# New model and heuristic solution approach for one-dimensional cutting stock problem with usable leftovers

Yaodong Cui[*], [1], Xiang Song[2], Yan Chen[1, 3], Yi-Ping Cui[3]

[1]College of Computer and Electronic Information, Guangxi University, Nanning 530004, China

[2]Department of Mathematics, University of Portsmouth, Portsmouth PO1 3HF, United Kingdom

[3] School of Business Administration, South China University of Technology, Guangzhou 510640, China

**Abstract:** In the one-dimensional cutting stock problem with usable leftovers (1DCSPUL), items of the current order are cut from stock bars to minimize material cost. Here, stock bars include both standard ones bought commercially and old leftovers generated in processing previous orders, and cutting patterns often include new leftovers that are usable in processing subsequent orders. Leftovers of the same length are considered to be of the same type. The number of types of leftovers should be limited to simplify the cutting process and reduce the storage area. This paper presents an integer programming model for the 1DCSPUL with limited leftover types and describes a heuristic algorithm based on a column-generation procedure to solve it. Computational results show that the proposed approach is more effective than several published algorithms in reducing trim loss, especially when the number of types of leftovers is limited.

**Keywords:** Cutting and packing; One-dimensional cutting stock; Column generation; Usable leftovers

## 1. Introduction

Cutting and packing problems have attracted much research interest (Beeker and Appa, 2015; Cui et al., 2015; Moreira de Carvalho et al., 2015). One such problem is the one-dimensional cutting stock problem (1DCSP). In the 1DCSP, a set of item types with specified lengths and demands is cut from linear objects (such as bars, tubes, and profiles; this paper only refers to *bars*) to manufacture various products. This problem appears in many industries such as the manufacturing of various vehicles, ships, doors, windows, and iron-made furniture. Good policies and algorithms are expected to improve material utilization in these situations.

In the one-dimensional multiple stock size cutting stock problem (1DMSSCSP), $m$ types of items with length $l_i$ and demand $d_i$ ($i = 1, \cdots, m$) are cut from $n$ types

[*] Corresponding author.
*E-mail address*: ydcui@263.net (Y. Cui)

of stock bars with length $L_j$ and supply $D_j$ ($j = 1, \cdots, n$) such that the material cost (total cost of bars used) is minimized. Studies have often set the cost of a bar as the bar length. The cost can also be set to include purchasing costs and other related expenses such as those incurred for shipping and handling. The 1DMSSCSP generalizes the one-dimensional single stock size cutting stock problem (1DSSSCSP) in which there is only one bar type ($n = 1$). As in previous studies, we assume that the lengths of both bars and items are integers.

The solution of the 1DCSP is a cutting plan that contains a set of different cutting patterns with corresponding frequencies (number of times a pattern is to be used). Each pattern uses a particular type of bar and includes some required items; the total length of the included items should not exceed the bar length. In the 1DCSP with usable leftovers (1DCSPUL), the residual length of a pattern is considered a leftover if it is not shorter than a threshold length and as trim-loss otherwise. Leftovers can be returned to stock for future use. The objective of solving the 1DCSPUL is often to minimize the material cost (difference between total cost of used bars and total value of generated leftovers, where if the cost of a bar is equal to the bar length, then the value of a leftover can also be set to be the leftover length).

Multiple bar types are useful for improving material utilization because they expand the solution space. When the number of standard bar types that can be bought commercially is very small, leftovers should be considered to improve material utilization.

Successive orders are processed when applying the policy of allowing leftovers (Cherri et al., 2013). A 1DCSPUL instance is solved for each order, with the objective of minimizing the material cost of the current order. This objective is reasonable because the information of the next orders is not known when the current order is being processed. In this paper, the following definitions for usable leftovers are used:

**Old leftovers**: Leftovers that are generated in the cutting processes of previous orders and that can be used as stock bars to produce items in the current order; also referred to as non-standard bar types.

**New leftovers**: Leftovers generated in the cutting process of the current order and returned to stock for future use.

**Novel leftovers**: New leftovers with lengths different from those of the old leftovers.

Usable leftovers of the same length belong to the same type (UL-type). The following symbols are used to characterize the basic 1DCSPUL for the current order:

Symbols corresponding to input data:

| | |
|---|---|
| $m$ | Number of item types. |
| $(l_1,\cdots,l_m)$ | Item lengths. |
| $(d_1,\cdots,d_m)$ | Item demands. |
| $n$ | Number of standard bar types. |
| $\lambda$ | Number of old UL-types (non-standard bar types) |
| $G$ | Total number of possible UL-types. |
| $(L_1,\ldots,L_{n+G})$ | Bar and leftover lengths. $(L_1,\ldots,L_n)$, $(L_{n+1},\ldots,L_{n+\lambda})$, and $(L_{n+\lambda+1},\ldots,L_{n+G})$ are the lengths of standard bar types, old UL-types, and novel UL-types, respectively. The standard bar types are arranged such that $(L_1 > \cdots > L_n)$. |
| $(D_1,\ldots,D_{n+\lambda})$ | Bar supplies. $(D_1,\ldots,D_n)$ and $(D_{n+1},\ldots,D_{n+\lambda})$ are the supplies of standard bar types and non-standard bar types, respectively. |
| $(p_1,\ldots,p_{n+G})$ | $p_i$ is the cost of a type-$i$ ($i=1,\cdots,n$) bar or the cost/value of a type-$i$ ($i=n+1,\cdots,n+G$) leftover. |

Symbols corresponding to generation of patterns:

| | |
|---|---|
| $K$ | Number of different cutting patterns. |
| $\alpha(k)$ | Bar type used by pattern $P_k$, $\alpha(k)\in(1,\cdots,n+\lambda)$, $k=1,\ldots,K$. |
| $\beta(k)$ | UL-type generated using $P_k$, $\beta(k)\in(0,n+1,\cdots,n+G)$. $\beta(k)=0$ denotes that no leftover is generated, in which case it is said that a null leftover is generated. |
| $L_0$ | Length of a null leftover ($L_0=0$). |
| $p_0$ | Value of a null leftover ($p_0=0$). |
| $a_{ik}$ | Number of type-$i$ items in pattern $P_k$, $i=1,\ldots,m$. |
| $P_k$ | Pattern $P_k=\left[\alpha(k),a_{1k},\ldots,a_{mk},\beta(k)\right]$, $k=1,\ldots,K$. The |

pattern uses bar type $\alpha(k)$ and generates UL-type $\beta(k)$. A pattern contains at most one leftover. It is assumed that patterns using non-standard bars cannot contain leftovers, i.e., $\beta(k)=0$ when $\alpha(k)>n$.

Symbols corresponding to decision variables:

| | |
|---|---|
| $x_k$ | Frequency of $P_k$ |
| $\mathbf{N}$ | Set of non-negative integers |

The 1DCSPUL can be formulated as the following integer programming (IP) model:

$$\min \sum_{k=1}^{K} \left(p_{\alpha(k)} - p_{\beta(k)}\right) x_k \tag{1-1}$$

$$\sum_{k=1}^{K} a_{ik} x_k \geq d_i, \quad i=1,\ldots,m \tag{1-2}$$

$$\sum_{k|\alpha(k)=j} x_k \leq D_j, \quad j=1,\ldots,n+\lambda \tag{1-3}$$

$$x_k \in \mathbf{N}, \quad k=1,\ldots,K \tag{1-4}$$

Formula (1-1) indicates that the objective is to minimize the material cost. Constraint (1-2) means that the item demands must be met. Constraint (1-3) means that the number of bars of each type used should not exceed the supply. Constraint (1-4) means that the frequency of each pattern is a non-negative integer.

For a particular pattern $P_k$, $k=1,\ldots,K$, the following constraint should be met:

$$\sum_{i=1}^{m} a_{ik} l_i + L_{\beta(k)} \leq L_{\alpha(k)} \tag{1-5}$$

It means that the total length of the included items and the new leftover should not exceed the bar length.

Leftovers often accumulate when successive orders are processed. To make the inventory level of leftovers reasonable, some researchers have proposed the use of an upper bound on the number of leftovers. Cui and Yang's (2010) model places an upper bound on the number of each UL-type. Arenales et al.'s (2015) model uses an overall upper bound on the total number of leftovers. This paper also uses an overall upper bound $N_{\mathrm{maxUL}}$ on the total number of leftovers. It means that $\sum_{j=1}^{\lambda} D_{n+j} \leq N_{\mathrm{maxUL}}$ holds when processing each order. We use 1DCSPUL_BN to denote that an overall upper Bound on the total Number of leftovers is used.

The number of possible UL-types may reach several thousands. In practical applications, a small number of UL-types are often expected in stock. This paper considers the constraint on the number of UL-types (referred to as the Bound on the number of leftover Types (BT) constraint). The related problem is referred to as the 1DCSPUL_BNT (1DCSPUL with Bounds on both the Number of leftovers and the number of leftover Types). Although the total number of possible UL-types $G$ is often large, the BT constraint requires at most $g$ ($g << G$) UL-types to be in stock, i.e., $\lambda \leq g$ holds when processing each order. The BT constraint is considered for the following reasons:

(1) Limitation of work area: Each new UL-type generated when processing the current order forms a stack (possibly in a bin). Therefore, the BT constraint is required for the available work area.

(2) Limitation of storage area. The storage area may be separated from the work area. It stores the UL-types accumulated in processing previous orders for future use. Therefore, the BT constraint is also required for the available storage area.

(3) Requirement of pattern reduction. Pattern reduction (reducing the number of patterns in the cutting plan) is necessary for some applications, where a setup cost is incurred for each new pattern. It is possible to have fewer patterns in the cutting plan when the frequency of a pattern using a UL-type is less limited. As the frequency of a pattern using a UL-type is restricted by the corresponding available supply, large average supply of the UL-types is helpful for pattern reduction. The average supply of the UL-types decreases with an increase in $g$. This means that a large $g$ value hinders pattern reduction, although it may be useful for improving material utilization. Both the setup and the material costs should be considered for selecting an appropriate $g$ value. Therefore, the BT constraint is also necessary for pattern reduction.

The existing approaches for the 1DCSPUL can deal with the BT constraint by specifying the lengths of the $g$ UL-types to be considered as inputs. Arenales et al.'s (2015) model allows all possible UL-types; however, in the experiment, only three UL-types are allowed, and their lengths (200, 300, and 400) are specified as inputs. Specifying the UL-types to stock often deteriorates the solution quality for the following two reasons:

(1) The number of possible combinations of $g$ UL-types is very large. Many other combinations may yield better solutions for the current order.

(2) The average solution quality may deteriorate seriously because the same combination is used for all orders.

This paper proposes an Integer Linear Programming (ILP) model for the 1DCSPUL_BNT. This model allows different combinations of UL-types when processing successive orders and determines the best combination for each current order to minimize the material cost. A two-phase algorithm is presented to solve the model heuristically. The computational results indicate that the model is more effective than existing ones in reducing trim-loss, manipulation, and storage area when the BT constraint is applied; the algorithm is also more effective than several published algorithms in solving the basic 1DCSPUL.

The remainder of this paper is organized as follows. Section 2 presents a literature review. Section 3 presents the formulation of the 1DCSPUL_BNT. Section 4 describes the solution approach. Section 5 presents the computational results. Finally, Section 6 presents the conclusions.

## 2. Literature review

The 1DCSP can be formulated as an ILP problem (Belov and Scheithauer, 2002; Valerio de Carvalho, 2005; Belov and Scheithauer, 2006) to minimize the material cost, where the decision variables are the pattern frequencies and should be integers. The linear relaxation (LR) is obtained by ignoring the integer constraint on the decision variables.

Two basic types of approaches are mainly used for the 1DCSP. The first type is column-generation-based approaches. These approaches can be either exact or heuristic in nature. Belov and Scheithauer (2002, 2006) and Valerio de Carvalho (2005) have proposed exact algorithms. Although their computation times for large instances may be long, approximate solutions can be obtained by using a time limit to stop the computation to select the best solution obtained so far. Heuristic algorithms usually solve the LR and apply rounding procedures to a generally non-integral solution (Gilmore and Gomory, 1963). They often lead to a proved optimum for the 1DCSPSSS, because the value obtained from rounding-up the LR solution value is often equal to that of the integer problem and is thus an effective lower bound (Scheithauer and Terno, 1995). The modified integer round-up property conjecture (Scheithauer and Terno, 1995) states that the gap between these values is always not larger than two. For the 1DCSPMSS, the combinations of different bar types help to improve material utilization; however, it is more difficult to find optimum solutions because of the complicated behaviour of the objective function (Belov and Scheithauer, 2002).

The second type of approach includes those based on the sequential heuristic procedure (SHP). These approaches generate each next pattern in the current cutting plan to fulfil some portion of the remaining demand and repeat until all demands are met. Multiple cutting plans can be generated using different parameters to select the best one. The SHP has the flexibility to consider practical restrictions and objectives such as open-stacks minimization (Belov and Scheithauer, 2007) and pattern reduction (Foerster and Wäscher, 2000; Yanasse and Limeira, 2006; Cui 2012).

In the 1DCSPUL_BNT addressed in this paper, successive orders are processed according to the arrival sequence. The demands of the items in each order must be met exactly. The items in the current order must be cut from available bars in stock (initially, only the standard bars bought from suppliers). Both the standard bars not used and the leftovers generated are carried on for use in processing the next orders. The inventory levels of standard bars are assumed to be constant, because they can be obtained from the warehouses of the suppliers with short lead time. The inventory level (total number) of leftovers may fluctuate; however, it should not exceed the specified upper bound. Although many UL-types can be considered, at most $g$ types are allowed in stock after completing each order. The objective is to minimize the material cost in completing each order. The ILP model proposed in this paper accurately depicts the 1DCSPUL_BNT. The proposed Two-Phase algorithm for the 1DCSPUL_BNT (TPBNT) solves the problem heuristically. The default range for the leftover lengths is determined based on the maximum standard bar length. The motivations and properties of this policy are described in the paragraphs below in comparison with algorithms reported in literature.

Some algorithms for the 1DCSPUL allow the generation of only one leftover in a cutting plan (Gradisar et al., 1999a; Gradisar et al., 1999b; Gradisar and Trkman, 2005). In contrast, the TPBNT allows the generation of multiple leftovers, where each pattern contains at most one leftover. This can often lead to better material utilization, although the cutting workload may be increased slightly because of the generation of multiple leftovers.

The TPBNT explicitly considers an upper bound on the number of leftovers, whereas some published algorithms (Scheithauer, 1991; Cherri et al., 2009; Cherri et al., 2013) do not. In processing successive orders, observing the upper-bound constraint is helpful to keep the inventory level of the leftovers under control. The upper bound is necessary when there is a budget limit on the total value of the leftovers in inventory.

Although Cui and Yang's (2010) and Arenales et al.'s (2015) models allow the generation of multiple leftovers and consider the upper bound on the number of leftovers, they do not consider the BT constraint. As described previously in Section 1, these two models can be used to solve the 1DCSPUL_BNT by specifying $g$ UL-types as inputs. The TPBNT optimizes the UL-types by explicitly considering the BT constraint. The computational results in this paper show that optimizing the UL-types is more effective than specifying the UL-types in reducing trim-loss, manipulation, and storage area.

## 3. Problem formulation

To formulate the 1DCSPUL_BNT, some additional symbols are defined as follows:

| | |
|---|---|
| $\delta_j$ | Flag for using type-*j* leftovers, $j = n+1,\cdots,n+\lambda$ . $\delta_j = 0$ if $\sum_{k\mid\alpha(k)=j} x_k = D_j$ ; $\delta_j = 1$ otherwise. |
| $\varepsilon_j$ | Flag for producing type-*j* leftovers, $j = n+1,\cdots,n+G$ . $\varepsilon_j = 1$ if $\sum_{k\mid\beta(k)=j} x_k > 0$ ; $\varepsilon_j = 0$ otherwise. |
| $N_{\text{maxUL}}$ | Upper bound on the total number of leftovers. |
| $g$ | Upper bound on the number of UL-types. |
| $\Psi$ | Large positive integer. |

The 1DCSPUL_BNT can be formulated as the following ILP model (2):

$$\min \sum_{k=1}^{K} \left( p_{\alpha(k)} - p_{\beta(k)} \right) x_k \tag{2-1}$$

$$\sum_{k=1}^{K} a_{ik} x_k \geq d_i \,, \quad i = 1,\ldots,m \tag{2-2}$$

$$\sum_{k\mid\alpha(k)=j} x_k \leq D_j \,, \quad j = 1,\ldots,n+\lambda \tag{2-3}$$

$$\sum_{j=n+1}^{n+\lambda} D_j + \sum_{k\mid\beta(k)>0} x_k - \sum_{k\mid\alpha(k)>n} x_k \leq N_{\text{maxUL}} \tag{2-4}$$

$$\delta_j \le D_j - \sum\nolimits_{k|\alpha(k)=j} x_k \le \Psi \delta_j ,\ \ j = n+1,\cdots,n+\lambda \tag{2-5}$$

$$\varepsilon_j \le \sum\nolimits_{k|\beta(k)=j} x_k \le \Psi \varepsilon_j ,\ \ j = n+1,\cdots,n+G \tag{2-6}$$

$$\sum\nolimits_{k|\alpha(k)=j} x_k \le D_j \left(1-\varepsilon_j\right),\ \ j = n+1,\cdots,n+\lambda \tag{2-7}$$

$$\lambda - \sum\nolimits_{j=n+1}^{n+\lambda} \left(1-\delta_j\right) + \sum\nolimits_{j=n+\lambda+1}^{n+G} \varepsilon_j \le g \tag{2-8}$$

$$\delta_j \in \{0,1\}\ ,\ j = n+1,\cdots,n+\lambda \tag{2-9}$$

$$\varepsilon_j \in \{0,1\},\ \ j = n+1,\cdots,n+G \tag{2-10}$$

$$x_k \in \mathbf{N},\ \ k = 1,\ldots,K \tag{2-11}$$

The number of constraints in (2-2)–(2-8) is $2+m+n+3\lambda+G$. The number of variables ($\delta_j$, $\varepsilon_j$, and $x_k$) in the model is $\lambda+G+K$.

Formula (2-1) indicates that the objective is to minimize the material cost. Constraint (2-2) means that the item demands must be met. Constraint (2-3) means that the number of bars of each bar type used should not exceed the supply. Constraint (2-4) means that the total number of leftovers should not exceed the upper bound after completing the current order, where $\sum\nolimits_{j=n+1}^{n+\lambda} D_j$ is the total number of old leftovers in stock before considering the order; $\sum\nolimits_{k|\beta(k)>0} x_k$, the number of new leftovers generated; and $\sum\nolimits_{k|\alpha(k)>n} x_k$, the number of old leftovers consumed. Constraints (2-5) and (2-6) are respectively used to validate the definitions of $\delta_j$ and $\varepsilon_j$.

Constraint (2-7) prohibits the simultaneous use and generation of the same UL-type. If leftovers of type-*j* are used $\left(\sum\nolimits_{k|\alpha(k)=j} x_k > 0\right)$, then $\varepsilon_j$ must be zero because of this constraint, thus prohibiting the generation of new leftovers of this type. If leftovers of type-*j* are generated $\left(\varepsilon_j = 1\right)$, then $\sum\nolimits_{k|\alpha(k)=j} x_k = 0$ because of this constraint, prohibiting the use of old leftovers of this type. The following illustration shows that these constraints are useful to simplify the cutting process, and they do not affect the objective value. Considering a specific old UL-type, there are two cases for generating $s$ ($s \ge 1$, integer) new leftovers and using $t$ ($t \ge 1$, integer) old leftovers of this UL-type: (1) If $s \ge t$, then the cutting plan can be adjusted by replacing the $t$ old leftovers with $t$ new leftovers without changing the objective value. With this adjustment, the cutting plan actually generates $s$ - $t$ new leftovers and uses zero leftovers of this UL-type. (2) If $t \geq s$, then the cutting plan can be adjusted by replacing $s$ old leftovers with $s$ new leftovers without changing the objective value. With this adjustment, the new cutting plan actually uses $t - s$ old leftovers and generates zero leftovers of this UL-type. In both cases, the cutting process of the new cutting plan is less complicated than that of the original cutting plan.

Constraint (2-8) indicates that after completing the current order, the number of remaining UL-types should not exceed $g$, where $\lambda - \sum_{j=n+1}^{n+\lambda}\left(1-\delta_j\right)$ is the number of remaining old UL-types, $\sum_{j=n+\lambda+1}^{n+G}\varepsilon_j$ is the number of novel UL-types generated. It is not necessary to consider non-novel new UL-types because of constraint (2-7). For example, if a non-novel leftover is generated, then the old leftovers of the same type cannot be used; therefore, the number of remaining UL-types is not affected.

For the current order, let $\lambda_1$ be the number of UL-types in stock at real time; then, the BT constraint guarantees that $\lambda_1 \leq g$ before and after the cutting process of the current order. It is clear that $\lambda_1 \leq g$ holds during the cutting process if the cutting plan does not contain novel leftovers. $\lambda_1 \leq g$ can also be guaranteed at any time in the cutting process if the cutting plan contains novel leftovers, as explained below.

The cutting process is divided into two stages. The patterns containing novel leftovers are cut in the second stage, and the others are cut in the first stage. It is clear that $\lambda_1 \leq g$ holds in the first stage because, initially, $\lambda_1 = \lambda \leq g$ and novel leftovers are not generated. In the second stage, the patterns only use standard bars (see the definition of $P_k$ in Section 1). $\lambda_1$ increases because old leftovers are not used and novel leftovers are generated. It reaches the maximum value at the end of the stage (end of cutting process). The maximum value does not exceed $g$ because of the BT constraint. Subsequently, $\lambda_1 \leq g$ holds at any time in the second stage.

## 4. Solution approach

### *4.1. Overview of TPBNT*

The ILP model can be rewritten as follows:

$$\min \sum_{k=1}^{K}\left(p_{\alpha(k)} - p_{\beta(k)}\right)x_k \quad (3\text{-}1)$$

$$\sum_{k=1}^{K} a_{ik}x_k \geq d_i\ ,\ \ i = 1,\ldots,m \quad (3\text{-}2)$$

$$\sum_{k|\alpha(k)=j} x_k \le D_j, \quad j=1,\ldots,n \tag{3-3}$$

$$\delta_j + \sum_{k|\alpha(k)=j} x_k \le D_j, \; j=n+1,\cdots,n+\lambda \tag{3-4}$$

$$\sum_{k|\beta(k)>0} x_k - \sum_{k|\alpha(k)>n} x_k \le N_{\text{maxUL}} - \sum_{j=n+1}^{n+\lambda} D_j \tag{3-5}$$

$$\Psi\delta_j + \sum_{k|\alpha(k)=j} x_k \ge D_j, \quad j=n+1,\cdots,n+\lambda \tag{3-6}$$

$$\sum_{k|\beta(k)=j} x_k - \varepsilon_j \ge 0, \quad j=n+1,\cdots,n+G \tag{3-7}$$

$$\Psi\varepsilon_j - \sum_{k|\beta(k)=j} x_k \ge 0, \quad j=n+1,\cdots,n+G \tag{3-8}$$

$$D_j\varepsilon_j + \sum_{k|\alpha(k)=j} x_k \le D_j, \quad j=n+1,\cdots,n+\lambda \tag{3-9}$$

$$\sum_{j=n+1}^{n+\lambda} \delta_j + \sum_{j=n+\lambda+1}^{n+G} \varepsilon_j \le g \tag{3-10}$$

$$\delta_j \in \{0,1\}, \; j=n+1,\cdots,n+\lambda \tag{3-11}$$

$$\varepsilon_j \in \{0,1\}, \quad j=n+1,\cdots,n+G \tag{3-12}$$

$$x_k \in \mathbf{N}, \quad k=1,\ldots,K \tag{3-13}$$

The ILP model refers to Model (3) from here on.

The original problem includes all items of the order. A residual problem includes some portion of the items. The TPBNT solves the ILP model in two phases. Residual problems are solved repeatedly by column-generation in Phase-1. The ILP model is solved using an optimization solver in Phase-2 over some patterns generated in Phase-1. Let $\Gamma$ be the set of patterns considered in Phase-2. Initially, let the remaining items include all items of the order. The procedure of the TPBNT is as follows:

```
Phase-1:
      While there are remaining item demands
1           Call SolveLPM to solve the LR of the current residual problem.
2           Call AdmitPats to admit some patterns into the Phase-1 solution.
3           Add all or some patterns generated in Step 1 to $\Gamma$.
Phase-2:
4     Use a MILP solver to solve the ILP model to obtain a Phase-2 solution.
5     Output the better one of the Phase-1 and Phase-2 solutions.
```

The SolveLPM procedure in Step 1 finds the optimal solution for the LR of the current residual problem by combining column-generation and dynamic programming techniques. The AdmitPats procedure in Step 2 admits some patterns in the LR solution into the Phase-1 solution and updates the remaining item demands and bar supplies accordingly. In Step 3, all patterns generated by solving the first three residual problems are admitted into $\mathit{\Gamma}$; for each other residual problem, only the patterns with positive frequency in the LR solution are admitted. This is useful to reduce the computational workload of Phase-2. Steps 1–3 are repeated until all demands are met; then, a MILP solver is used in Phase-2 to solve the ILP model over the patterns in $\mathit{\Gamma}$ to obtain a Phase-2 solution. Finally, in Step 5, the better one of the Phase-1 and Phase-2 solutions is chosen. The SolveLPM and AdmitPats procedures are described in detail in the following two sub-sections.

### *4.2.* SolveLPM *Procedure*

Let $r_i$ be the remaining demand of type-$i$ items, $i=1,\dots,m$. Let $R_j$ be the remaining number of type-$j$ bars, $j=1,\cdots,M$, where $M=n+G$, the first $n$ types are standard bars, the next $\lambda$ types are old leftovers, and the last $G-\lambda$ types are novel leftovers. For the initial residual problem, $r_i=d_i$, $i=1,\dots,m$; $R_j=D_j$ for $j=1,\cdots,n+\lambda$, $R_j=0$ for $j=n+\lambda+1,\cdots,n+G$. The ILP model of the residual problem is similar to that of the original problem, where $d_i$ should be replaced by $r_i$ and $D_j$, by $R_j$.

The procedure is described only for the first residual problem that is the same as the original problem. The column-generation method solves the LR through iteration. A new pattern should be generated in each run of the iteration. The number of constraints in (3-2)–(3-10) is $\eta=2+m+n+3\lambda+2G$. Let $\left(\pi_1,\cdots,\pi_\eta\right)$ be the duals of the current LR solution. The following bounded knapsack problem determines the optimal layout of the items on the largest bar length $L_1$, where $y_i$ is the number of type-$i$ items included:

$$F\left(L_1\right)=\max\left(\sum_{i=1}^{m}\pi_i y_i\right);\ \sum_{i=1}^{m}l_i y_i\le L_1;\ \ y_i\in\mathbf{N}\wedge y_i\le r_i,\ \ i=1,\cdots,m \tag{4}$$

The classical dynamic programming recursion (Kellerer et al., 2004) can be used to solve the problem. It has the all-capacity property: once the solution to the largest bar length $L_1$ is obtained, the solution to any bar length $L$ is also obtained,

$L = 0, 1, \cdots, L_1 - 1$. The complexity of the recursion is $O\left(L_1 \sum_{i=1}^{m} \left[\min\left(r_i, \lfloor L_1 / l_i \rfloor\right)\right]\right)$. This paper proposes the GetPattern procedure to generate the best pattern (new pattern). The best pattern has value $V_{best}$; initially, $V_{best} = 0$. Furthermore, a *combination* $(\alpha, \beta)$ corresponds to a pattern using a bar of type-$\alpha$ and generating a new leftover of type-$\beta$. No leftover is generated when $\beta = 0$.

```
GetPattern Procedure:
1   Solve Model (4) to obtain $F(L)$ and the related item layouts,
2   $L = 0, 1, \cdots, L_1$.
3   For each $\alpha \in \{1, \cdots, n + \lambda\}$ and $R_\alpha > 0$
4       Consider combination $(\alpha, 0)$ to improve the best pattern.
5       If $\alpha \le n$ then
6           For $\beta = n + 1, \cdots, n + G$
7               Consider combination $(\alpha, \beta)$ to improve the best pattern
    Return the best pattern.
```

In this procedure, Step 3 considers patterns that do not contain leftovers. Steps 4–6 consider patterns using a bar of type-$\alpha$ and generating a new leftover of type-$\beta$, where $\alpha \le n$ in Step 4 indicates that a new leftover can be generated only when the pattern uses standard bar types. Let $V$ be the value of a pattern determined in Step 3 or 6. The following two paragraphs illustrate how the $V_{best}$ value and the corresponding best pattern are achieved, which is followed by the termination criterion.

For combination $(\alpha, \beta)$, the bar length $L$ used to pack the items is equal to $L_\alpha - L_\beta$. If $F(L) = 0$, then the combination is discarded. Otherwise, the current pattern $P$ is completely determined as $P = (\alpha, z_1, \cdots, z_m, \beta)$, because $(z_1, \cdots, z_m)$, the numbers of items included, have been determined in Step 1. If the current pattern is introduced into the LR and the pattern frequency is denoted as $x$, then the coefficients of $x$ in the constraints of the ILP model are also known and are denoted as $Q = \left[q_1, \cdots, q_\eta\right]^T$, where $q_j$ is the coefficient in the j-*th* constraint. The value of pattern $P$ is $V = \sum_{j=1}^{\eta} \pi_j q_j$. If $V > V_{best}$, then let $V_{best} = V$, and record pattern $P$ as the best pattern.

After calling the GetPattern procedure, the best pattern is introduced into the LR, and the iteration continues if $V_{best} > p_\alpha - p_\beta$; otherwise, the iteration terminates.

To guarantee the convergence of the AdmitPats procedure described in the next sub-section, the LR of the residual problem is solved in two stages using the SolveLPM procedure. New leftovers are not allowed in the first stage until the solution cannot be improved further (when $V_{best} = p_\alpha - p_\beta$); then, the pattern of maximum frequency in the first-stage solution is recorded as a *special pattern* that may be used in the AdmitPats procedure. After that, the iteration of the second stage starts by allowing new leftovers.

## *4.2.* AdmitPats *Procedure*

In Step 1 of the TPBNT, the SolveLPM procedure obtains the optimal solution of the current LR, where the frequencies of the patterns in the optimal solution are positive and often fractional (patterns of zero frequencies are not considered). In Step 2 of the TPBNT, the AdmitPats procedure is called to admit some of the patterns into the Phase-1 solution to meet the demands of some remaining items.

Let $N_{curUL}$ be the total number of leftovers and $g_{cur}$, the number of UL-types currently in stock. $N_{curUL} = \sum_{j=n+1}^{n+G} R_j$ ; $g_{cur} = \sum_{j=n+1}^{n+G} \sigma_j$ , where $\sigma_j = 1$ if $R_j > 0$ , $\sigma_j = 0$ if $R_j = 0$ . Let $P = [\alpha, z_1, \cdots, z_m, \beta]$ be the current pattern under consideration. Let $f$ be the frequency of $P$ in the optimal LR solution. The integer frequency is $f_0$ if the pattern is admitted. The steps in the AdmitPats procedure are as follows:

Step 1. Sort the patterns of the optimal LR solution according to the non-increasing order of their frequencies. Let $n_{admitted} = 0$ .

Step 2. Consider the patterns one by one using Steps 3–11. Go to Step 12 when all patterns are considered.

Step 3. Go to Step 2 if $R_\alpha = 0$, because the related bar type is used up. Go to Step 4 otherwise.

Step 4. Let $f_0 = \max\{1, \lfloor f \rfloor\}$ .

Step 5. If $f_0 > R_\alpha$ then let $f_0 = R_\alpha$ to consider the bar supply.

Step 6. Let $k = \min\{\lfloor r_i / z_i \rfloor | i : z_i > 0 \wedge i \in I\}$, where $I = \{1, \cdots, m\}$ . If $k < f_0$ , then let $f_0 = k$ to avoid surplus items. Go to Step 9 if $\beta = 0$.

Step 7. Admitting $P$ will generate $f_0$ new leftovers because $\beta > 0$. If $N_{curUL} + f_0 > N_{\max UL}$, then let $f_0 = N_{\max UL} - N_{curUL}$ to consider the upper bound of the total number of leftovers.

Step 8. If $R_\beta = 0$, admitting $P$ will generate a novel leftover type; then, let $f_0 = 0$ if $g_{cur} = g$ to guarantee that the upper bound of the UL-types is not exceeded.

Step 9. If $f_0 = 0$, then go to Step 2 to consider the next pattern.

Step 10. Admit $P$ into the Phase-1 solution. Set $n_{admitted} = n_{admitted} + 1$. Set $R_\alpha = R_\alpha - f_0$ to update the bar supply. Set $r_i = r_i - f_0 z_i$ to update the remaining demands, $i \in I$. Update $N_{curUL}$ and $g_{cur}$ correspondingly.

Step 11. If ($f \geq 1 \wedge n_{admitted} = \max\{1, \lfloor m/3 \rfloor\}$) or ($f < 1 \wedge n_{admitted} = 1$), then terminate the procedure; otherwise, go to Step 2 to consider the next pattern.

Step 12. If $n_{admitted} = 0$, then admit the special pattern with $f_0 = 1$.

In Step 11, at most $\max\{1, \lfloor m/3 \rfloor\}$ patterns will be admitted in each call of the procedure; when the largest frequency of patterns in the LR solution is smaller than 1, at most one pattern will be admitted in each call.

This procedure guarantees that at least one pattern is admitted because of Step 12, where the special pattern uses a bar type with positive supply and does not contain any leftover; its frequency can be set to be 1 without producing surplus items because constraint $z_i \leq r_i$ is guaranteed by Model (4), $i \in I$.

## 5. Computational results

The TPBNT was coded in C# and executed on a Dell computer (Inspiron 3847, Intel Core i5-4440 3.3 GHz CPU, 8 GB RAM), and the MILP solver CPLEX Version 12.5 was used as the optimization engine. The time limit for the computation in Phase-2 was 5 s.

Although the TPBNT is designed to solve the 1DCSPUL_BNT, it can also solve the 1DCSPUL and the 1DCSPUL_BN by using infinite bounds. In the following sub-sections, first, parameter values are introduced, and then, several sets of benchmark instances are used to demonstrate the effectiveness of the TPBNT.

## *5.1. Parameters*

Let the threshold of leftovers be $\eta L_1$, $0 < \eta < 1$ and $\eta = 0.5$ by default. Let $l_{\min}$ be the minimum length of the items and $L_{\mathrm{maxUL}}$, the maximum length of the new UL-types. Naturally, we have $L_{\mathrm{maxUL}} \geq \eta L_1$ and $L_{\max UL} = L_1 - l_{\min}$. Considering that the number of constraints in (2-6) may be too large if any integer in $[\eta L_1, L_{\mathrm{maxUL}}]$ represents a valid leftover length, we use

$$\eta L_1 + (j-1)\delta_{UL}, \quad j = 1, \cdots, k$$

to represent the possible lengths of the new UL-types, where $\delta_{UL}$ is a positive integer denoting the difference of two adjacent leftover lengths ($\delta_{UL} = 1$ by default), and $k = 1 + \lfloor (L_{\mathrm{maxUL}} - \eta L_1)/\delta_{UL} \rfloor$ is the maximum number of the new UL-types. As a new UL-type and an old UL-type can be combined into one type if they have the same length, we have $G \leq \lambda + k$. The UL-types in the ILP model are indexed such that old UL-types have smaller indices than novel UL-types.

Let $\left\lceil \sum_{i=1}^{m} l_i d_i / L_1 \right\rceil$ be the estimated number of the longest bars required to produce the items of an order. Set the upper bound of the total number of leftovers as $N_{\mathrm{maxUL}} = \left\lceil \sum_{i=1}^{m} l_i d_i / L_1 \right\rceil$ by default.

If the lead time for the replenishment of standard bars is negligible, it is not necessary to hold safety stock to avoid shortages of standard bars. Leftovers are held in stock, and their capital cost should be considered. We use parameter $\gamma$ to denote the capital cost of each unit-length of the leftovers. The costs of the standard bars are set to be their lengths, namely, $p_i = L_i$, $i = 1, \cdots, n$. The costs of non-standard bars or the values of leftovers are set to be $(1-\gamma)$ multiple of their lengths, namely, $p_i = (1-\gamma) L_i$, $i = n+1, \cdots, n+G$. The default parameter is $\gamma = 0.0001$. By default, the objective of the TPBNT may be approximately seen as minimizing the net bar length (difference between the total length of the bars used and that of the new leftovers generated) because of the small $\gamma$ value.

## *5.2. Processing orders independently*

Cherri et al.'s (2009) and Cui and Yang's (2010) algorithms were tested by processing the orders independently, where the leftovers generated in processing an order are not used in processing the next orders. Cherri et al.'s (2009) algorithms solve the basic 1DCSPUL. Although Cui and Yang's (2010) algorithm can solve the 1DCSPUL_BN, it was tested only for solving the basic 1DCSPUL.

Sixteen classes of instances generated in Cherri et al. (2009) are used. Each class contains 20 instances. For each instance, the number of item types $m$ is in the range [10, 40], number of standard bar types is $n=2$ ($L_1 = 1100$, $L_2 = 1000$, $D_1 = D_2 = 100$), and number of non-standard bar types $\lambda$ is in the range [3, 7]. The lengths of the non-standard bar types are smaller than 1000, and the available numbers are in the range [1, 10]. More detailed descriptions of the instances are available in Cherri et al. (2009).

To consider the same constraints imposed on the feasible solutions, the TPBNT solved the instances by using $N_{\text{maxUL}} = g = +\infty$ and allowing the generation of all leftover lengths not smaller than the threshold $\left(\sum_{i=1}^{m} l_i\right)\Big/m$ used in the literature. Considering that leftover reduction (reducing the number of leftovers in stock) is often desirable, $\gamma = 0.005$ is used by the TPBNT to consider this auxiliary objective. The computational results are shown in Table 1, where "leftover reduction" is the number of leftovers reduced, which is equal to the difference between the number of non-standard bars used and the number of new leftovers generated. Both the trim loss and the leftover reduction values are averaged over all instances. The TPBNT yielded the smallest trim loss and the largest leftover reduction, indicating that it is effective in solving the instances. The average computation time of an instance is 15.4 seconds for the TPBNT. It can be seen as reasonable for practical applications. The average computation time is 22.6 seconds for the best algorithm in Cherri et al. (2009), on a computer of 3GHz CPU and 2GB RAM; it is 1.6 seconds for the algorithm in Cui and Yang (2010), on a computer of Intel Core 2 Duo CPU E4500 2.20GHz and 1GB RAM.

Table 1. Results for processing orders independently

| | TPBNT | Cui and Yang (2010) | Cherri et al. (2009) |
|---|---|---|---|
| Trim loss | 2.3 | 9.8 | 11.5 |
| Leftover reduction | 22.7 | 22.4 | 2.7 |

When the default value $\gamma = 0.0001$ is used, the average trim loss and leftover reduction values of the TPBNT are respectively 0 and 21.1. In other words, the TPBNT solved all instances to optimality if trim loss minimization is the only objective to consider.

## *5.3. Processing orders dependently*

Cherri et al.'s (2013) algorithms also solve the basic 1DCSPUL. They were tested on orders that were processed successively using a computer with a Pentium IV 3.2 GHz CPU and 2 GB RAM. The leftovers generated in processing the current order can be used in processing the next orders.

There are two sets of instances. Each set contains 20 groups, and each group consists of 12 orders/instances. For each instance, $n = 2$, $L_1 = 1100$, $L_2 = 1000$, and $D_1 = D_2 = +\infty$ are assumed; the number of item types $m$ is in the range [20, 40]; the length $l_i$ of an item type is in the range [0.01$L$, 0.25$L$] for the first set and in the range [0.01$L$, 0.4$L$] for the second set, where $L = (L_1 + L_2)/2$; the demand $d_i$ of an item type is in the range [200, 500] for the first 10 item types in each instance and in the range [1, 10] for other item types. Furthermore, the lengths of the first 10 item types are the same for all instances. The motives and detailed descriptions about the generation of the instances are available in Cherri et al. (2013).

The 12 orders in a group are processed successively. This is equivalent to 12 successive periods in each of which an order must be processed. Cherri et al. (2013) used discounted costs of leftovers to reduce the inventory level of leftovers. However, this approach cannot constrain the total number of leftovers and number of UL-types. The TPBNT uses the following parameters to make fair comparisons: $N_{\max UL} = g = +\infty$. Considering that Cherri et al. (2013) used the total number of leftovers in stock after 12 periods as an indicator to evaluate the solution quality, the TPBNT uses $\gamma = 0.05$ in the last period and the default value in other periods; default values are used for the other parameters. Using a larger $\gamma$ value in the last period is useful to reduce the total number of leftovers in stock after 12 periods.

Table 2 summarizes the computational results, where 'no. of leftovers' denotes the number of leftovers after 12 periods (averaged over 20 groups in the set) and $t$, the computation time; both the computation time and the trim loss are averaged over all instances in the set. The RGRLP is Cherri et al.'s (2013) best algorithm. The TPBNT yielded better values for all parameters. For the instances of the first set, the trim loss was 17.46% (3.3/18.9 = 17.46%) of that of the RGRLP; for the instances of the second set, the trim loss was only 0.33% (5.1/1545.1 = 0.33%) of that of the RGRLP. Therefore, the TPBNT is more effective than the RGRLP in solving these two sets of basic 1DCSPUL instances.

Table 2. Results for processing orders dependently

| | First set | | Second set | |
|---|---|---|---|---|
| | TPBNT | RGRLP | TPBNT | RGRLP |
| Trim loss | 3.3 | 18.9 | 5.1 | 1545.1 |
| No. of leftovers | 0.95 | 1.1 | 0.65 | 1.2 |
| $t$ (s) | 8.0 | 37.8 | 6.2 | 36.5 |

*5.4. Considering BT constraint*

The 18 classes of instances often used in solving the 1DCSP with pattern reduction (Foerster and Wäscher, 2000; Yanasse and Limeira, 2006; Cui, 2012) are used. Each class contains 100 instances whose properties are summarized in Table 3, where $\overline{d}$ denotes the average demand of an item type and $L_{items}$, the total length of the items in a class. There is one standard bar type of length 1000. Only the first 20 instances in each class were solved by the TPBNT to reduce the total computation time.

Table 3. Properties of the instances

| Class | $m$ | $l_i$ range | $\overline{d}$ | $L_{items}$ | Class | $m$ | $l_i$ range | $\overline{d}$ | $L_{items}$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 10 | [10, 200] | 10 | 222344 | 10 | 20 | [10, 800] | 100 | 17492208 |
| 2 | 10 | [10, 200] | 100 | 2160857 | 11 | 40 | [10, 800] | 10 | 3223629 |
| 3 | 20 | [10, 200] | 10 | 448767 | 12 | 40 | [10, 800] | 100 | 32158806 |
| 4 | 20 | [10, 200] | 100 | 4511971 | 13 | 10 | [200, 800] | 10 | 1038896 |
| 5 | 40 | [10, 200] | 10 | 836104 | 14 | 10 | [200, 800] | 100 | 10192316 |
| 6 | 40 | [10, 200] | 100 | 8342336 | 15 | 20 | [200, 800] | 10 | 2090495 |
| 7 | 10 | [10, 800] | 10 | 861178 | 16 | 20 | [200, 800] | 100 | 20980993 |
| 8 | 10 | [10, 800] | 100 | 8353123 | 17 | 40 | [200, 800] | 10 | 3987594 |
| 9 | 20 | [10, 800] | 10 | 1739182 | 18 | 40 | [200, 800] | 100 | 39817014 |

As mentioned previously in the literature review, existing models (Cui and Yang, 2010; Arenales et al., 2015) can be used to solve the 1DCSPUL_BNT by specifying $g$ UL-types as inputs, whereas the TPBNT optimizes the UL-types by explicitly considering the BT constraint. The TPBNT also specifies the UL-types if $\lambda = G = g$ is used. The following two cases are compared:

Case A: Optimizing method — Optimize the UL-types by using default $(\eta, N_{\max UL}, \gamma)$ values, $\delta_{UL} = 10$, and specified $g$ value.

Case B: Specifying method — Specify the UL-types by using $\lambda = G = g$, default $(\eta, N_{\max UL}, \gamma)$ values, and $\delta_{UL}$ determined from the following expression, where $l_{\min}^{0}$ is the average minimum item length of all instances in a class:

$$L_{\max UL} = L_1 - l_{\min}^0 , \quad \delta_{UL} = \lfloor (L_{\max UL} - \eta L_1)/(g-1) \rfloor$$

For example, $l_{\min}^0 = 23$ for Class 1; when $\lambda = G = g = 4$, $L_{\max UL} = L_1 - l_{\min}^0 = 977$, and $\delta_{UL}$ is determined as $\delta_{UL} = \lfloor (977 - 0.5 \times 1000)/(4-1) \rfloor = 159$. The specified leftover lengths are respectively 500, 659, 818, and 977.

First, we demonstrate that when the same $g$ value is used by both methods, using the optimizing method will lead to significant reduction in trim-loss.

Table 4 shows detailed comparisons between the results of Cases A and B for $g =$ 6. For Case A, $T_L^A$ is the total trim-loss length of a class, and $U_A$ is the material utilization determined as $U_A = L_{items} \big/ \left( L_{items} + T_L^A \right) \times 100$. $T_L^B$ and $U_B$ are those for Case B. $\Delta U = U_B - U_A$. The total trim-loss length of Case A is smaller than that of Case B in each class. $T_L^A / T_L^B$ values range from 8.57% to 97.65%. The average value of 58.98% is obviously smaller than 100%. The average $\Delta U$ value in the last row of the table indicates that optimizing the UL-types can lead to 1.05% increase in material utilization.

Table 4. Effect of optimizing the UL-types ( $g = 6$)

| Class | $T_L^A$ | $T_L^B$ | $T_L^A / T_L^B$ (%) | $U_A$% | $U_B$% | $\Delta U$ |
|---|---|---|---|---|---|---|
| 1 | 96 | 466 | 20.60 | 99.96 | 99.79 | 0.17 |
| 2 | 133 | 693 | 19.19 | 99.99 | 99.97 | 0.03 |
| 3 | 93 | 758 | 12.27 | 99.98 | 99.83 | 0.15 |
| 4 | 99 | 674 | 14.69 | 100.00 | 99.99 | 0.01 |
| 5 | 96 | 1120 | 8.57 | 99.99 | 99.87 | 0.12 |
| 6 | 114 | 970 | 11.75 | 100.00 | 99.99 | 0.01 |
| 7 | 107702 | 122322 | 88.05 | 88.88 | 87.56 | 1.32 |
| 8 | 1164557 | 1192641 | 97.65 | 87.76 | 87.51 | 0.26 |
| 9 | 109278 | 155990 | 70.05 | 94.09 | 91.77 | 2.32 |
| 10 | 1293152 | 1689352 | 76.55 | 93.12 | 91.19 | 1.92 |
| 11 | 138231 | 234213 | 59.02 | 95.89 | 93.23 | 2.66 |
| 12 | 1062974 | 2092902 | 50.79 | 96.80 | 93.89 | 2.91 |
| 13 | 176604 | 195604 | 90.29 | 85.47 | 84.16 | 1.32 |
| 14 | 1641164 | 1846589 | 88.88 | 86.13 | 84.66 | 1.47 |
| 15 | 274685 | 311301 | 88.24 | 88.39 | 87.04 | 1.35 |
| 16 | 2770137 | 3147147 | 88.02 | 88.34 | 86.96 | 1.38 |
| 17 | 265206 | 299069 | 88.68 | 93.76 | 93.02 | 0.74 |
| 18 | 2500986 | 2829986 | 88.37 | 94.09 | 93.36 | 0.73 |

| | | |
|---|---|---|
| Avg. | 58.98 | 1.05 |

Table 5 lists the $T_L^A/T_L^B$ and $\Delta U$ values for different $g$ values, where the data in the columns for $g=6$ are copied from Table 4. Observations similar to those of Table 4 can be obtained from the last row of the table, where the $T_L^A/T_L^B$ values vary from 54.93% to 74.40% and the average is 63.47%; the $\Delta U$ values vary from 0.60% to 1.44% and the average is 1.07%. The TPBNT yields 36.53% (100% - 63.47% = 36.53%) reduction in trim-loss and 1.07% increase in material utilization, indicating that optimizing the UL-types is more effective than specifying them.

Table 5. Summary of the effects of optimizing UL-types

| Class | $T_L^A/T_L^B$ | | | | | $\Delta U$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $g=2$ | 4 | 6 | 8 | 10 | $g=2$ | 4 | 6 | 8 | 10 |
| 1 | 45.66 | 5.74 | 20.60 | 20.17 | 20.56 | 1.10 | 0.63 | 0.17 | 0.17 | 0.17 |
| 2 | 12.77 | 13.16 | 19.19 | 33.67 | 55.65 | 0.19 | 0.04 | 0.03 | 0.01 | 0.00 |
| 3 | 45.23 | 5.91 | 12.27 | 14.98 | 17.85 | 0.69 | 0.33 | 0.15 | 0.12 | 0.10 |
| 4 | 67.49 | 9.41 | 14.69 | 25.19 | 17.93 | 0.03 | 0.02 | 0.01 | 0.01 | 0.01 |
| 5 | 103.27 | 6.05 | 8.57 | 13.93 | 20.08 | -0.02 | 0.18 | 0.12 | 0.07 | 0.05 |
| 6 | 102.18 | 11.38 | 11.75 | 20.36 | 24.68 | 0.00 | 0.01 | 0.01 | 0.01 | 0.00 |
| 7 | 85.89 | 83.72 | 88.05 | 87.06 | 95.75 | 1.76 | 1.96 | 1.32 | 1.40 | 0.42 |
| 8 | 96.45 | 94.99 | 97.65 | 92.89 | 96.36 | 0.42 | 0.57 | 0.26 | 0.80 | 0.40 |
| 9 | 74.28 | 61.64 | 70.05 | 72.04 | 75.56 | 2.41 | 3.41 | 2.32 | 2.00 | 1.66 |
| 10 | 76.86 | 70.60 | 76.55 | 80.08 | 77.34 | 2.34 | 2.87 | 1.92 | 1.51 | 1.72 |
| 11 | 74.64 | 66.84 | 59.02 | 61.24 | 61.33 | 1.66 | 2.18 | 2.66 | 2.25 | 2.25 |
| 12 | 73.19 | 58.92 | 50.79 | 53.77 | 56.17 | 1.59 | 2.46 | 2.91 | 2.48 | 2.26 |
| 13 | 85.59 | 83.77 | 90.29 | 97.36 | 98.88 | 2.21 | 2.38 | 1.32 | 0.34 | 0.14 |
| 14 | 85.63 | 83.89 | 88.88 | 98.44 | 102.85 | 2.14 | 2.27 | 1.47 | 0.19 | -0.34 |
| 15 | 85.59 | 87.57 | 88.24 | 94.24 | 96.16 | 1.93 | 1.49 | 1.35 | 0.62 | 0.40 |
| 16 | 79.70 | 86.94 | 88.02 | 90.49 | 96.40 | 2.81 | 1.59 | 1.38 | 1.06 | 0.38 |
| 17 | 74.29 | 80.79 | 88.68 | 85.73 | 90.39 | 2.23 | 1.47 | 0.74 | 0.95 | 0.60 |
| 18 | 70.41 | 77.39 | 88.37 | 87.20 | 89.88 | 2.52 | 1.69 | 0.73 | 0.81 | 0.61 |
| Avg. | 74.40 | 54.93 | 58.98 | 62.71 | 66.32 | 1.44 | 1.42 | 1.05 | 0.82 | 0.60 |

It is known from Table 3 that the average item length of Classes 7–12 is larger than that of Classes 1–6. The results in Table 5 show that the $\Delta U$ values are larger in Classes 7–12 than in Classes 1–6, indicating that the effect of optimizing the UL-types is generally stronger when the average item length is larger.

Table 6 shows the total trim-loss length $T_L^A$ for each combination of (Class, $g$).

The corresponding total net bar length $L_{items} + T_L^A$ and material utilization $U_A$ can be obtained indirectly from $T_L^A$ in this table and $L_{items}$ in Table 3. They should be useful to future researchers for comparing their algorithms with the TPBNT.

Table 6. Total trim-loss lengths for Case A

| Class | $g = 2$ | $g = 4$ | $g = 8$ | $g = 10$ |
|---|---|---|---|---|
| 1 | 2126 | 86 | 96 | 96 |
| 2 | 593 | 133 | 133 | 133 |
| 3 | 2593 | 93 | 93 | 93 |
| 4 | 2719 | 99 | 99 | 99 |
| 5 | 5056 | 96 | 96 | 96 |
| 6 | 3744 | 114 | 114 | 114 |
| 7 | 122672 | 113712 | 103752 | 103392 |
| 8 | 1251277 | 1189667 | 1146127 | 1152107 |
| 9 | 145818 | 112068 | 102828 | 101738 |
| 10 | 1674152 | 1459152 | 1232652 | 1190212 |
| 11 | 179031 | 159441 | 127031 | 127061 |
| 12 | 1564284 | 1255764 | 1012224 | 1013204 |
| 13 | 198664 | 180604 | 176604 | 176604 |
| 14 | 1872464 | 1676304 | 1641164 | 1684164 |
| 15 | 327825 | 288405 | 272165 | 270865 |
| 16 | 3162007 | 2936677 | 2743207 | 2732687 |
| 17 | 305616 | 288906 | 260106 | 258696 |
| 18 | 2807986 | 2666986 | 2499486 | 2447806 |

Table 6 shows that the total trim-loss length generally decreases with increasing $g$ value, but occasionally increases slightly. This is not surprising because the TPBNT is a heuristic algorithm.

Second, we demonstrate that using the optimizing method can significantly reduce the manipulation and storage area without increasing the trim-loss.

Considering that comparing the results of all classes is tedious, the total trim-loss of all classes is used. The total trim-loss is 11505307 for the optimizing method with $g$= 6. Table 7 shows the total trim-losses for the specifying method with different $g$ values, where $g_1$ denotes the maximum number of UL-types actually stored in stock.

Table 7. Total trim-loss length for a specific $g$ value (specifying method)

| $g$ | 6 | 12 | 18 | 23 |
|---|---|---|---|---|
| $g_1$ | 6 | 8 | 10 | 11 |
| Trim-loss | 14121797 | 12572982 | 11998425 | 11577591 |

Assume that bins are used to hold the leftovers and that each bin is used for holding one UL-type. The necessary storage area increases with the number of bins used. The following observations can be made from the results:

(1) Using the optimizing method can significantly reduce the storage area. The specifying method requires 11 bins to yield a trim-loss of 11577591, whereas the optimizing method uses only 6 bins to yield approximately the same trim-loss.

(2) Using the optimizing method can reduce the manipulation of leftovers because of the small number of UL-types in stock.

(3) For the specifying method, although $g_1$ the maximum number of UL-types actually stored in stock is generally smaller than $g$, it may reach $g$ in the worst case, if additional instances are generated and solved. The number of bins reserved for holding the leftovers should be equal to $g$, in response to the worst case.

## 6. Conclusions

For practical applications, it is often expected to keep only a small number of UL-types in stock. Existing approaches can limit the number of UL-types by specifying their lengths as inputs. The proposed approach explicitly considers the constraint on the number of UL-types and determines the UL-types to stock through optimization. Compared with the specifying method, optimizing the UL-types often leads to significant reduction in trim-loss. For the benchmark instances tested in Section 5.4, optimizing the UL-types leads to more than 30% reduction in trim-loss.

For practical applications, reductions in manipulation and storage area are also expected. The computational results in Section 5.4 show that to yield the same total trim-loss length, the optimizing method requires a significantly lesser number of UL-types in stock than the specifying method. This observation indicates that the TPBNT is more effective than published approaches to reduce manipulation and storage area.

Although the proposed TPBNT algorithm is designed to solve the 1DCSPUL_BNT, it can also solve the basic 1DCSPUL by using infinite bounds. The computational results in Sections 5.2 and 5.3 show that the TPBNT outperforms published algorithms in solving all sets of basic 1DCSPUL instances.

In future research, we may extend the TPBNT to consider other objectives such as setup and open stack minimization (Belov and Scheithauer, 2007).

**Acknowledgments**

This research is part of Projects 71371058 and 61363026 supported by the National Natural Science Foundation of China.

## References

Arenales, M., Cherri, A., Nascimento, D. N., Vianna, A. (2015). A new mathematical model for the cutting stock/leftover problem. Pesquisa Operacional, 2015, 35, 509-522

Beeker, K. H., Appa, G. (2015). A heuristic for the Minimum Score Separation Problem, a combinatorial problem associated with the cutting stock problem. Journal of the Operational Research Society, 66, 1297-1311

Belov, G., Scheithauer, G. (2002). A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. European Journal of Operational Research, 141, 274-294

Belov, G., Scheithauer, G. (2006). A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. European Journal of Operational Research, 171, 85-106

Belov, G., Scheithauer, G. (2007). Setup and open-stacks minimization in one-dimensional stock cutting. INFORMS Journal on Computing, 19, 27-35

Cherri, A. C., Arenales, M. N., Yanasse, H. H. (2009). The one dimensional cutting stock problem with usable leftover - a heuristic approach. European Journal of Operational Research, 196, 897-908

Cherri, A. C., Arenales, M. N., Yanasse, H. H. (2013). The usable leftover one-dimensional cutting stock problem - a priority-in-use heuristic. International Transactions in Operational Research, 20, 189-199

Cui, Y. (2012). A CAM system for one-dimensional stock cutting. Advances in Engineering Software, 47, 7-16

Cui, Y., Yang, Y. (2010). A heuristic for the one−dimensional cutting stock problem with usable leftover. European Journal of Operational Research, 204, 245-250

Cui, Y-P., Cui, Y., Tang, T., Hu, W. (2015). Heuristic for constrained two-dimensional three-staged patterns. Journal of the Operational Research Society, 66, 647-656

Foerster, H., Wäscher, G. (2000). Pattern reduction in one dimensional cutting stock problems. International Journal of Production Research, 38, 1657-1676

Gilmore, P. C., Gomory, R. E. (1963). A linear programming approach to the cutting-stock problem (Part II). Operations Research, 11, 863-887

Gradisar, M., Kljajic, M. Resinovic, C. (1999a). A sequential heuristic procedure for one-dimensional cutting. European Journal of Operational Research, 114, 557-568

Gradisar, M., Resinovic, C., Kljajic, M. (1999b). A hybrid approach for optimization of one-dimensional cutting, European Journal of Operational Research 119 (1999) 719-728

Gradisar, M., Trkman, P. (2005). A combined approach to the solution to the general one-dimensional cutting stock problem. Computers & Operations Research, 32, 1793-1807

Kellerer, H., Pferschy, U., Pisinger, D. (2004). Knapsack Problems. Springer Verlag, Berlin.

Moreira de Carvalho, M. A., Soma, N. Y. (2015). A breadth-first search applied to the minimization of the open stacks. Journal of the Operational Research Society, 66, 936-946

Scheithauer, G. (1991). A note on handling residual lengths. Optimization, 22, 461-466

Scheithauer, G., Terno, J. (1995). The modified integer round-up property of the one-dimensional cutting stock problem. European Journal of Operational Research, 84, 562-571

Valerio de Carvalho, J. M. V. (2005). Using extra dual cuts to accelerate column generation. INFORMS Journal on Computing, 17, 175-182

Yanasse, H. H., Limeira, M. S. (2006). A hybrid heuristic to reduce the number of different patterns in cutting stock problems. Computers & Operations Research, 33, 2744-2756