $x-$PSP after retrieving the values of the dual variables from the optimal solution of SRMP. After iterations of $x-$PSP on $\bar{I}$, the algorithm calls $y-$PSP if a negative reduced cost n is found for at least one i. If consecutive calls to $y-$ and $x-$PSP do not result in a negative reduced cost column, (identified as -vely priced column in Figure 1), the algorithm moves on to the row-generating PSP. The row-generating PSP is constructed to price out first stage cutting patterns, which contain intermediate rolls in $I\backslash\bar{I}$. Generation of these new $y-$variables triggers new linking constraints corresponding to $i \in I\backslash\bar{I}$ and new $x-$variables cut from these i to be added to SRMP, which leads SRMP to grow both column-wise and row-wise. If consecutive calls to all three PSP do not result in a negative cost column, the current solution of SRMP is optimal for the LP relaxation of (1)-(5).

The TSCS problem can also be formulated in an alternative way so as to circumvent simultaneous column-and-row generation. Rather than utilizing two types of variables as in

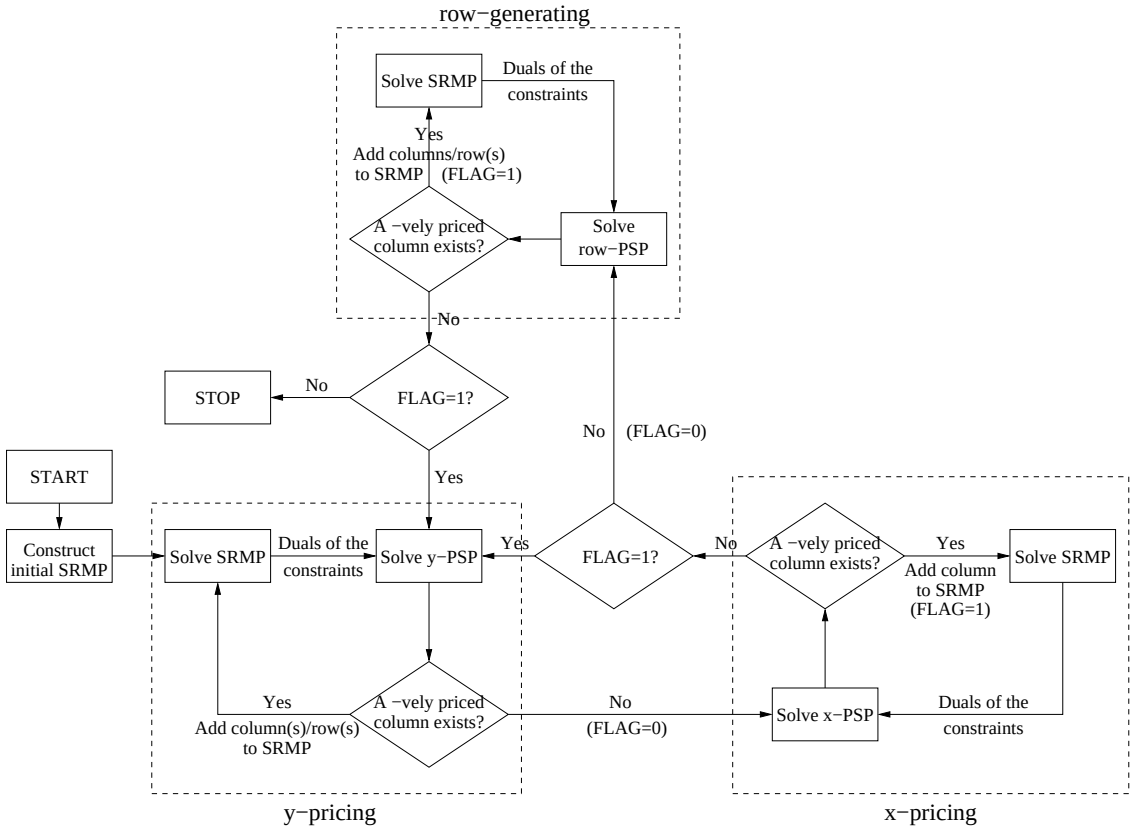

Figure 1: Simultaneous column-and-row-generation algorithm (Muter et al. (2013a)).

(1)-(5), the alternative model is composed of only y_k, $k \in K$, and a new matrix A, in which an entry A_{mk} represents the number of finished roll $m \in M$ cut in pattern $k \in K$. Therefore, the TSCS can be reformulated as follows:

$$\text{minimize} \quad \sum_{k \in K} y_k, \tag{18}$$

$$\text{subject to} \quad \sum_{k \in K} A_{mk} y_k \geq b_m, \qquad m \in M, \tag{19}$$

$$y_k \geq 0, \qquad k \in K. \tag{20}$$

This model, which is similar to the prominent Gilmore-Gomory model, does not have the linking constraints, and therefore, it can be solved by column generation without the simultaneous generation of rows. The intermediate rolls are implicitly incorporated within first stage cutting patterns $k \in K$ as these rolls are formed by second stage cutting patterns formed of finished rolls. Hence, those $m \in M$ for which $A_{mk} > 0$ are feasibly grouped into feasible intermediate rolls. The RMP of this problem comprises a subset of patterns $\bar{K}$, and its solution yields the values of the dual variables v_m, $m \in M$ associated with (19). The $y-$PSP and $x-$PSP, which are solved over existing intermediate rolls as explained previously, cannot partake in

the column generation algorithm to solve (18)-(20) due to the absence of dual variables associated with the intermediate rolls. In Algorithm 1, these subproblems are invoked before the row-generating PSP in an effort to minimize the number of times this complex subproblem is solved. Consequently, the solution of (18)-(20) by column generation mandates invoking the computationally burdensome row-generating PSP at each iteration, which clearly is a disadvantage over the simultaneous column-and-row generation (Note that the term *row-generating* used to identify the subproblem refers to the generation of new intermediate rolls along with the cutting patterns). Moreover, we show in Section 4 through computational experiments that only a small set of rows are generated by the simultaneous column-and-row generation algorithm, which does not have a discernible impact on the overall solution performance.

3. Row-Generating PSP

The difficulty in constructing a row-generating PSP is twofold: first, widths of intermediate rolls currently not included in $\bar{I}$ are unknown except for their allowable interval, and many of these rolls may need to be generated simultaneously in the solution of the row-generating PSP. Second, dual variables associated with these rolls are not known except that they are non-negative in a feasible dual solution, as imposed in (15). These dual variables must be correctly estimated to be factored in the solution of the row-generating PSP.

3.1. Mathematical Model and Optimality

The objective of the row-generating PSP is to find a negative reduced cost k that contains $i \in I \backslash \bar{I}$, hence, it can be formulated as an extension of (16) as follows:

$$\zeta_{xy} = \text{maximize} \quad \sum_{i \in \bar{I}} w_i C_i + \sum_{i \in I \backslash \bar{I}} w_i C_i, \tag{21}$$

$$\text{subject to} \quad \sum_{i \in \bar{I}} \gamma_i C_i + \sum_{i \in I \backslash \bar{I}} \gamma_i C_i \leq W, \tag{22}$$

$$s^{min} \leq \gamma_i \leq s^{max}, \quad i \in I \backslash \bar{I}, \tag{23}$$

$$w_i \geq 0, \quad i \in I \backslash \bar{I}, \tag{24}$$

$$C_i \in \mathbb{Z}^+ \cup \{0\}, \quad i \in I, \tag{25}$$

which is similar to $y-$PSP except for the second terms in (21) and (22) associated with the unknown intermediate rolls in $I \backslash \bar{I}$ and bounding constraints (23) and (24). 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

- an intermediate roll can appear in an optimal solution if

$$s^{min} \leq \gamma_i = \sum_{m \in M} \alpha_m B_m^i + e^{min} \leq s^{max} \tag{26}$$

for some second stage cutting pattern (in which case, the corresponding second stage trim is exactly e^{min}), or

$$\gamma_i = s^{min} > \sum_{m \in M} \alpha_m B_m^i + e^{min} \tag{27}$$

(in which case, second stage trim is larger than e^{min}). Note that inclusion of an intermediate roll having width s^{min} into the initial SRMP ensures that any new intermediate roll generated by the row-generating PSP satisfies (26).

- 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.

$$w_i = \sum_{m \in M} v_m B_m^i, \tag{28}$$

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

Plugging in (26) and (28), we reach the following mathematical model of the row-generating PSP:

$$\zeta_{xy} = \text{maximize} \quad \sum_{i \in \bar{I}} w_i C_i + \sum_{i \in I \backslash \bar{I}} (\sum_{m \in M} v_m B_m^i) C_i, \tag{29}$$

$$\text{subject to} \quad \sum_{i \in \bar{I}} \gamma_i C_i + \sum_{i \in I \backslash \bar{I}} (\sum_{m \in M} \alpha_m B_m^i + e^{min}) C_i \leq W, \tag{30}$$

$$s^{min} \leq \sum_{m \in M} \alpha_m B_m^i + e^{min} \leq s^{max}, \quad i \in I \backslash \bar{I} \tag{31}$$

$$C_i \in \mathbb{Z}^+ \cup \{0\}, \quad i \in I, \tag{32}$$

$$B_m^i \in \mathbb{Z}^+ \cup \{0\}, \quad i \in I \backslash \bar{I}, \tag{33}$$

in which v_m values are retrieved from the solution of SRMP. If the solution of (29)-(33) returns $\zeta_{xy} > 1$, the row-generating PSP yields a k containing a set of $i \in I \backslash \bar{I}$ together with new sets of linking constraints and n induced by these new i. This model is still intractable due to non-linearity associated with the unknown intermediate rolls, which also features a large number of columns and constraints.

In order to alleviate these difficulties, Zak (2002a) restricts the number of new intermediate rolls added to SRMP at each call of the row-generating PSP to one. For this new intermediate roll $i \in I \backslash \bar{I}$, (29)-(33) is solved for integer values of C_i between $0 \leq C_i \leq (W - e^{min})/s^{min}$, and the solution that attains the largest objective function value is selected. This heuristic approach is shown to perform successfully in reaching LP optimality for a large number of instances, as observed in the computational experiments conducted in Zak (2002a) and those explained in Section 4.

Having formulated the row-generating PSP, LP convergence of the proposed methodology given in Figure 1 can now be proven. This proof is a special version of that presented for general column-dependent-rows problems in Muter et al. (2013a).

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. Let SRMP($\bar{K}$, $\bar{N}$, $\bar{I}$) denote the current SRMP and $\mathbf{B}$ the corresponding optimal basis, a $\delta \times \delta$ matrix. The solution of the row-generating PSP 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 $\sum_{m \in M} v_m B^i_m$. Therefore, after y_k is added, 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 (28) 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.

When the row-generating PSP is invoked, it is already guaranteed by $\zeta_y \leq 1$ and $\zeta^i_x \leq w_i$, $i \in \bar{I}$ 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 $\delta_k \times \delta_k$ 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 $\delta \times |\Delta_k|$ matrix $\mathbf{F} = \begin{pmatrix} \mathbf{B}^T_{new} & \mathbf{0} \end{pmatrix}^T$ 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 $|\Delta_k| \times |\Delta_k|$ submatrix $-\mathbf{I}$ is comprised of $D_{in} = -1$ for $i \in \Delta_k$ and $n \in S_N(k)$. Finally, the $|\Delta_k| \times \delta$ submatrix $\mathbf{0}$ contains the entries of basic variables in 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 dual variables induced by $\mathbf{B}_k$ become

$$\begin{pmatrix} \mathbf{c}_\mathcal{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{B}^{-1} & \mathbf{B}^{-1}\mathbf{F} \\ \mathbf{0} & -\mathbf{I} \end{pmatrix} = \begin{pmatrix} \mathbf{c}_\mathcal{B}\mathbf{B}^{-1} & \mathbf{c}_\mathcal{B}\mathbf{B}^{-1}\mathbf{F} \end{pmatrix}, \tag{34}$$

where $\begin{pmatrix} \mathbf{c}_\mathcal{B} & \mathbf{0} \end{pmatrix}$ is the vector of objective function coefficients of 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 (34), the values of dual variables after augmenting the basis are preserved as $\mathbf{c}_\mathcal{B}\mathbf{B}^{-1} = \begin{pmatrix} \mathbf{v} & \bar{\mathbf{w}} \end{pmatrix}$ where $\mathbf{v}$ and $\bar{\mathbf{w}}$ contain v_m, $m \in M$ and w_i, $i \in \bar{I}$. Moreover, we can show that the values of 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

$$\mathbf{c}_{\mathcal{B}}\mathbf{B}^{-1}\mathbf{F} = \begin{pmatrix} \mathbf{v} & \bar{\mathbf{w}} \end{pmatrix} \begin{pmatrix} \mathbf{B}_{new} \\ \mathbf{0} \end{pmatrix} = \mathbf{v}\mathbf{B}_{new}. \tag{35}$$

Therefore, the values of w_i, $i \in \Delta_k$ are as set in (28). 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 (29)-(33) 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 $\mathbf{B}_k$ enable to calculate the reduced cost of y_k correctly, which is achieved by (29)-(33).

Using the same augmentation, an optimal basis can be formed for SRMP($\bar{K}$, N, I) by selecting an $x-$variable as basic for each $i \in I \backslash \bar{I}$. Therefore, when a call to (29)-(33) 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). □

3.2. Proposed Algorithms

The main motivation behind the proposed algorithms is to solve the row-generating PSP that generates a new first stage cutting pattern featuring new intermediate rolls $i \in I \backslash \bar{I}$. Generation of these intermediate rolls is directly related to the generation of feasible second stage cutting patterns through equations (26) and (28). All three algorithms are, therefore, based on generating either the *best* second stage cutting pattern in an iterative manner or a collection of good second stage cutting patterns through implicit enumeration.

To generate feasible cutting patterns, Valério de Carvalho (2002) developed an arc flow model that is based on a shortest path formulation of the knapsack problem. However, the lower and the upper bounds as well as additional constraints –which are explained along with the proposed algorithms– imposed on the generation of second stage cutting patterns prompted us to cast a shortest path problem with resource constraints. For each proposed method, a variation of the labeling algorithm described in Irnich and Desaulniers (2005) is applied. These labeling algorithms differ in the set of resources and the dominance relations used to eliminate unnecessary second stage cutting patterns, and are defined for each algorithm

separately in their respective sections. Nevertheless, they all operate on a common 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 a dummy sink node $t = |M| + 1$. The nodes in G are topologically sorted so that each node is treated only once in ascending order. Associated with each node $\imath \in V$, there is a width $\alpha_\imath$ and a dual value $v_\imath$, and for s and t, both values are zero. $A = A_1 \cup A_2$ is the set of arcs which consists of $A_1 = \{(\imath, \jmath) : \imath \in V, \jmath \in V, \imath < \jmath\}$ and $A_2 = \{(\imath, \imath) : \imath \in V \backslash \{s, t\}\}$. When forming a path, 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. A complete path from s to t corresponds to a second stage cutting pattern and the loop arcs in A_2 allow inclusion of more than one finished roll in these patterns. Let label $X_p^\imath$ denote a partial path p on G, starting from s and ending at some node $\imath \in V \backslash \{0\}$ and let $B_\ell^{\imath^p}$, $\ell \in V$ indicate the number of times node ℓ is visited on path p. Define $R_D^{\imath^p} = \sum_{\ell \in V} v_\ell B_\ell^{\imath^p}$ and $R_W^{\imath^p} = \sum_{\ell \in V} \alpha_\ell B_\ell^{\imath^p}$ associated with $X_p^\imath$. A label $X_p^\imath$ corresponding to path p can be extended from node $\imath$ to node $\jmath \in V \backslash \{t\}$ for $\jmath \geq \imath$ using arc $(\imath, \jmath)$ if $R_W^{\imath^p} + \alpha_\jmath \leq s^{max}$. In this case, a new label corresponding to a new path q, denoted by $X_q^\jmath$, is formed with $R_D^{\jmath^q} = R_D^{\imath^p} + V_\jmath$, $R_W^{\jmath^q} = R_W^{\imath^p} + \alpha_\jmath$ and $B_\ell^{\jmath^q} = B_\ell^{\imath^p}$ for $\ell \in V \backslash \{\jmath\}$ and $B_\jmath^{\jmath^q} = B_\jmath^{\imath^q} + 1$. On the other hand, when $\jmath = t$, the feasibility condition is $R_W^{\imath^p} \geq s^{min}$, which ensures that a 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 $R_D^{s^p} = 0$, $R_W^{s^p} = e^{min}$ and $B_\ell^{s^p} = 0$, $\ell \in V$. When node $\imath \in V$ is selected to be treated at the $\imath^{th}$ iteration of the algorithm, labels at this node are first extended with the loop arc $(\imath, \imath)$. After no other feasible label can be created with the inclusion of the loop arc, all labels on this node are extended to reachable nodes $\jmath > \imath$ by applying the dominance rules described for each algorithm, so that only non-dominated labels are preserved.

3.2.1. Two-Phase Approach

As the name implies, the two-phase approach tackles the row-generating PSP in two phases. In the first-phase, a subset of intermediate rolls are generated through (26) and (28) by implicit enumeration based on dynamic programming. In the second-phase, this subset of intermediate rolls are substituted for the unknown intermediate rolls in (29)-(33) to generate a first stage cutting pattern with a minimum reduced cost. Observe that with this substitution the row-generating PSP reduces to $y-$PSP and can easily be solved as an unbounded knapsack problem. Such a two-phase algorithm has been applied to TSCS problem by Valério de Carvalho and Rodrigues (1995), in which only homogeneous intermediate rolls were considered. 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). It is well-known that not all intermediate rolls are essential in the solution of unbounded knapsack problems as some of them may be dominated by others, and therefore, can be fathomed from the problem without affecting the optimal solution. One basic dominance relation which can be applied to $y-$PSP is given below.

DEFINITION 3.1 *For a pair of intermediate rolls i and j, i dominates j if $w_i \geq w_j$ and $\gamma_i \leq \gamma_j$.*

In fact, Valério de Carvalho and Rodrigues (1995) used a variation of this rule for a

knapsack subproblem in their work. The two-phase algorithm developed in this paper is based on a similar idea, where in the first-phase the dominance rules applied in the labeling algorithm are modified to consider (26) and (28). Given these non-dominated intermediate rolls, the row-generating PSP is then solved as an unbounded knapsack problem in the second phase. Since v_m, $m \in M$ change after the solution of SRMP, this labeling algorithm is applied at each call to the row-generating PSP.

One obvious dominance relation is determined by the lower bound imposed on widths of intermediate rolls. Without s^{min}, dominance relation given in Definition 3.1 applied to a given pair of labels, corresponding to paths p and q at node $\imath$, would result in elimination of q if $R_D^{\imath^p} \geq R_D^{\imath^q}$ and $R_W^{\imath^p} \leq R_W^{\imath^q}$. However, with s^{min} imposed as a lower bound, this dominance relation applies only if both $R_W^{\imath^p} \geq s^{min}$ and $R_W^{\imath^q} \geq s^{min}$ or if $\imath = t$. 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 $\imath$ satisfying $R_D^{\imath^p} \geq R_D^{\imath^q}$, $R_W^{\imath^p} \leq R_W^{\imath^q}$ and $R_W^{\imath^p} < s^{min}$ are not comparable since not all extensions of these paths to t that are 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. $R_W^{\imath^p} = R_W^{\imath^q}$. Therefore, the following dominance relation is used.

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

- $\imath \neq t$, $R_W^{\imath^p} < s^{min}$ *and* $R_W^{\imath^p} = R_W^{\imath^q}$
- $\imath \neq t$, $s^{min} \leq R_W^{\imath^p} \leq R_W^{\imath^q}$
- $\imath = t$ *and* $R_W^{\imath^p} \leq R_W^{\imath^q}$

When all nodes $\imath \in V$ are treated, labels at node t constitute the non-dominated intermediate roll set, denoted by $I^{'}$. After adding intermediate rolls in $I^{'}$ to $\bar{I}$, the knapsack problem (16) is solved in the second-phase of the two-phase approach. To further eliminate some of the intermediate rolls in $I^{'}$ a well-known result of integer programming, which was also employed in Muter et al. (2014), is used. The implication of this result to 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) is the objective function value of the LP relaxation that is readily available after putting intermediate rolls in a non-increasing order of (w_i/γ_i) for $i \in I^{'}$. Let the indices of intermediate rolls correspond to their positions 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 on ζ_{xy}, is $UB = W w_{i_1}/\gamma_{i_1}$. In the dual optimal solution, dual value of the knapsack constraint 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 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.

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

It is shown 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 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).

3.2.2. Branch-and-Price

The second algorithm tackles the reformulation (29)-(33) by column generation, in which the unknown intermediate rolls associated with C_i, $i \in I\backslash\bar{I}$ are generated via solving a PSP at a second level. Solving both the MP and the row-generating PSP using column generation constitute nested column generation for the TSCS problem. The second-level column generation method to solve the row-generating PSP is embedded in a branch-and-bound algorithm to find the integer values of C_i, which reveals a novel branch-and-price algorithm designed specifically for a knapsack problem.

Let RMP at the second-level, referred to as RMPSP, be initialized with C_i, $i \in \bar{I}$ in (29) and (30). In its initial form RMPSP is equivalent to the LP relaxation of (16). Then a second level PSP, referred to as PSPSP, can be constructed using (26) and (28) to generate positive reduced cost intermediate rolls in $I\backslash\bar{I}$ as

$$\begin{aligned} \text{(PSPSP)} \quad \tau_{2S} = \text{maximize} \quad & \sum_{m\in M} v_m B_m - \beta(\sum_{m\in M} \alpha_m B_m + e^{min}) \\ & = \sum_{m\in M} (v_m - \beta\alpha_m) B_m - \beta e^{min}, \\ \text{subject to} \quad & s^{min} \leq \sum_{m\in M} \alpha_m B_m + e^{min} \leq s^{max}, \\ & B_m \in \mathbb{Z}^+ \cup \{0\}, m \in M, \end{aligned} \tag{36}$$

where β denotes the dual variable corresponding to the knapsack constraint (30) in RMPSP. (36) is known as an unbounded knapsack problem with a minimum filling constraint in the literature. Xu (2013) has introduced the binary version of this problem, and proposed an approximation algorithm for its solution. If the optimal solution of PSPSP, denoted by $\hat{B}_m$, $m \in M$, attains $\tau_{2S} > 0$, an intermediate roll i^* with $w_{i^*} = \sum_{m\in M} v_m \hat{B}_m$ and $\gamma_{i^*} = \sum_{m\in M} \alpha_m \hat{B}_m + e^{min}$ is added to $\bar{I}$, and RMPSP is solved again. It should be pointed out that although $\tau_{2S} > 0$ guarantees $w_{i^*}/\gamma_{i^*} > \beta$ by

$$\sum_{m\in M} (v_m - \beta\alpha_m)\hat{B}_m - \beta e^{min} = \quad w_{i^*} - \beta\gamma_{i^*} > 0,$$

it does not guarantee that it generates the intermediate roll with the largest ratio, which requires solving an integer linear fractional programming problem. When PSPSP does not result in an intermediate roll with a positive reduced cost, the second level column generation terminates with an optimal solution to the LP relaxation of RMPSP. To find an optimal integer

solution of the row-generating PSP, 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, needs to be developed.

At the root node of the branch-and-price framework, the primal and the dual optimal solutions of RMPSP are $\hat{C}_{i_1} = W/\gamma_{i_1}$, $\hat{C}_i = 0$, $i \in \bar{I}\backslash\{i_1\}$, and $\beta = w_{i_1}/\gamma_{i_1}$, respectively. Suppose that, given β, the optimal solution of PSPSP renders $\tau_{2S} > 0$, and $i^* \in I \setminus \bar{I}$ is the newly generated intermediate roll satisfying $w_{i^*}/\gamma_{i^*} > \beta$. After i^* is added to RMPSP, the new dual solution becomes $\beta = w_{i^*}/\gamma_{i^*}$. Suppose there exists no other $i \in I\backslash\bar{I}$ whose ratio is larger than w_{i^*}/γ_{i^*}, and the only nonzero variable in the solution, that is $\hat{C}_{i^*} = W/\gamma_{i^*}$, is fractional. 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 RMPSP becomes $\hat{C}_{i^*} = \lfloor W/\gamma_{i^*} \rfloor$ and $\hat{C}_{i_1} = (W - \gamma_{i^*} \lfloor W/\gamma_{i^*} \rfloor)/\gamma_{i_1}$. Yet the dual optimal solution returns $\beta = w_{i_1}/\gamma_{i_1}$, and thus, previously generated intermediate roll i^*, whose reduced cost is non-positive in RMPSP, is generated again by PSPSP with a positive reduced cost in the next iteration. This points to a crucial issue of integrating dual variables associated with the branching constraints to the subproblem in branch-and-price frameworks. Similar difficulties in branching were also faced by Vance (1998) in solving the conventional one-dimensional cutting stock problem. In her work, she transformed an unbounded knapsack subproblem to a 0-1 knapsack problem using the transformation algorithm given in Martello and Toth (1990) and solved it with a modified version of the method developed in Horowitz and Sahni (1974) by applying prevention rules. Application of this technique to the row-generating PSP proved inefficient in the preliminary experiments due to the increase in the number of nodes and the prevention constraints applied at each node. Consequently, to solve (36) in the course of a branch-and-price algorithm, a knapsack algorithm that avoids regeneration of existing intermediate rolls is developed.

In our proposed approach, column generation is integrated within a 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 never been applied directly to a knapsack problem in the literature, the details of this embedded column generation approach is presented in Algorithm 1. The column generation procedure (ColGen), given in Algorithm 2, is called during Algorithm 1 to search for the promising intermediate rolls in $I\backslash\bar{I}$ when branching.

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

```
1  Input: z = 0, z* = 1, j = 0, Γ = 1000, rest = W, sol_l = 0, l = 1, 2, ..., |Ī| + 1 ;
2  Output: bestsol, z* ;
3  l = j ;
4  ColGen ;
5  l = l + 1 ;
6  UB = W(w_{i_l}/γ_{i_l}) ;
7  if UB < z* then
8      Stop ;
9  Step 1 ;
10 while rest ≥ s^min ∧ l < |Ī| + 1 do
11     sol_l = ⌊rest/γ_{i_l}⌋ ;
12     rest = rest − sol_l γ_{i_l} ;
13     z = z + sol_l w_{i_l} ;
14     ColGen ;
15     l = l + 1 ;
16     u = rest w_{i_l}/γ_{i_l} ;
17     if z* ≥ z + u then
18         Go to Step 3 ;
19     if u = 0 then
20         Go to Step 2 ;
21 end
22 Step 2 ;
23 if z* < z then
24     bestsol = sol ;
25     z* = z ;
26 if z* = UB then
27     Stop ;
28 Go to Step 3 ;
29 Step 3 ;
30 j = l − 1 ;
31 while j > 0 ∧ sol_j = 0 do
32     j = j − 1 ;
33 end
34 if j = 0 then
35     Stop ;
36 sol_j = sol_j − 1 ;
37 rest = rest + γ_{i_j} ;
38 z = z − w_j ;
39 if z* ≥ z + rest w_{i_{j+1}}/γ_{i_{j+1}} then
40     Go to Step 3 ;
41 l = j + 1 ;
42 Go to Step 1 ;
```

In initializing Algorithm 1, $i \in \bar{I}$ are rearranged in a non-increasing order of their w_i/γ_i ratios such that $w_{i_1}/\gamma_{i_1} > w_{i_2}/\gamma_{i_2} > ... > w_{i_{\bar{I}}}/\gamma_{i_{\bar{I}}}$, and an artificial intermediate roll with $w_i = 0$ and $\gamma_i = s^{max}$ is placed at the position $|\bar{I}| + 1$. The objective function value of the current solution, denoted by z, and the best lower bound, denoted by z^*, are set to zero and one, respectively. Let j indicate the index of the branching variable, which is initially zero. The algorithm begins with identifying the intermediate roll with the largest ratio so that an upper bound (Line 6) for the row-generating PSP can be calculated. Since the list of intermediate rolls is not complete, ColGen is invoked (Line 4) to search whether there exists an i^* satisfying $w_{i^*}/\gamma_{i^*} > w_{i_1}/\gamma_{i_1}$. In Step 1, the algorithm starts building a solution by iteratively incrementing l by one, which enables the most promising intermediate roll in $\bar{I}$ to be considered first in the solution. ColGen is invoked therewith to check if there exists an $i \in I \backslash \bar{I}$ whose ratio is larger than $w_{i_{l+1}}/\gamma_{i_{l+1}}$. However, similar to the issue with that of branching mentioned previously, PSPSP may result in an intermediate roll already existing in $\bar{I}$ whose ratio is larger than or equal to w_{i_l}/γ_{i_l}. Next lemma restricts the solution set of

PSPSP by ruling out intermediate rolls whose ratios are larger than w_{i_l}/γ_{i_l}, which accelerates the solution of this problem.

Algorithm 2: ColGen

```
1  Input: l, Γ ;
2  Output: Γ ;
3  if w_{i_{l+1}}/γ_{i_{l+1}} < Γ then
4      while true do
5          β = w_{i_{l+1}}/γ_{i_{l+1}} ;
6          Solve PSPSP1 ;
7          if τ_{2S} > 0 then
8              Insert i* in l + 1^{st} position ;
9          else
10             Γ = w_{i_{l+1}}/γ_{i_{l+1}} ;
11             break out of while ;
12     end
```

LEMMA 3.1 *If ColGen does not return a positive reduced cost intermediate roll for i_l, it can not return a positive reduced cost intermediate roll whose ratio is larger than that of i_l when solved for i_{l+1}.*

PROOF. By contradiction. Suppose that the column generation procedure is invoked before i_{l+1} is considered for branching, then the dual optimal solution of RMPSP is $\beta = w_{i_{l+1}}/\gamma_{i_{l+1}}$. Suppose also that PSPSP formed with this β renders $\tau_{2S} > 0$, and generates an intermediate roll i^* whose ratio is also larger than i_l. If $w_{i^*}/\gamma_{i^*} > w_{i_l}/\gamma_{i_l}$ then PSPSP solved for i_l with $\beta = w_{i_l}/\gamma_{i_l}$ could not have rendered $\tau_{2S} = w_{i^*} - (w_{i_l}/\gamma_{i_l})\gamma_{i^*} \leq 0$.

□

Using the result of the above lemma, the ratio of intermediate rolls that can be generated by PSPSP when $\beta = w_{i_{l+1}}/\gamma_{i_{l+1}}$ is restricted to be no larger than that of i_l by adding a constraint to (36) as

$$\sum_{m \in M} (v_m - (w_{i_l}/\gamma_{i_l})\alpha_m)B_m \leq (w_{i_l}/\gamma_{i_l})e^{min}. \tag{37}$$

The extension of PSPSP with (37) is referred to as PSPSP1, and is solved within Algorithm 2 (Line 6) to generate intermediate rolls whose ratios are between the ratios of i_l and i_{l+1}. If solving PSPSP1 with $\beta = w_{i_{l+1}}/\gamma_{i_{l+1}}$ results in $\tau_{2S} > 0$, a newly generated i^* satisfying $w_{i^*}/\gamma_{i^*} > w_{i_{l+1}}/\gamma_{i_{l+1}}$ is added to the list in the $l + 1^{st}$ position by ColGen (Line 8), and 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_l and i_{l+1}, as pointed out earlier. To keep track of the last β value for which PSPSP1 did not generate an intermediate roll, a global parameter Γ is defined. This parameter is initialized in Algorithm 1 with a sufficiently large number so that no intermediate roll is missed out in the first call to ColGen (Line 4). In any other call to ColGen with i_{l+1}, $l + 1 \neq 1$ (Line 14), the value of Γ is equal to w_{i_l}/γ_{i_l}, since Γ changes only if PSPSP1 renders $\tau_{2S} \leq 0$ and the lexicographic algorithm continues by considering the next item in the list (Line 15). It is clear that solving PSPSP1 would be futile if the current β is not smaller than Γ, hence, a control statement is executed in the beginning of ColGen (Line 3) to determine whether to proceed with the procedure. The remaining steps of Algorithm 1 are self-explanatory.

PSPSP1 is modeled as a shortest path problem with resource constraints on graph $G = (V, A)$, in which resources are defined for the total width and the left-and-side of constraint (37). The labeling algorithm described previously is adapted to reflect the characteristics of this optimization problem. At an iteration of PSPSP1, where $\beta = w_{i_{l+1}}/\gamma_{i_{l+1}}$ and $\Gamma = w_{i_l}/\gamma_{i_l}$, let $\bar{v}_m = v_m - \beta\alpha_m$ and $\hat{v}_m = v_m - \Gamma\alpha_m$, $m \in M$, denote the objective function coefficient, and the left-hand-side coefficient of (37), respectively. These values are set to zero for both s and t in G. Observe that $\bar{v}_\imath$ and $\hat{v}_\imath$ can have non-positive values for some $\imath \in V$. In an unbounded knapsack problem items with a non-positive objective function coefficient can be disregarded. In PSPSP1, however, finished rolls with $\bar{v}_\imath \leq 0$ and $\hat{v}_\imath \leq 0$, $\imath \in V$ must be considered due to the resource constraints. The nodes corresponding to finished rolls $\imath \in V\backslash\{s,t\}$ in G, are sorted in a descending order of $\bar{v}_\imath/\alpha_\imath$ ratios to accelerate the enumeration process using a bounding procedure. In addition to the previously defined resource $R_W^{\imath^p}$, $R_O^{\imath^p} = \sum_{\ell \in V} \bar{v}_\ell B_\ell^{\imath^p}$ and $R_L^{\imath^p} = \sum_{\ell \in V} \bar{v}_\ell B_\ell^{\imath^p}$ are employed to calculate the objective function value and the left-hand-side of (37) associated with label $X_p^\imath$, respectively. These values are initialized as zero at node s. Along the extension to $\jmath \in V\backslash\{t\}$ from $\imath \leq \jmath$, a new label corresponding to path q, denoted by $X_q^\jmath$, is formed by setting $R_W^{\jmath^q} = R_W^{\imath^p} + \alpha_\jmath$, $R_O^{\jmath^q} = R_O^{\imath^p} + \bar{v}_\jmath$, $R_L^{\jmath^q} = R_L^{\imath^p} + \hat{v}_\jmath$ and $B_\ell^{\jmath^q} = B_\ell^{\imath^p}$ for $\ell \in V\backslash\{\jmath\}$ and $B_\jmath^{\jmath^q} = B_\jmath^{\imath^p} + 1$.

PSPSP1 aims to find the second stage cutting pattern that maximizes the objective function. This pattern is a candidate to be inserted at the $l + 1^{st}$ position only if $R_O^{\imath^p} > \beta e^{min}$. Hence, a parameter $\rho = \beta e^{min}$ is initialized at the beginning of the labeling algorithm to keep track of the maximum objective function value of PSPSP1, and the following extension rule is applied:

DEFINITION 3.3 *A label $X_p^\imath$ corresponding to path p can be extended from node $\imath$ to node $\jmath \in V\backslash\{t\}$ for $\jmath \geq \imath$ using arc $(\imath, \jmath)$ if $R_W^{\imath^p} + \alpha_\jmath \leq s^{max}$, and one of the following conditions is satisfied:*

- $\bar{v}_\jmath > 0$ *and* $R_O^{\imath^p} + (s^{max} - R_W^{\imath^p})\bar{v}_\jmath/\alpha_\jmath > \rho$
- $\bar{v}_\jmath < 0$ *and* $R_O^{\imath^p} > \rho$

In the above conditions an upper bound is calculated for $R_O^{\jmath^q}$ to check whether partial path p can attain the best objective function value in the PSPSP1. When $\bar{v}_\jmath < 0$, the current objective function value $R_O^{\imath^p}$ is used as an upper bound. These upper bounds are valid since finished rolls are sorted in a non-increasing order of $\bar{v}_\jmath/\alpha_\jmath$ ratios and help preserve only those labels that can possibly attain the best objective function value. On the other hand, when extending paths to node t, the value of ρ is updated if a second stage cutting pattern satisfying $\sum_{\ell \in V} \bar{v}_\ell B_\ell^{\imath^p} > \rho$ is found. The extension rules to node t are summarized in Definition (3.4).

DEFINITION 3.4 *A label $X_p^\imath$ corresponding to path p can be extended from node $\imath$ to node $\jmath = t$ using arc $(\imath, \jmath)$ if all of the following conditions are satisfied:*

- $R_W^{\jmath^q} \geq s^{min}$, *i.e., it is within allowable widths,*
- $R_O^{\jmath^q} > \rho$, *i.e., it attains the maximum objective function value,*
- $R_L^{\jmath^q} \leq \Gamma e^{min}$, *i.e., its ratio is smaller than or equal to* Γ.

With the above conditions it is assured that a second stage cutting pattern found at the end of PSPSP1 is a feasible intermediate roll with a ratio between i_l and i_{l+1} that maximizes the objective function of PSPSP1.

In many calls to PSPSP1 through 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 PSPSP1 not only keeps the non-dominated rolls in $\bar{I}$ but also ensures that even if the optimal solution satisfies (37) 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 with the exception that at the terminal node t path q is compared with each $i \in \bar{I}$. Computational experiments has shown that the branch-and-price approach generates a smaller number of intermediate rolls to solve the row-generating PSP than other proposed methods.

3.2.3. 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 PSPSP1 through its course. PSPSP1 is generally called a 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 $l+1^{st}$ item at each iteration of ColGen, the set of non-dominated intermediate rolls between i_l and i_{l+1} are enumerated. 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 explained in Section 3.2.1 is applied. The aim of the hybrid approach is to generate the set of non-dominated intermediate rolls whose ratios are between the ratios of i_l and i_{l+1} when PSPSP1 is formed with $\beta = w_{i_{l+1}}/\gamma_{i_{l+1}}$ in Algorithm 1. In other words, all possible solutions satisfying the constraints of PSPSP1 are generated before the next item is considered in the lexicographic algorithm.

For the enumeration of the non-dominated rolls between a pair of adjacent rolls in $\bar{I}$, graph G defined in the solution procedure of PSPSP1 is utilized and the extension conditions in the labeling algorithm designed for PSPSP1 is adapted for the enumeration process. First, the parameter ρ is fixed to βe^{min} at termination to generate all positive reduced cost intermediate rolls satisfying $R_O^{\jmath^q} > \beta e^{min}$, $\jmath = t$. The dominance conditions defined in Definition 3.2 are also employed to rule out useless labels. Moreover, in order to ensure that $\bar{I}$ contains only the non-dominated intermediate rolls, the dominance conditions given in Definition 3.1 are applied while adding the feasible rolls at node t to $\bar{I}$. After the non-dominated rolls are inserted in $\bar{I}$, Γ is set to β. Following the lines of Algorithm 2, unless $w_{i_{l+1}}/\gamma_{i_{l+1}} < \Gamma$, the enumeration procedure is not invoked.

In the preliminary experiments, it is observed that the hybrid algorithm calls the enumeration scheme a large number of times, and in most of these calls the difference between w_{i_l}/γ_{i_l} and $w_{i_{l+1}}/\gamma_{i_{l+1}}$ is very small. In order to boost the performance of this algorithm unless $w_{i_l}/\gamma_{i_l} - w_{i_{l+j}}/\gamma_{i_{l+j}} > \epsilon$, where initially $j = 1$ and ϵ is a prespecified parameter, the value of β is set to the ratio of the next item in $\bar{I}$ by incrementing j by one. Enlarging the gap between Γ and β increases the number of intermediate rolls created by the enumeration procedure.

However, experiments reveal that the savings from the reduction in the number of calls to the enumeration procedure compensates for this increase.

4. Computational Experiments

The performance of the simultaneous column-and-row generation algorithm proposed for the TSCS problem is evaluated and the three algorithms designed to solve the row-generating PSP are compared using extensive computational tests on two sets of randomly generated data, I1 and I2, with parameters summarized in Table 1. Each data set consists of ten subsets for ten values of $|M|$ from 10 to 100 with increments of 10, and five instances are generated for each subset. The widths and demands of finished rolls are generated from integer uniform distributions also given in Table 1. For comparison purposes the parameters in I1 are chosen to be the same as those in Zak (2002a) and the parameters of I2 are selected to generate harder problems compared to I1. While enlarging W allows for 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.

Computations 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.

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)

Furthermore, an experiment is conducted where second stage cutting patterns $n \in N$ satisfying $s^{max} \geq \sum_{m \in M} \alpha_m B_{mn} + e^{min} \geq s^{min}$ are pre-enumerated. These patterns identify all possible intermediate rolls γ_i, $i \in I$ along with s_{min}. After the enumeration of these sets, the TSCS model (1)-(5) is solved directly by column generation without simultaneous increase in the rows. In fact, column generation invokes $y-$PSP, which is a knapsack problem involving set I, and $x-$PSP, which is solved only for s^{min} to generate second stage cutting patterns $n \in N$ that can appear in an optimal solution with a second stage trim larger than e^{min}. The preliminary experiments conducted to evaluate the efficacy of this *a priori* approach show that the large cardinality of I has a detrimental impact on the solution of $y-$PSP, which prolongs the termination of the column generation algorithm.

The average results of the proposed simultaneous column-and-row generation algorithm on I1 and I2 are presented in Tables 2 and 3, respectively. The results associated with the algorithms proposed for the row-generating PSP in Sections 3.2.1, 3.2.2 and 3.2.3 are given under the columns "2-Phase", "BP" and "Hybrid", respectively. All results on times are reported in seconds. 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) are reported. The instances for which the average time is larger than one hour are indicated by "-". For the generated data all three algorithms are run for both $e^{min} = 0$ and $e^{min} = 50$. In the Hybrid algorithm, the parameter ϵ is chosen to be $\epsilon = 10^{-10}$ for $e^{min} = 0$ and $\epsilon = 10^{-5}$ for $e^{min} = 50$, based on preliminary experimentation.

Tables 2 and 3 convey similar comparative results of the proposed algorithms. In terms of computational time, Hybrid outperforms 2-Phase by a small margin, 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 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, the instance sets I1 and I2 are solved 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. The solution time (Time) and the optimality gap (%Gap) of his algorithm averaged over the five replications are reported in the last columns of Tables 2 and 3. 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 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, SRMP is solved as an integer program whose optimal objective function value is denoted by z^{MIP}. 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 $|M| = 90$, 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, one dimensional TSCS problem is considered and an exact simultaneous column-and-row generation algorithm is presented for its solution. 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. To solve this PSP three methods are proposed. The first method is based on an intelligent enumeration of intermediate rolls that employs a simple dominance relation defined for knapsack problems. The second method is a branch-and-price algorithm that uses a lexicographic approach for the branch-and-bound and a novel procedure for column generation. The last algorithm, the hybrid algorithm, integrates the first two. It is observed from computational experiments that the hybrid algorithm is the best performing of all three. These experiments also reveal that the upper bound obtained by solving SRMP using a mixed integer programming (MIP) solver coincides with the optimal objective value of the original problem in most of the instances.

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.

Maher, S. J., 2015. Solving the integrated airline recovery problem using column-and-row generation. Transportation Science 50 (1), 216–239.

Mamer, J. W, McBride, R. D, 2000. A decomposition-based pricing procedure for large-scale linear programs: an application to the linear multicommodity flow problem. Management Science 46 (5), 693–709.

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., 1999. Exact solution of bin-packing problems using column generation and branch-and-bound. Annals of Operations Research 86, 629–659.

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.