Mathematical Biosciences and Engineering

MBE, 20(4): 7453–7486.
DOI: 10.3934/mbe.2023323
Received: 27 October 2022
Revised: 19 January 2023
Accepted: 29 January 2023
Published: 15 February 2023

http://www.aimspress.com/journal/MBE

Research article

A method for demand-accurate one-dimensional cutting problems with pattern reduction

Haihua Xiao[1,2], Qiaokang Liang[1,2,*], Dan Zhang[3], Suhua Xiao[4] and Gangzhuo Nie[5]

[1] College of Electrical and Information Engineering, Hunan University, Changsha 410082, China
[2] National Engineering Laboratory for Robot Vision Perception and Control, Changsha 410082, China
[3] Department of Mechanical Engineering, York University, Toronto ONM3J1P3, Canada
[4] College of Electromechanical Engineering, Guangdong Polytechnic Normal University, Guangzhou 510635, China
[5] Aluminum Corporation of China, Beijing 100000, China

* **Correspondence:** Email: qiaokang@hnu.edu.cn; Tel: +8613487599867.

Abstract: The main objective in the one-dimensional cutting stock problem (1D-CSP) is to minimize material costs. In practice, it is useful to focus on auxiliary objectives, one of which is to reduce the number of different cutting patterns. This paper discusses the classical integer IDCSP, where only one type of stock object is included. Meanwhile, the demands of various items must be precisely satisfied in the constraints. In other words, no overproduction or underproduction is allowed. Therefore, to solve this issue, a variable-to-constant method based on a new mathematical model is proposed. In addition, we integrate the approach with two other representative methods to demonstrate its effectiveness. Both benchmark instances and real instances are used in the experiments, and the results show that the methodology is effective in reducing patterns. In particular, in terms of the solutions to the real-life instances, the proposed approach presents a 31.93 to 37.6% pattern reduction compared to other similar methods (including commercial software).

Keywords: cutting; cutting stock; cutting pattern; column generation; optimization algorithms

1. Introduction

The classical one-dimensional cutting stock problem (1D-CSP) is a typical combinatorial

optimization problem. In the 1D-CSP, there are m items with lengths (l_1..., l_m) and demands (d^1, d^2..., d^m) that are cut from the stock objects of length L to minimize the material cost. The first notable research result in solving the 1D-CSP was based on the simplex column generation method [1]. In most of the existing studies, it is challenging to optimize the material cost and setup cost simultaneously, and fewer patterns lead to higher material costs [2–5].

In practice, the position of the cutting tool in the cutter must be adjusted every time before another cutting pattern is switched. Therefore, the time cost of adjusting the position of the cutting tool in the machine equipment cannot be ignored [6–8].

The algorithm proposed to solve the 1D-CSP model is called the variable-to-constant (VTC) algorithm. The demands of the items in the model must be precisely satisfied. It solves the 1D-CSP in two stages. In the first stage, it uses a column generation method to obtain the lower bound (LB), which is the minimum value of the number of stock objects used (material cost). In the second stage, VTC is used to generate the cutting patterns and to obtain a new objective function value. If the objective function value is less than or equal to the LB, the calculation is stopped. Otherwise, the calculation is continued by iterating m times or $2m$ times (each new cutting pattern generated is considered one iteration) to obtain a feasible initial solution. In parallel, the new method is fused with two authoritative methods in the literature to verify the validity of the methodology proposed in this paper.

1.1. Literature review

For cutting problems, López de Lacalle et al. [9] proposed a technical model to estimate the value of cutting forces, providing manufacturers with the opportunity to reduce production and delivery times. Approaches for determining the integer solution to the 1D-CSP (the demand is met exactly) fall into two main categories: heuristic approaches construct a good cutting pattern and use it as much as possible (constructive heuristics) and heuristic approaches round the relaxation solution (residual heuristic). Hinxman et al. [10] provided a detailed description of constructive heuristics, such as the first-fit decreasing (FFD) algorithm. The core of FFD is to first place the largest item into the pattern with the highest possible number of cutting patterns and no more than the demand. If the largest item no longer fits that cutting pattern, the second largest item is selected, and so on. The greedy heuristic follows the same philosophy as the FFD heuristic, but without prioritizing the largest items to construct the new cutting pattern. Later, Ongkunaruk et al. [11] modified the FFD heuristic to solve the bin packing problem (BPP). In 2009, the FFD and greedy heuristics were further modified to allow the redesign of cutting patterns with undesirable residues [12]. Specifically, such cutting patterns are not large enough to continue to be used or not small enough to be accepted as waste. In the same year, Poldi and Arenales [13] proposed three versions of greed-based residual heuristic rounding techniques, namely, GRH1, GRH2 and GRH3. Their core ideology is also based on using the relaxation solution to obtain the approximate integer solutions. In contrast, Cui et al. [14] solved the one-dimensional cutting problem for multiple stock objects in two stages. The first stage uses a pattern-set generation algorithm to generate patterns and combine them with column generation techniques to solve the residual problems. In the second stage, to further reduce the material cost indicator, the ILP model is solved using the CPLEX solver as a way to obtain a solution for stage two. The best solution for both phases is selected. In their second-stage solution method, overproduction occurs.

Recently, new research has been conducted to advance the development in this field. For example, a modified greedy heuristic (MGH) was proposed to optimize material cost [15]. It shows more

promising integer solutions than other methods but does not obtain the best cutting pattern (losing to the GRH algorithm in terms of pattern reduction). One thing in common among these methods is that they are all based on the slack solution of the column generation method to perform rounding to obtain the integer solution.

Here, we also give an overview of the literature on the cutting stock problem with setup cost (CSP-S). The sequential heuristic procedure of Haessler [16] was one of the first methods to deal with the CSP-S. It is essential to address the CSP-S in order to find the optimal trade-off between the numbers of objects and patterns [17–20]. Other SHP-based approaches can be found; for example, Mobasher and Ekici [21] proposed two local search algorithms and a column generation-based heuristic algorithm for CSP-S in order to minimize the total production cost, including material and setup costs. Cui et al. [4] presented a heuristic algorithm to deal with the CSP-S in two stages. In the first stage, they first used the heuristic to generate cutting patterns. In the second stage, based on the optimizer solver, a bi-objective optimization model of material and setup costs was developed to further reduce the setup cost. Martin et al. [22] modified Haessler's sequential heuristic procedure, but only individual instances were tested better than the other methods, and most instances were tested to perform poorly. Lately, Martin et al. [5] proposed a pattern-based pseudo-polynomial ILP formulation to solve the CSP-S, which depends on an upper bound on the maximum frequency of each pattern in the cutting scheme. We must emphasize one point: The current research works for solving the CSP-S allow for overproduction ($Ax \geq d$) in their models. In contrast, in the mathematical model developed in this study, the demand must be exactly satisfied ($Ax = d$).

A problem associated with pattern reduction is the pattern minimization problem (PMP). This is a problem of minimizing the number of patterns with a finite number of objects, and it is a nonlinear integer programming problem. It is well known that the PMP was proven to be a hard NP problem by McDiarmid [23]. Some exact methods for solving the PMP exist in the literature, and the solutions of these algorithms have met the demand exactly without overproduction [24–28]. Vanderbeck [24] reformulated the PMP model by using the Dantzig-Wolfe decomposition principle and adapted it to integer programming. The authors did this by dualizing the relevant nonlinear constraints in a Lagrangian fashion, and the problem was decomposed into K identical subproblems. They solved these subproblems to generate new columns. The experiments indicated that this exact algorithm reduced the number of setups by an average of 63% compared to the initial solution to the standard cutting problem. Alves and de Carvalho [26] further improved the model proposed by Vanderbeck [24] by adding a constraint on the total waste to the subproblem. This allows the number of column-generated subproblems (knapsack problems) to be solved to be significantly reduced. They treated the LP-optimal solution as an arc-flow formulation with an integer variable and used the branch-and-price-and-cut algorithm to solve the master problem. However, because of the branching constraint and the fact that the demand is exactly satisfied, the solution is forced to exceed the maximum waste of the optimal 1D-CSP solution. In Alves et al. [27], the authors developed a CP model to derive two tighter LBs for the PMP. They also used the CP model to derive some efficient inequalities to deal with the hard constraints. From the experimental results of the 16 tested real-life cases, only three had worse LBs than those of Vanderbeck [24]. Three others had tighter LBs, while the remaining 10 cases had the same LBs. The above-mentioned developers of the three exact algorithms for solving the PMP did not report material usage cost in the experiments they conducted. Afterward, Mobasher and Ekici [8] briefly stated that the main drawback of the PMP model (Vanderbeck's model) is the weak LP relaxation bound. In addition, this compact model has to meet the demand exactly without allowing

overproduction. This may increase the amount of material used and the number of different patterns used.

In conclusion, several of the above-mentioned exact methods for solving PMP models that have appeared thus far are computationally time-consuming (2 h per instance). In particular, their solution accuracy is very low for larger demands or problems with dimensionality greater than 20. However, the current literature shows that several residual heuristics can solve the 1D-CSP model with a single objective of material cost very well and with a short computational time. In their models, the demands are precisely satisfied. The shortcoming of the current solutions for such a 1D-CSP is that the cutting pattern cannot be reduced [15].

1.2. Our contributions

In this paper, the solution algorithm (VTC) for the 1D-CSP is developed by considering the requirements of item production in practical applications (the demands for the items must be met precisely). The authors provide important contributions in this domain. They rely on a mathematical model that is built to obtain the solution. In our approach, the feasible solution for the 1D-CSP is obtained by updating the established mathematical model once for each generated cutting pattern. Our approach attempts to form a linear relationship between the patterns and the variables by fixing one of the decision variables as a constant. Meeting the demand precisely is not conducive to the reduction of cutting patterns and minimization of material cost [8]. Two sets of well-known benchmark examples from the literature and a set of real instances are used to evaluate the advantages or disadvantages of the algorithms. The results indicate that the implementation of the new approach using a generic ILP solver (Gurobi) is able to obtain the optimal solution for the 1D-CSP in some instances and to acquire fewer cutting patterns.

2. Problem definition and mathematical formulations

In this section, we formally describe the 1D-CSP define some necessary notations and solve its relaxation solution by using the column generation approach.

2.1. Problem description

A 1D-CSP in which the demands are precisely met consists of the following parts: given an unlimited number of identical stock objects of length L (e.g., lengths of wood, aluminum alloy, rolls of paper), the mission is to cut d^i pieces of items of length l_i for $i \in I = \{1..., m\}$ to meet the quantities produced for the various items, keeping the number of stock objects used to a minimum. To simplify the model, we use p to define a pattern and its index. We also use P to denote a set of patterns and the indices of the patterns. In the 1D-CSP, a pattern p is represented by a column vector $a_p = (a_{1p}, \cdots, a_{ip}, \cdots, a_{mp})^t$, where a_{ip} denotes the number of items i in the cutting pattern p. The pattern p is required to fulfill

$$\sum_{i=1}^{m} a_{ip} l_i \le L, \tag{1}$$

$$0 \le a_{ip} \le d^i, \text{and integer } i = (1, \cdots, m). \tag{2}$$

We define an integer decision variable by x_n with each $p \in P$, referring to the number of cutting patterns p used. d^i represents the quantity corresponding to item i. Therefore, the main objective of the classic 1D-CSP is generally to minimize material cost (the number of objects used), and its mathematical model can be formulated as

$$\text{Minimize } \sum_{p \in P} x_p \tag{3}$$

$$\text{s.t. } \sum_{p \in P} a_{ip} x_p = d^i \qquad (i = 1, ..., m) \tag{4}$$

$$x_p \ge 0 \text{ and integer } (p \in P). \tag{5}$$

2.2. Column generation

Let us briefly describe the relaxed solution model for Eqs (3)–(5), where we simply remove the integer constraints. Furthermore, an initial feasible solution $\overline{p} \subseteq P$ can be easily achieved by initialization (e.g., through the use of heuristics to obtain this solution). Equations (3)–(5) then become

$$\text{Minimize } \sum_{p \in \overline{P}} x_p \tag{6}$$

$$\text{s.t. } \sum_{p \in \overline{P}} a_{ip} x_p = d^i \qquad (i = 1, ..., m) \tag{7}$$

$$x_p \ge 0, \text{ and } \overline{p} \in P. \tag{8}$$

The optimization problem described by Eqs (6)–(8) is called the restricted master problem (RMP). The RMP can provide the basis matrix B of the current iteration to update B^{-1} in the objective function of the subproblem. The computation stops when the objective function value in Eq (9) is greater than zero. Otherwise, a new column $p \notin \overline{p}$ can be generated iteratively to reduce the objective function value in Eq (6), and it is added to the RMP. The resulting basis matrix B is updated. The subproblem thus solved is as follows:

$$\text{Minimize } 1 - c_B B^{-1} a_p \quad c_B = (1, \cdots, 1) \tag{9}$$

$$\text{s.t. } \sum_{i=1}^{m} a_{ip} l_i \le L \tag{10}$$

$$0 \le a_{ip} \le d^i, \text{ an integer } i = (1, \cdots, m). \tag{11}$$

In (9), c_B is a row vector with *m* columns and all its elements are one. B is a matrix with *m* rows and *m* columns. $a_p = (a_{1p}, \cdots, a_{1p}, \cdots a_{mp})^t$ denotes a variable cutting pattern, expressed as a column vector. In (10), a_{ip} indicates the number of items *i* in a cutting pattern p, and l_i refers to the length of item i. *L* is expressed as the length of the stock object. Constraint (11) prevents the number of items *i* in the cutting pattern p from exceeding the total number demanded.

3. Solution methods

3.1. Our methodology for generating columns

In this section, we first build a new mathematical model and then solve it by using a new method called VTC. The new model for the 1D-CSP instances is built by fixing one of the decision variables as a constant. This enables a linear relationship between the decision variables and the columns (cutting patterns), which facilitates cutting pattern reduction. The main features of VTC are as follows.

1) In terms of pattern reduction, the solution quality of the method for some instances is much better than the solution quality of the standard 1D-CSP model. The demands d^i (i = 1..., m) in (4) are exactly satisfied.

2) In terms of constraints, the number of constraints is forced to increase after fixing one of the decision variables. Obtaining the solution is more time-consuming, which is not true for low-demand problems.

3) After fixing a decision variable as a constant, all decision variables $x = (x_1, \cdots, x_m)$ and the new column $a_p = (a_{1p}, \cdots, a_{ip}, \cdots, a_{mp})^t$ form a linear relationship. The objective function is transformed into a linear objective function related to the variable vector a_p.

To conveniently describe the solution process of the method in this paper, we first represent (1)–(5) in matrix form.

$$\text{Minimize} \quad f(x) = cx \tag{12}$$

$$\text{s.t.} \quad Ax = d \tag{13}$$

$$la_p \le L \tag{14}$$

$$0 \le a_{ip} \le d^i, i = (1, \cdots, m), \text{ an integer} \tag{15}$$

$$x \ge 0, \text{ an integer.} \tag{16}$$

In (12), $x = (x_1, \cdots x_n)^t$ indicates the column vector of decision variables, and *c* = (1..., 1) is a row vector containing *n* columns, where all elements are equal to one. The column vector a_p is a vector of variables, which we denote as $a_p = (a_{1p}, \cdots, a_{ip}, \cdots, a_{mp})^t$. Constraint (16) enables all decision variables to be nonnegative integers. Constraint (13) enables the quantity of all items to be precisely as required (i.e., demands are met exactly). $\boldsymbol{A}$ represents an m × n matrix in which each column is a cutting pattern. Its right end is a demand vector $d = (d^1, \cdots, d^m)^t$, expressing the number of different items produced.

$\boldsymbol{l}$=(l_1..., l_m) represents a row vector, where l_i denotes the size of the i^{th} item. L designates the length of the stock object. Constraints (14) and (15) indicate that the elements in the cutting pattern a_p have to satisfy $a_{1p}l_1 + \cdots + a_{mp}l_m \le L,$ and $0 \le a_{ip} \le d^i, i = (1, \cdots, m)$, an integer.

3.1.1. Mathematical modeling

To solve the optimization problem described in Section 3.1, we reconstruct the model of the 1D-CSP. We consider only m decision variables, i.e., their number is equal to the number of item types. Each iteration, which generates a new column, also represents the generation of a new cutting pattern p. Let us introduce a vector of integer column variables a_p which, for each $p \in P$, gives the number of items cut for each type of item in the cutting pattern p. Furthermore, we assume that the 1D-CSP for the k^{th} iteration with m decision variables and m items is established by generating the k^{th} column. Consequently, we express the mathematical model at each iteration in terms of the matrix and the vector.

The following notations are used to describe it:

$f(x)$	objective value (total number of stock objects used)
x_G	decision variable, which is currently fixed as a constant
d_k	column vector obtained by performing the elementary row transformation of a matrix on the demand vector $d_{k\text{-}1}$
$\overline{d_k}$	column vector obtained by forcing the k^{th} element of d_k to be 0
$s_{k\text{-}1}$	matrix obtained by performing the elementary row transformation of a matrix on matrix s_{k-2}
$\overline{s_{k\text{-}1}}$	vector obtained by forcing all elements of the k^{th} row of the matrix $s_{k\text{-}1}$ to 0
ss	row vector constructed from the k^{th} row of s_k
l	row vector representing the dimensions of the m items
a_p	cutting pattern p currently being solved
t	row vector with m columns and element values equal to one
m	number of item types
K	objective optimal LB, which can be obtained from the column generation algorithm
A_k	cutting scheme corresponding to the generation of the k^{th} column

The 1D-CSP model described is remodeled as

$$\text{Minimize } f(x) = x_1 + x_2 + \cdots + x_m$$

$$= x_G + t\overline{d_k} - t(\overline{s_{k\text{-}1}}a_p) \times x_G \tag{17}$$

$$\text{s.t.} \quad \overline{d_k} - \overline{s_{k\text{-}1}}a_p \times x_G \ge 0 \tag{18}$$

$$la_p \le L \tag{19}$$

$$ssa_p = 1 \tag{20}$$

$$a_p = (a_{1p}, \cdots, a_{ip}, \cdots a_{mp})^t, \ 0 \le a_{ip} \le d^i \ , \text{an integer} \tag{21}$$

$$k \in (0,1,\cdots,m,m+1\cdots,n), i \in (1,\cdots,m)$$

$$s_0 = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}, \ d_0 = \begin{pmatrix} d^1 \\ \vdots \\ d^m \end{pmatrix}, \ A_0 = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} \tag{22}$$

In the above model, the number of iterations k = 1, 2, 3..., n, where each iteration produces a new column and an objective function value. $\overline{d_k}$ and $\overline{s_{k\text{-}1}}$ are distinct from the previous iterations at each iteration. That is, the objective function and constraints need to be updated for each iteration. Equation (17) indicates that the objective is to minimize the material cost (the number of objects used), where x_G is equal to the value of the decision variable corresponding to column k at the previous iteration. Constraint (18) means that all decision variables are constrained to be greater than or equal to zero, except for the decision variable corresponding to column k. Constraint (19) limits the size of the space where the item is placed in the pattern a_p. Equation (20) is an equation constraint that arises after fixing a decision variable corresponding to the k^{th} column as a constant x_G. Equation (22) gives the initial matrices s_0, A_0 and d_0 created at the beginning iteration.

In an effort to determine an efficient solution to the above optimization problem, we present below the computational procedure that implements the new mathematical model constructed. The solution procedure is shown below.

Step 1: Input the initialization matrix for s_0, d_0, A_0 and set k = 1. Additionally, sort the items, i.e., $l_1 > l_2 >, \cdots, > l_m$ or $l_1 < l_2 <, \cdots, < l_m$.

Step 2: Update the objective function and constraints by $ss, \overline{s_{k\text{-}1}}$ and $\overline{d_k}$. ss can be obtained by forcing the elements in row one to zero in matrix $s_{k\text{-}1}$.

Step 3: Let the decision variable x_k be a constant x_G.

Step 4: Use the Gurobi solver to solve the k^{th} mathematical model to obtain a_p.

Step 5: Go to Step 6 if $f(x) > K \ \ and \ \ k < \beta m \ \ (\beta = 1 \ \text{or} \ \ \beta = 2)$; otherwise, stop the calculation. Record all cutting patterns produced and their decision variables.

Step 6: Compute $\overline{s_{k\text{-}1}}$, a_p and ssa_p, and apply their results as the k^{th} column in A.

Step 7: Perform elementary row transformation on matrix A while performing the same transformation on matrix $s_{k\text{-}1}$ and matrix d_k as that performed on matrix A.

Step 8: Let $k = k + 1,$ and return to Step 2.

Step 5 indicates that the method does not necessarily solve for the optimal solution. Each generation determines a cutting pattern a_p (Steps 2–8). For each cutting pattern solved, the objective

function and constraints need to be updated once (Step 2). Steps 6 and 7 depict a matrix elementary row transformation after each updated column in $\boldsymbol{A}$. In Step 1, different sequencing may result in different VTC calculations. The experiments conducted later in this paper rank the items from largest to smallest before solving.

3.1.2. Numerical example

In this section, we present a simple example to illustrate the solution process of the VTC method proposed in this paper.

Example. Consider an instance of the 1D-CSP with m = 4 items and demand vector $d_0 = (d^1,d^2,d^3,d^4) = (6,10,8,5)^{\mathrm{T}}$. Assume that L = 300, $\boldsymbol{l} = (l_1, l_2, l_3, l_4)$ = (150, 50, 40, 10) and row vector $\boldsymbol{t}$ = (1, 1, 1, 1). We also assume that the column vector $a_p = \begin{pmatrix} a_{1p} \\ a_{2p} \\ a_{3p} \\ a_{4p} \end{pmatrix}$. Before the iterative calculation, the following matrix is first initialized.

We let

$$A_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{23}$$

Meanwhile, we set

$$s_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{24}$$

Using (12)–(16), we obtain the following objective problem:

$$\text{Minimize } f_0(x) = x_1 + x_2 + x_3 + x_4. \tag{25}$$

This is subject to

$$A_0 x = d_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 10 \\ 8 \\ 5 \end{pmatrix} \quad . \tag{26}$$

As a result,

$$x_1 = 6, x_2 = 10, x_3 = 8, x_4 = 5 \tag{27}$$

and

$$f_0(x) = 6 + 10 + 8 + 5 = 29. \tag{28}$$

Concurrently, the LB $K = \lceil 5.9 \rceil = 6$ is obtained by the column generation method, which is the

minimum material cost. This can be used as a condition for whether the iterative calculation is stopped. Since $f_0(x) > K$, the calculation continues, and the next step involves generating new columns.

Iteration 1 (Column 1 is generated):

Replacing column 1 of A_0 in (26) with the unknown column vector a_p and using (12)–(14), we obtain the following objective problem:

$$\text{Minimize} \quad f_1(x) = x_1 + x_2 + x_3 + x_4 . \tag{29}$$

This is subject to

$$A_1 x = d_1 = \begin{pmatrix} a_{1p} & 0 & 0 & 0 \\ a_{2p} & 1 & 0 & 0 \\ a_{3p} & 0 & 1 & 0 \\ a_{4p} & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 10 \\ 8 \\ 5 \end{pmatrix}, \tag{30}$$

and

$$a_{1p} l_1 + a_{2p} l_2 + a_{3p} l_3 + a_{4p} l_4 \le L, \tag{31}$$

$$0 \le a_{1p} \le 6, 0 \le a_{2p} \le 10, 0 \le a_{3p} \le 8, 0 \le a_{4p} \le 5 \text{, an integer,} \tag{32}$$

$$x_1 \ge 0, x_2 \ge 0, x_3 \ge 0, x_4 \ge 0 . \tag{33}$$

To simplify the tedious calculation of the matrix, we can express the first column in matrix A_1 as

$$\begin{pmatrix} a_{1p} \\ a_{2p} \\ a_{3p} \\ a_{4p} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{1p} \\ a_{2p} \\ a_{3p} \\ a_{4p} \end{pmatrix} = s_0 a_p . \tag{34}$$

Fixing the decision variable x_1 to 6, we obtain

$$x_2 = 10 - 6a_{2p}, \tag{35}$$

$$x_3 = 8 - 6a_{3p}, \tag{36}$$

$$x_4 = 5 - 6a_{4p}, \tag{37}$$

$$a_{1p} = 1. \tag{38}$$

Let $\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \overline{s_0}$ and $\begin{pmatrix} 0 \\ 10 \\ 8 \\ 5 \end{pmatrix} = \overline{d_1}$; then, by observing the relationship between Eqs (30) and (34), (35)–(37) can be transformed into

$$\begin{pmatrix} 0 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 10 \\ 8 \\ 5 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times a_p \times x_1 = \overline{d_1} - \overline{s_0} a_p \times 6. \tag{39}$$

Now, substituting Eqs (35)–(37) into (29), the objective function can be written as follows:

$$\begin{aligned} \text{Minimize} \quad f_1(x) &= x_1 + x_2 + x_3 + x_4 \\ &= 29 - 6(a_{2p} + a_{3p} + a_{4p}) \\ &= 29 - 6t(\overline{s_0} a_p). \end{aligned} \tag{40}$$

The constraints are as follows:

$$x_2 = 10 - 6a_{2p} \geq 0, \tag{41}$$

$$x_3 = 8 - 6a_{3p} \geq 0, \tag{42}$$

$$x_4 = 5 - 6a_{4p} \geq 0, \tag{43}$$

$$a_{1p} l_1 + a_{2p} l_2 + a_{3p} l_3 + a_{4p} l_4 \leq L, \tag{44}$$

$$a_{1p} = 1, 0 \leq a_{2p} \leq 10, 0 \leq a_{3p} \leq 8, 0 \leq a_{3p} \leq 5, \text{ an integer.} \tag{45}$$

Constraints (41) to (44) are converted into matrix and vector forms:

$$\begin{pmatrix} 0 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 10 \\ 8 \\ 5 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times a_p \times x_1 = \overline{d_1} - \overline{s_0} a_p \times 6 \tag{46}$$

$$la_p = \begin{pmatrix} 150 & 50 & 40 & 10 \end{pmatrix} \cdot a_p \leq L \tag{47}$$

The Gurobi solver is used to solve the above optimization problem. Therefore, we obtain

$$a_1p=1, a_2p=1, a_3p=1, a_4p=0; x_1=6, x_2=4, x_3=2, x_4=5; f_1(x)=17.$$

Accordingly, A_1x is

$$A_1x=\begin{pmatrix}1&0&0&0\\1&1&0&0\\1&0&1&0\\0&0&0&1\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\\x_3\\x_4\end{pmatrix}=\begin{pmatrix}6\\10\\8\\5\end{pmatrix}. \tag{48}$$

Iteration 2 (Column 2 is generated):

Similarly, replacing column 2 of A_1 in Eq (48) with the unknown column vector a_p and using Eqs (12)–(14), we obtain the following objective problem:

$$\text{Minimize}\quad f_2(x)=x_1+x_2+x_3+x_4. \tag{49}$$

This is subject to

$$A_2x=d_2=\begin{pmatrix}1&a_{1p}&0&0\\1&a_{2p}&0&0\\1&a_{3p}&1&0\\0&a_{4p}&0&1\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\\x_3\\x_4\end{pmatrix}=\begin{pmatrix}6\\10\\8\\5\end{pmatrix}, \tag{50}$$

$$a_{1p}l_1+a_{2p}l_2+a_{3p}l_3+a_{4p}l_4\le L, \tag{51}$$

$$0\le a_{1p}\le 6, 0\le a_{2p}\le 10, 0\le a_{3p}\le 8, 0\le a_{3p}\le 5, \text{ an integer}, \tag{52}$$

$$x_1\ge 0, x_2\ge 0, x_3\ge 0, x_4\ge 0. \tag{53}$$

Next, we formulate Eq (50) as (54) and simultaneously perform the same operation on matrix s_1, with the following result. Thus,

$$A_3x=\begin{pmatrix}1&a_{1p}&0&0\\0&a_{2p}-a_{1p}&0&0\\0&a_{3p}-a_{1p}&1&0\\0&a_{4p}&0&1\end{pmatrix}\cdot\begin{pmatrix}x_1\\x_2\\x_3\\x_4\end{pmatrix}=\begin{pmatrix}6\\4\\2\\5\end{pmatrix}, \tag{54}$$

and then, s_0 in Eq (24) becomes s_1 in (55):

$$s_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{55}$$

At this point, fixing the decision variable x_2 to 4, we obtain

$$x_1 = 6 - 4a_{1p}, \tag{56}$$

$$x_3 = 2 - 4(a_{3p} - a_1 p), \tag{57}$$

$$x_4 = 5 - 4a_{4p}, \tag{58}$$

$$(a_{2p} - a_{1p}) \times 4 = 4. \tag{59}$$

Because $s_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ in (55), we can let $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \overline{s_1}$. Because $A_3 x = \begin{pmatrix} 6 \\ 4 \\ 2 \\ 5 \end{pmatrix}$ in (54), we can let $\begin{pmatrix} 6 \\ 0 \\ 2 \\ 5 \end{pmatrix} = \overline{d_2}$. Then, Eqs (56)–(58) can be expressed as

$$\begin{pmatrix} x_1 \\ 0 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 0 \\ 2 \\ 5 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times a_p \times x_2 = \overline{d_2} - \overline{s_1} a_p \times 4. \tag{60}$$

Through Eqs (56)–(58) and (49), the objective function is easily formulated as

$$\begin{aligned} \text{Minimize } f_2(x) &= x_1 + x_2 + x_3 + x_4 \\ &= 17 - 4(a_{3p} + a_{4p}) \\ &= 17 - 4t(\overline{s_1} a_p). \end{aligned} \tag{61}$$

The constraints are as follows:

$$x_1 = 6 - 4a_{1p} \geq 0, \tag{62}$$

$$x_3 = 2 - 4(a_{3p} - a_{1p}) \geq 0, \tag{63}$$

$$x_4 = 5 - 4a_{4p} \geq 0, \tag{64}$$

$$a_{1p}l_1 + a_{2p}l_2 + a_{3p}l_3 + a_{4p}l_4 \leq L, \tag{65}$$

$$(a_{2p} - a_{1p}) \times 4 = 4, \tag{66}$$

$$0 \leq a_{1p} \leq 6, 0 \leq a_{2p} \leq 10, 0 \leq a_{3p} \leq 8, 0 \leq a_{4p} \leq 5, \text{ an integer.} \tag{67}$$

Constraints (62)–(65) are converted into matrix and vector forms:

$$\begin{pmatrix} x_1 \\ 0 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 0 \\ 2 \\ 5 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times a_p \times x_2 = \overline{d_2} - \overline{s_1} a_p \times 4 \geq 0, \tag{68}$$

$$la_p = \begin{pmatrix} 150 & 50 & 40 & 10 \end{pmatrix} \cdot a_p \leq L. \tag{69}$$

Using row 2 of matrix s_2 as a row vector ***ss,*** such that $ss = (-1,1,0,0)$, (66) is as follows:

$$(a_{2p} - a_{1p}) = \begin{pmatrix} -1 & 1 & 0 & 0 \end{pmatrix} \cdot a_p = ssa_p = 1. \tag{70}$$

The Gurobi solver is used to solve the above optimization problem. Therefore, we obtain

$$a_1 p = 1, a_2 p = 2, a_3 p = 1, a_4 p = 1;$$
$$x_1 = 2, x_2 = 4, x_3 = 2, x_4 = 1; f_2(x) = 9.$$

In this way, $A_2 x$ can be written as

$$A_2 x = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 10 \\ 8 \\ 5 \end{pmatrix}. \tag{71}$$

Iteration 3 (Column 3 is generated):

Since $f_2(x) > K$, the calculation continues to produce the next column. The objective optimization problem is formulated as

$$\text{Minimize} \quad f_3(x) = x_1 + x_2 + x_3 + x_4. \tag{72}$$

This is subject to

$$A_3 x = d = \begin{pmatrix} 1 & 1 & a_{1p} & 0 \\ 1 & 2 & a_{2p} & 0 \\ 1 & 1 & a_{3p} & 0 \\ 0 & 1 & a_{4p} & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 10 \\ 8 \\ 5 \end{pmatrix}, \tag{73}$$

$$a_{1p}l_1 + a_{2p}l_2 + a_{3p}l_3 + a_{4p}l_4 \le L, \tag{74}$$

$$0 \le a_{1p} \le 6, 0 \le a_{2p} \le 10, 0 \le a_{3p} \le 8, 0 \le a_{3p} \le 5, \text{ an integer,} \tag{75}$$

$$x_1 \ge 0, x_2 \ge 0, x_3 \ge 0, x_4 \ge 0. \tag{76}$$

The next step of expressing (73) as (77) while performing the same elimination operation as that performed for matrix s_2 results in the following:

$$A_4 x = \begin{pmatrix} 1 & 0 & 2a_{1p} - a_{2p} & 0 \\ 0 & 1 & a_{2p} - a_{1p} & 0 \\ 0 & 0 & a_{3p} - a_{1p} & 0 \\ 0 & 0 & a_{4p} - a_{2p} + a_{1p} & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 2 \\ 1 \end{pmatrix}. \tag{77}$$

Then, s_1 in (55) becomes s_2 in (78):

$$s_2 = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & 0 & 1 \end{pmatrix}. \tag{78}$$

As such, fixing the decision variable x_3 to 2 gives

$$x_1 = 2 - 2(2a_{1p} - a_{2p}), \tag{79}$$

$$x_2 = 4 - 2(a_{2p} - a_1 p), \tag{80}$$

$$x_4 = 1 - 2(a_{4p} - a_{2p} + a_{1p}), \tag{81}$$

$$(a_{3p} - a_{1p}) \times 2 = 2 . \tag{82}$$

Then, let $\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 1 \end{pmatrix} = \overline{s_2}$, $\begin{pmatrix} 2 \\ 4 \\ 0 \\ 1 \end{pmatrix} = \overline{d_3}$, so that Eqs (79)–(81) are formulated as

$$\begin{pmatrix} x_1 \\ x_2 \\ 0 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 0 \\ 1 \end{pmatrix} - \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 1 \end{pmatrix} \times a_p \times x_3 = \overline{d_3} - \overline{s_2} a_p \times 2 \quad . \tag{83}$$

For this iteration, the objective function can be represented as follows:

$$\begin{aligned} \text{Minimize} \quad f_3(x) &= x_1 + x_2 + x_3 + x_4 \\ &= 9 - 2(2a_{1p} + a_{2p} + a_{4p}) \\ &= 9 - 2t(\overline{s_2} a_p) . \end{aligned} \tag{84}$$

The constraints are as follows:

$$x_1 = 2 - 2(2a_{1p} - a_{2p}) \geq 0, \tag{85}$$

$$x_2 = 4 - 2(a_{2p} - a_1 p) \geq 0, \tag{86}$$

$$x_4 = 1 - 2(a_{4p} - a_{2p} + a_{1p}) \geq 0, \tag{87}$$

$$a_{1p} l_1 + a_{2p} l_2 + a_{3p} l_3 + a_{4p} l_4 \leq L, \tag{88}$$

$$(a_{3p} - a_{1p}) \times 2 = 2, \tag{89}$$

$$0 \leq a_{1p} \leq 6, 0 \leq a_{2p} \leq 10, 0 \leq a_{3p} \leq 8, 0 \leq a_{4p} \leq 5 , \text{ an integer.} \tag{90}$$

Constraints (85) to (88) are represented in matrix and vector forms:

$$\begin{pmatrix} x_1 \\ x_2 \\ 0 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 0 \\ 1 \end{pmatrix} - \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 1 \end{pmatrix} \times a_p \times x_3 = \overline{d_3} - \overline{s_2} a_p \times 2 \geq 0, \tag{91}$$

$$la_p = \begin{pmatrix} 150 & 50 & 40 & 10 \end{pmatrix} \cdot a_p \leq L . \tag{92}$$

Using row 3 of matrix s_3 as a row vector ***ss,*** such that $ss = (-1,0,1,0)$, (89) is as follows:

$$(a_{3p} - a_{1p}) = \begin{pmatrix} -1 & 0 & 1 & 0 \end{pmatrix} \cdot a_p = ssa_p = 1. \tag{93}$$

The Gurobi solver is used to solve the above optimization problem. Therefore, we obtain

$$\begin{aligned} &a_1 p = 0, a_2 p = 0, a_3 p = 1, a_4 p = 0; \\ &x_1 = 2, x_2 = 4, x_3 = 2, x_4 = 1; f_3(x) = 9. \end{aligned}$$

Note that $f_2(x)$ has the same result as $f_3(x)$ and that A_3x is the same as A_2x; Thus, A_3x is as follows:

$$A_3x = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 10 \\ 8 \\ 5 \end{pmatrix}. \tag{94}$$

Iteration 4 (Column 4 is generated):

Because $f_3(x) > K$, the calculation continues. The objective optimization problem is described as follows:

$$\text{Minimize} \quad f_4(x) = x_1 + x_2 + x_3 + x_4. \tag{95}$$

This is subject to

$$A_4x = d_4 = \begin{pmatrix} 1 & 1 & 0 & a_{1p} \\ 1 & 2 & 0 & a_{2p} \\ 1 & 1 & 1 & a_{3p} \\ 0 & 1 & 0 & a_{4p} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 6 \\ 10 \\ 8 \\ 5 \end{pmatrix}, \tag{96}$$

$$a_{1p}l_1 + a_{2p}l_2 + a_{3p}l_3 + a_{4p}l_4 \le L, \tag{97}$$

$$0 \le a_{1p} \le 6, 0 \le a_{2p} \le 10, 0 \le a_{3p} \le 8, 0 \le a_{3p} \le 5, \text{ an integer,} \tag{98}$$

$$x_1 \ge 0, x_2 \ge 0, x_3 \ge 0, x_4 \ge 0. \tag{99}$$

Consequently, by the matrix elimination operation, Eq (96) can be presented as (100), and the result is as follows:

$$A_5x = \begin{pmatrix} 1 & 0 & 0 & 2a_{1p} - a_{2p} \\ 0 & 1 & 0 & a_{2p} - a_{1p} \\ 0 & 0 & 1 & a_{3p} - a_{1p} \\ 0 & 0 & 0 & a_{4p} - a_{2p} + a_{1p} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 2 \\ 1 \end{pmatrix}. \tag{100}$$

Additionally, s_2 in Eq (78) is written as s_3 below:

$$s_3 = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & 0 & 1 \end{pmatrix}. \tag{101}$$

Observing the relationship between the matrix A_5 in Eqs (100) and (101), we conclude that the following equation holds:

$$\begin{pmatrix} 2a_{1p} - a_{2p} \\ a_{2p} - a_{1p} \\ a_{3p} - a_{1p} \\ a_{4p} - a_{2p} + a_{1p} \end{pmatrix} = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} a_{1p} \\ a_{2p} \\ a_{3p} \\ a_{4p} \end{pmatrix}. \tag{102}$$

Therefore, after fixing the decision variable x_4 to 1, the expressions for the other variables can be written as

$$x_1 = 2 - (2a_{1p} - a_{2p}), \tag{103}$$

$$x_2 = 4 - (a_{2p} - a_1 p), \tag{104}$$

$$x_3 = 2 - (a_{3p} - a_{1p}), \tag{105}$$

$$(a_{4p} - a_{2p} + a_{1p}) \times 1 = 1. \tag{106}$$

Given that $\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \overline{s_3}$ and $\begin{pmatrix} 2 \\ 4 \\ 2 \\ 0 \end{pmatrix} = \overline{d_4}$, Eqs (103)–(105) are described as

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 2 \\ 0 \end{pmatrix} - \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \times a_p \times x_4 = \overline{d_4} - \overline{s_3} a_p \times 1. \tag{107}$$

The objective optimization problem is as follows:

$$\text{Minimize} \quad f_4(x) = x_1 + x_2 + x_3 + x_4$$

$$= 9 - a_{3p} \tag{108}$$

$$= 9 - t(\overline{s_3} a_p)$$

The constraints are as follows:

$$x_1 = 2 - (2a_{1p} - a_{2p}) \geq 0, \tag{109}$$

$$x_2 = 4 - (a_{2p} - a_1 p) \geq 0, \tag{110}$$

$$x_3 = 2 - (a_{3p} - a_{1p}) \geq 0, \tag{111}$$

$$a_{1p} l_1 + a_{2p} l_2 + a_{3p} l_3 + a_{4p} l_4 \leq L, \tag{112}$$

$$(a_{4p} - a_{2p} + a_{1p}) \times 1 = 1, \tag{113}$$

$$0 \leq a_{1p} \leq 6, 0 \leq a_{2p} \leq 10, 0 \leq a_{3p} \leq 8, 0 \leq a_{4p} \leq 5 \text{, an integer.} \tag{114}$$

Consequently, Eqs (109)–(111) can be represented in matrix and vector forms:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 2 \\ 0 \end{pmatrix} - \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \times a_p \times 1 = \overline{d_4} - \overline{s_3} a_p \times 1 \geq 0, \tag{115}$$

$$la_p = \begin{pmatrix} 150 & 50 & 40 & 10 \end{pmatrix} \cdot a_p \leq L. \tag{116}$$

Using row 4 of matrix s_4 as a row vector ***ss,*** such that $ss = (1,-1,0,1)$, the constraint given by (113) can be written as

$$(a_{4p} - a_{2p} + a_{1p}) = \begin{pmatrix} 1 & -1 & 0 & 1 \end{pmatrix} \cdot a_p = ssa_p = 1. \tag{117}$$

The Gurobi solver is used to solve the above optimization problem. Consequently, we obtain

$$a_1 p = 1, a_2 p = 0, a_3 p = 3, a_4 p = 0;$$
$$x_1 = 0, x_2 = 5, x_3 = 0, x_4 = 1; f_4(x) = 6.$$

Because $f_4(x) = K = 6$, the calculation is cut off. If $f_4(x) > K$, we can perform the same operation as the above operation and loop to generate a new column; the optimal outcome is as follows:

$$x=\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}=\begin{pmatrix} 0 \\ 5 \\ 0 \\ 1 \end{pmatrix};\ A=\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 2 & 0 & 0 \\ 1 & 1 & 1 & 3 \\ 0 & 1 & 0 & 0 \end{pmatrix};\ \text{minimize}\ f(x)=6.$$

Ax exactly meets the demand. That is, $Ax=d$. From the results, only two columns are needed to reach the optimal integer solution. As the decision variables x_1 and x_3 are zero, columns 1 and 3 are therefore invalid. In view of practical applications, it is beneficial to reduce the setup cost.

Note that our solution procedure for generating columns is relaxed with respect to the constraints on the decision variables (i.e., the decision variables are not restricted to integers). The experimental results show that integer solutions can be obtained. We elaborate further on how to deal with this problem.

3.2. Other approaches

To produce an ideal or at least acceptable solution, we introduce several heuristics in some well-known literature. With respect to solving the 1D-CSP, there are two main types of methods, one being constructive heuristics, and the other being residual heuristics.

3.2.1. Constructive heuristics

Constructive heuristics is a way of determining an integer solution to a one-dimensional cutting problem; specifically, it is a way of constructing a good cutting pattern and using as much of it as possible [10]. No items are allowed to be overproduced. Two well-known procedures for constructing cutting patterns are FFD and greedy approaches.

The general framework for constructive heuristics is as follows.

Step 1: Construct a good cutting pattern for a type of stock length.

Step 2: Among the cutting patterns generated in Step 1, select the one with minimum waste.

Step 3: Use the cutting pattern in Step 2 as much as possible without overproduction.

Step 4: Update the demand of the items.

Step 5: If the demand for each of these items is met precisely, stop. Otherwise, go to Step 1.

1) FFD heuristic

The procedure is to prioritize the largest item into the pattern until its demand is met. If the largest item cannot be placed, the second largest item is considered for placement, and so on. The cutting pattern is completed when all demands of the items have been precisely met.

2) Greedy heuristic

In this paper, the greedy procedure consists of solving the knapsack problem in Step 1, which has only one type of stock length as a raw material (stock object). The backpack problem appears as follows.

$$\text{Maximize}\ \ l_1a_{1\mathrm{p}}+l_2a_{2\mathrm{p}}+\cdots l_ma_{m\mathrm{p}}$$

$$\text{s.t.}\ \ l_1a_{1\mathrm{p}}+l_2a_{2\mathrm{p}}+\cdots l_ma_{m\mathrm{p}}\le L$$

$$0 \le a_{ip} \le r_i,\ a_{ip} \text{ integer},\ i = 1, \cdots, m. \tag{118}$$

In (118), l_i refers to the length of item type *i*, $i = 1, \cdots, m$ and r_i is the residual demand for item type *i*. r_i is updated in Step 4. At the beginning, $r_i = d^i$, and d^i is the demand for item *i*, $i = 1, \cdots, m$.

3.2.2. Residual heuristics

Residual heuristics produce an optimal integer solution for the continuous relaxation of (6)–(8). If at least one element of the relaxation solution vector is not a nonnegative integer, then we can use the residual heuristic for rounding. Otherwise, the relaxation solution is the optimal integer solution. The residual heuristics are described before we define a residual problem.

Definition 1 (Residual problem). Let $\bar{x}$ be an approximate integer solution rounded down for $\boldsymbol{x}$. $\bar{x} = (\lfloor x_1 \rfloor, \cdots, \lfloor x_m \rfloor)$. $r = d - A\bar{x}$ stands for the residual demand. We can formulate the residual problem in the form of (6) to (8), where demand d varies with r.

Note that the cutting patterns of the optimization problem consist of two parts: the first part consists of the cutting patterns (columns) corresponding to the relaxation solution, and the second part consists of the cutting patterns generated by the residual problem.

The general framework for residual heuristics:

Step 1: Let $c = 0$ and $r^c = d$ be the initial data for the beginning residual problem.

Step 2: Solve the relaxation solution to the residual equation problem by using the column generation technique. Assume that the relaxation solution $\boldsymbol{x}$ comprises all nonnegative integers, and then stop. Otherwise, go to Step 3.

Step 3: Find an approximate integer solution $\bar{x}^c$. If it is a null vector, go to Step 5.

Step 4: Update the demand for the residual problem, $r^{c+1} = r^c - A\bar{x}^c$. If $r^c = 0$, then stop. Otherwise, go to Step 2.

Step 5: Solve the remaining residual problem.

Whether this algorithm is considered good or bad is mainly related to determining how to go about rounding through Step 3 and how to solve for the remaining problem through Step 5. Nevertheless, Step 2 is even more critical when considering the auxiliary indicator (setup), as it can directly affect the number of cutting patterns. There are two main types of Step 3 rounding methods: one is downward rounding, as in the FFD and greedy approaches; the other is a downward or upward rounding strategy, such as GRH1, GRH2 and GRH3 (see Poldi et al. [13]). Note that the GRH used in the experimental tests in this paper refer to GRH1.

For Step 5, the remaining problem can be solved and made optimal by using some other method [29,30]. An integer linear optimization problem (considering the generation of all columns) can also be built to solve it. Such approaches, however, allow for overproduction, whereas the optimization problem studied in this paper does not allow for overproduction.

3.3. Our algorithms

In this section, we fuse the VTC method with the FFD and greedy methods. Thus, two improved algorithms (i.e., Residual-VTC-FFD and Residual-VTC-Greedy, respectively) are obtained. The effectiveness of the VTC method is verified. More specifically, VTC can be used as described in

Section 3.2.2 as a methodology for solving Step 2 and for Step 5. The definition of the residual problem, including its associated parameters, is assumed to be identical to that given in Section 3.2.2.

General framework of our approach:

Step 1: Let $c = 0$ and $r^c = d$ be the initial data for the original residual problem.

Step 2: Solve the relaxation solution to the residual equation problem by using the VTC technique in Section 3.1. Assume that the obtained objective function value $f(x) \leq K$ (K is defined as an LB of the current residual problem). If $f(x) > K$, the relaxation solution obtained by the column generation technique replaces the relaxation solution found via the VTC technique.

Step 3: Assume that the relaxation solution $\boldsymbol{x}$ is all nonnegative integers, and then stop. Otherwise, go to Step 4.

Step 4: Find an approximate integer solution $\bar{x}^c$. If it is a null vector, go to Step 6.

Step 5: Update the demand for the residual problem, $r^{c+1} = r^c - A\bar{x}^c$. If $r^c = 0$, then stop. Otherwise, go to Step 2.

Step 6: Use the VTC and greedy methods to solve the remaining residual problems separately and choose the best solution as the final integer solution.

In Step 6, we prefer the side with the smallest objective function value as the final solution. If both have the same objective function value, the side with the lowest cutting patterns is chosen as the final solution. Our method differs from the residual heuristic in Steps 2 and 6. The good optimization capability of the column generation technique is exploited to compensate for the weak optimization capability of the VTC method. The quality of the solution can be improved. The cutting patterns (setups) can be reduced while obtaining a satisfactory material cost.

4. Computational experiments

We used two different instances, one instance is from a previous study in the literature and the other is from an aluminum alloy factory in China. Benchmark instances are used to compare our algorithm with the published 1D-CSP algorithms. To solve the 1D-CSP model presented in this paper, there are six main published algorithms: FFD, greedy, residual-FFD, residual-greedy, residual-GRH and residual-BPP algorithms. These algorithms can be found in Cerqueira et al. [15] and Poldi et al. [13]. All experiments were implemented with Python 3.7 installed on a computer with an Intel Core i5-10400 processor at 2.90 GHz. The calculation time for each instance was limited to 20 s.

First set of instances: The 17 hard instances given by Wascher and Gau [29] are available at http://or.dei.unibo.it/library/bpplib.

Second set of instances: Foerster and Wäscher [31] used the problem generator CutGen1 to randomly generate a set of instances with 18 different classes.

Third set of instances: The authors collected 20 practical instances of making aluminum doors and windows in an aluminum alloy factory (see the Appendix).

4.1. Results from the first set of instances

To test several algorithms on more challenging benchmarks, we tested 17 small-scale instances that were difficult to solve. These instances were designed by Wascher and Gau [29]. The stock object length L = 10,000, and the number of required items varies from 33 to 63. The majority of these items tend to have less demand. The minimum requirement of the item is only 1; thus, these instances can

be used to estimate the performance of the constructive heuristic and our method. This is because the residual problem is a solution process for a lower-demand problem. For this set of instances, the number of iterations of our method was set to m, i.e., $\beta = 1$.

Table 1. Results of 17 hard instances (L = 10000).

Name	Constructive-FFD		Constructive-Greedy		VTC		
	Obj	N_P	Obj	N_P	Obj	N_P	LB
Waescher TEST0005	33	32	29	27	32	**22**	28
Waescher TEST0014	28	28	26	24	30	**17**	24
Waescher TEST0022	16	16	15	15	16	**11**	14
Waescher TEST0030	31	31	28	27	33	**22**	27
Waescher TEST0044	15	15	14	14	16	**12**	14
Waescher TEST0049	12	12	11	11	12	**9**	11
Waescher TEST0054	15	15	15	15	16	**12**	14
Waescher TEST0055A	16	16	15	15	17	**10**	15
Waescher TEST0055B	21	21	20	20	21	**15**	20
Waescher TEST0058	23	23	21	20	22	**14**	20
Waescher TEST0065	18	18	16	16	19	**13**	15
Waescher TEST0068	13	13	12	12	13	**9**	12
Waescher TEST0075	14	14	13	13	15	**12**	13
Waescher TEST0082	31	31	26	25	**26**	**22**	24
Waescher TEST0084	18	18	16	16	17	16	16
Waescher TEST0095	18	18	16	16	17	**12**	16
Waescher TEST0097	13	13	12	12	13	**8**	12
Total	335	334	305	298	335	236	295

The results of three different methods are given in Table 1. The solution results of two well-known constructive heuristics and the proposed VTC technique are reported. Column 1 defines the name of the instances being tested, and the last column is the ideal optimal value of the objective function value. In the table, Obj represents the value of the objective function found, and N_P refers to the number of different cutting patterns (setups). The last line shows the sum of the counts for all instances.

The sum of the numbers of different cutting patterns for the VTC solution was significantly smaller than that for the other two methods. There is only one instance here, i.e., Waescher_TEST0084, where the N_P metric did not outperform the other two methods. This indicates that VTC is powerful in terms of simplifying setups (reducing the cutting patterns).

Comparing the FFD and greedy methods, the greedy method was best in terms of optimal objective function values, as opposed to our method, which was indistinguishable from FFD. For the Waescher_TEST0082 instance, our method obtained the same objective function value as the other methods. In this instance, a 12.0% reduction in pattern count could be obtained by using VTC instead of the greedy algorithm, and a 29.0% (= 1-22/31) reduction in pattern count relative to FFD was observed. Relative to the greedy algorithm, the new solution reduced the number of patterns by a total of 20.8%, and a total of 29.3% relative to FFD.

4.2. Results from the second set of instances

The second set consists of 18 classes of benchmark instances from Foerster and Wäscher [31].

There are 100 instances in each class, and we only tested the first 10 instances in each class. In accordance with Gau and Wäscher [32], the randomly generated instances could be the same as the original ones. The characteristics of the instances are shown in Table 2, where d denotes the number of item demands, d_r represents the average demand and m represents the number of item types. Different combinations of v_1 and v_2 were used to determine the size of items randomly generated in the interval $[v_1L, v_2L]$. The results of the calculation are shown in Table 3. Obj denotes the average of the number of stock objects used. N_P indicates the average number of cutting patterns. $\overline{LB}$ stands for the average of the ideal objective function values. Using *Gap* as the distance between the actual objective function value and the ideal solution (LB), we report it as a percentage:

$$Gap_i = \frac{Obj - LB}{Obj} \times 100\ \%, \quad Gap = \frac{Gap_1 + \cdots + Gap_{10}}{10} \tag{119}$$

Table 2. Characteristics of instances in Set 2.

Class	m	v_1	v_2	d_r
1	10	0.01	0.2	10
2	10	0.01	0.2	100
3	20	0.01	0.2	10
4	20	0.01	0.2	100
5	40	0.01	0.2	10
6	40	0.01	0.2	100
7	10	0.01	0.8	10
8	10	0.01	0.8	100
9	20	0.01	0.8	10
10	20	0.01	0.8	100
11	40	0.01	0.8	10
12	40	0.01	0.8	100
13	10	0.2	0.8	10
14	10	0.2	0.8	100
15	20	0.2	0.8	10
16	20	0.2	0.8	100
17	40	0.2	0.8	10
18	40	0.2	0.8	100

Our objective function considered only the quantity of stock objects used, and the demand for each item type had to be met precisely, as in the mathematical model studied by Cerqueira et al. [15]. We tested the set of instances with each of the six representative mainstream algorithms, without considering the MGH method used by the authors in the literature. This is because the MGH method considered the results of two types of stock objects $L_1 = 1000$ and $L_2 = 1001$.

These 18 different types of instances can be divided into three groups, each containing six types of instances and having the same l_i range (see Table 2). Specifically, Classes 1 to 6 were treated as Group 1, Classes 7 to 12 were treated as Group 2 and Classes 13 to 18 were treated as Group 3. The average pattern count and the average number of objects used count for each group of instances are reported in Tables 4 and 5, respectively. The algorithms shown in line 1 of Tables 4 and 5 correspond to all of the algorithms in Table 3. The fifth row in Tables 4 and 5 shows the average of the three previous sets of results. The last row in Table 4 represents the sum of the pattern counts for all instances in Table 3. The last row in Table 5 presents the sum of the stock objects used for all instances.

Table 3. Results of 10 instances in each one of the 18 classes (L = 1000).

	Constructive-FFD			Constructive-Greedy			Residual-FFD			Residual-Greedy			Residual-GRH			Residual-BPP			Residual-VTC-FFD			Residual-VTC-Greedy			
Class	Obj	N_P	Gap	Obj	N_P	Gap	Obj	N_P	Gap	Obj	N_P	Gap	Obj	N_P	Gap	Obj	N_P	Gap	Obj	N_P	Gap	Obj	N_P	Gap	$\overline{LB}$
1	11.7	9.4	2.02	11.5	9.6	7.6	11.5	9.1	0.77	11.4	9.0	0	11.4	8.8	0	11.4	9.0	0	**11.4**	**7.8**	**0**	**11.4**	**7.8**	**0**	11.4
2	114.8	17.6	3.90	111.9	24.9	1.53	110.4	13.4	0.16	110.2	13.2	0	**110.2**	**12.7**	**0**	110.2	12.9	0	110.5	13.5	0.13	110.3	12.3	0.07	110.2
3	24.3	20.4	3.64	23.4	20.9	0	23.5	17.2	0.5	23.4	17.4	0	23.4	17.3	0	23.4	17.4	0	**23.4**	**14.9**	**0**	**23.4**	**14.9**	**0**	23.4
4	238.4	32.8	44.4	229.8	72.1	0.84	227.9	27.0	0.58	227.7	26.8	0	227.7	25.8	0	227.7	25.5	0	**227.7**	**25.1**	**0**	**227.7**	**25.1**	**0**	227.7
5	44.8	36.7	5.57	42.5	37.6	0.46	43.1	32.3	1.89	42.3	31.5	0	42.3	32.1	0	42.3	32.8	0	42.7	30.4	0.47	**42.3**	**27.1**	**0**	42.3
6	445.0	64.4	4.39	423.0	134.0	0.37	420.6	49.4	0.10	420.0	48.8	0	420.0	48.1	0	420.0	50.0	0	420.4	47.6	0.08	**420.0**	**48.1**	**0**	420.0
7	56.1	11.4	10.88	53.4	11.8	5.9	50.7	10.8	0.85	50.3	10.4	0	50.3	10.1	0	50.3	10.1	0	**50.3**	**9.7**	**0**	**50.3**	**9.7**	**0**	50.3
8	571.8	13.6	9.29	550.2	15.9	6.71	516.5	11.1	0.03	516.3	12.2	0	516.3	11.8	0.	516.3	11.6	0	**516.3**	**11.2**	**0**	**516.3**	**11.2**	0	516.3
9	121.6	20.5	17.87	108.0	22.3	7.51	100.8	19.9	0.08	100.6	19.8	0	100.6	19.7	0	100.6	19.8	0	**100.6**	**19.6**	**0**	**100.6**	**19.6**	0	100.6
10	1211.8	24.5	17.84	996.4	45.3	9.02	1003.6	22.1	0.03	1003.2	21.7	0	1003.3	20.1	0.02	1003.2	21.9	0	1003.3	20.9	0.01	**1003.2**	**20.8**	**0**	1003.2
11	221.6	46.7	22.5	195.7	48.3	12.17	172.2	41.0	0.06	172.1	40.9	0	172.1	40.1	0	172.1	41.2	0	**172.1**	**39.9**	**0**	**172.1**	**39.9**	0	172.1
12	2217.1	52.4	20.32	1941.1	126.0	11.35	1724.0	45.6	0.06	1723.4	45.0	0.03	1723.3	44.1	0.02	1723.0	45.6	0.01	1723.4	43.7	0.03	**1723.3**	**43.5**	**0.02**	1722.9
13	69.7	11.4	10.85	60.0	10.8	3.39	62.7	10.6	0.22	62.6	10.5	0	62.6	10.3	0	62.6	11.2	0	**62.6**	**10.1**	**0**	**62.6**	**10.1**	**0**	62.6
14	691.6	10.9	10.48	648.4	11.0	4.01	623.8	11.0	0	623.8	11.0	0	623.8	10.9	0	623.8	10.8	0	**623.8**	**10.7**	**0**	**623.8**	**10.7**	**0**	623.8
15	149.4	23.1	15.75	130.8	21.8	3.38	126.9	20.2	0.16	**126.7**	**20.0**	**0**	126.7	20.1	0	126.7	21.3	0	126.8	20.1	0.09	**126.7**	**20.0**	**0**	126.7
16	1489.9	24.9	15.83	1305.0	32.3	3.28	1265.5	21.1	0.01	**1265.4**	**21.0**	0	1265.4	21.1	0	1265.4	21.2	0	1265.4	21.7	0	**1265.4**	**21.0**	**0**	1265.4
17	280.5	47.8	21.74	239.9	48.2	8.33	220.0	39.1	0.12	219.7	38.8	0	**219.7**	**37.9**	**0**	219.7	38.9	0	219.8	39.4	0.04	219.7	38.8	0	219.7
18	2802.8	48.1	21.5	2421.8	97.7	9.35	2194.1	43.0	0.01	2193.8	42.7	0	**2193.8**	**41.4**	**0**	2193.8	42.8	0	2193.9	43.0	0.01	2193.8	42.7	0	2193.8

The six algorithmic solutions had a sum of material costs close to the LB (8892.4), whereas the other two constructive heuristics failed to reach optimality. The best performance in terms of material cost savings was the R-BPP algorithm. Our solution in terms of material cost was also closer to the LB. Remarkably, our method acquired the least number of cutting patterns compared to all other methods. This indicates that the algorithm (R-VTC-Greedy) developed can reduce cutting patterns.

Comparing R-VTC-Greedy and R-Greedy, the sum of the cutting pattern counts of the R-VTC-Greedy algorithm was smaller than that of the R-Greedy algorithm. As seen in Table 4, the total sum of pattern counts decreased by 3.95% (= (1 - 423.3) / 440.7). On average, a 3.92% (= (1 - 23.52) / 24.48) reduction in pattern counts could be achieved by the R-VTC-Greedy algorithm. This illustrates that the VTC method we developed can improve the solution quality of R-Greedy.

Comparing R-VTC-FDD and R-FDD, the sum of the cutting pattern counts of R-VTC-FDD was smaller than that of R-FDD. From the last row of Table 4, the sum of pattern counts solved by R-VTC-FDD was reduced by 3.28% (= (1 - 429.3) / 443.9). On average, a 3.92% (= (1 - 23.85) / 24.66) reduction in pattern counts could be achieved by R-VTC-FDD. This illustrates that the VTC method we developed can also improve the solution quality of R-FDD.

From the results for the set of instances, the reduction in cutting patterns was not very significant, mainly for two reasons. One is that the VTC algorithm developed by the authors currently has difficulty in obtaining fewer cutting patterns for a 1D-CSP with higher demands. The second is that the problem also leads to a reduction in the optimization capability of VTC when the demand is higher. This can be analyzed from the solution for the first set of instances and the solution for the third set of instances that follow.

In our methodology, the number of iterations was set to 2 m to solve this set of instances. We limited the computational time to 20 s for each instance, considering the effectiveness of the solution for instances with higher dimensionality.

Table 4. Summary of the first set in terms of the cutting pattern.

	FFD	Greedy	R-FFD	R-Greedy	R-GRH	R-BPP	R-VTC-FDD	R-VTC-Greedy
Group 1	30.22	49.85	24.73	24.45	24.13	24.57	23.22	**22.55**
Group 2	28.18	44.93	25.08	25.0	24.32	25.03	24.17	**24.12**
Group 3	27.70	36.97	24.17	24.0	23.62	24.37	24.17	**23.88**
Average	28.7	43.92	24.66	24.48	24.02	24.66	23.85	**23.52**
Total	516.6	790.5	443.9	440.7	432.4	444	429.3	**423.3**

Table 5. Summary of the first set in terms of the material cost.

	FFD	Greedy	R-FFD	R-Greedy	R-GRH	R-BPP	R-VTC-FDD	R-VTC-Greedy	LB
Group 1	146.5	140.35	139.50	**139.17**	**139.17**	**139.17**	139.35	139.18	139.17
Group 2	733.30	640.80	594.63	594.32	594.30	**594.25**	594.33	594.30	748.67
Group 3	913.98	800.98	748.83	**748.67**	**748.67**	**748.67**	748.72	**748.67**	748.67
Average	597.93	527.38	494.32	494.05	494.05	**494.03**	494.13	494.05	494.02
Total	10762.9	9492.8	8897.8	8892.9	8892.9	**8892.5**	8894.4	8892.9	8892.4

4.3. Results for the third set of instances

In the third set, there were 20 practical instances of industrial on-site processing. Tests on aluminum cutting were used to evaluate the performance of the best solutions generated by the new methodology for cost (material cost) minimization problems, as well as the usefulness of pattern reduction in practical cutting. Within a manufacturing plant, the objects in stock are 6000 cm in length. The engineering instances being solved at random were selected. These instances are provided in Table A.1 in the Appendix. For this set of instances, the number of iterations for our method was set to m, i.e., $\beta = 2$. In Table 6, the software Chuangying was developed by China Zibo Zhiying Network Technology Co., Ltd., which is a well-known and professional developer of door and window design and management software in China. The results of the calculated instances are exhibited in Table 6. To facilitate the analysis of the results, we set *Obj* to the number of objects used, and N_P is defined as the number of cutting patterns. *Gap* is the relative optimal gap, expressed as a percentage. N_p^{VTC}, N_p^{FDD}, N_p^{Greedy}, $N_p^{\text{Chuangying}}$, $N_p^{\text{Residual-FFD}}$, $N_p^{\text{Residual-}Greedy}$, $N_p^{Residual\text{-}GRH}$ and $N_p^{Residual\text{-}VTC\text{-}Greedy}$ represent the total numbers of cutting patterns produced by the different methods. As Table 6 shows, Residual-VTC-Greedy obtained a more satisfactory solution than several other similar mainstream algorithms.

The following data were obtained from the last row of the table:

$$\frac{N_p^{\text{Chuangying}} - N_p^{\text{Residual-VTC-Greedy}}}{N_p^{\text{Chuangying}}} = 31.93\% \qquad \frac{N_p^{\text{FFD}} - N_p^{\text{Residual-VTC-Greedy}}}{N_p^{\text{FFD}}} = 36.9\%$$

$$\frac{N_p^{Greedy} - N_p^{\text{Residual-VTC-Greedy}}}{N_p^{Greedy}} = 37.6\% \qquad \frac{N_p^{Residual\text{-}FFD} - N_p^{\text{Residual-VTC-Greedy}}}{N_p^{Residual\text{-}FFD}} = 35.8\%$$

$$\frac{N_p^{Residual\text{-}Greedy} - N_p^{\text{Residual-VTC-Greedy}}}{N_p^{Residual\text{-}Greedy}} = 33.1\% \qquad \frac{N_p^{Residual\text{-}GRH} - N_p^{\text{Residual-VTC-Greedy}}}{N_p^{Residual\text{-}GRH}} = 32.7\%$$

They state that, by addressing the 1D-CSP model, the rate of improvement in the cutting pattern was 31.93% relative to Chuangying, 36.9% relative to FFD, 37.6% relative to the greedy algorithm, 35.8% relative to Residual-FFD, 33.1% relative to Residual-Greedy and 32.7% relative to Residual-GRH. Among the 20 instances reported, only instances 7_6000, 14_6000 and 18_6000 had cutting patterns that were not improved by our method. Closer inspection of the results shows that our method is effective not only in solving the original problem, but also in solving the remaining problems. For example, comparing the results for instances 8, 11, 13, 15, 19 and 20, when VTC failed to obtain the optimal objective function value, the cutting patterns could still be reduced in the solution to the residual problem at a later stage.

Figures 1 and 2 display the cutting patterns and their respective corresponding integer decision variables obtained by solving instance 1_6000 using our method and the commercial software, respectively. The symbol 1*6000*4 stands for the first cutting pattern produced and requires four objects to be cut according to that cutting pattern. The length of the object was 6000 cm. As can be seen from the graph, the two different methods solved for the same number of objects used, but our solution produced only four different cutting patterns, whereas the commercial software produced eight different cutting patterns, and the other methods in Table 6 produced even more cutting patterns.

Table 6. Results of 20 practical instances (L = 6000).

	Chuangying			Constructive-FFD			Constructive-Greedy			Residual-FFD			Residual-Greedy			Residual-GRH			VTC			Residual-VTC-Greedy			
Name	Obj	N_P	Gap	Obj	N_P	Gap	Obj	N_P	Gap	Obj	N_P	Gap	Obj	N_P	Gap	Obj	N_P	Gap	Obj	N_P	Gap	Obj	N_P	Gap	LB
1_6000	20	8	0	22	11	9.1	20	13	0	21	11	4.8	20	10	0	20	13	0	**20**	**4**	**0**	**20**	**4**	**0**	20
2_6000	10	10	0	10	9	0	10	10	0	10	9	0	10	9	0	10	9	0	**10**	**3**	**0**	**10**	**3**	**0**	10
3_6000	8	7	0	10	9	20	8	8	0	8	7	0	8	7	0	8	7	0	**8**	**5**	**0**	**8**	**5**	**0**	8
4_6000	10	8	0	10	9	0	10	10	0	10	9	0	10	9	0	10	9	0	**10**	**4**	**0**	**10**	**4**	**0**	10
5_6000	8	7	0	10	8	20	9	8	11	8	7	0	8	7	0	8	7	0	**8**	**3**	**0**	**8**	**3**	**0**	8
6_6000	18	11	0	22	11	18.2	20	11	10	19	14	5.3	18	13	0	19	11	5.3	**18**	**6**	**0**	**18**	**6**	**0**	18
7_6000	12	11	8.3	13	9	15.4	**12**	**9**	**8.3**	12	11	8.3	12	11	8.3	12	10	8.3	14	6	0	12	11	8.3	11
8_6000	6	5	0	7	7	14.3	6	6	0	6	5	0	6	5	0	6	5	0	8	3	25	**6**	**3**	**0**	6
9_6000	9	9	0	10	9	10.0	9	9	0	10	9	10.0	**9**	**8**	**0**	10	8	10.0	10	5	10	**9**	**8**	**0**	9
10_6000	9	9	0	10	9	10.0	9	9	0	10	9	10.0	**9**	**8**	**0**	10	8	10.0	10	5	10	**9**	**8**	**0**	9
11_6000	11	10	0	12	8	8.3	11	10	0	11	8	0	11	8	0	11	8	0	12	3	8.3	**11**	**6**	**0**	11
12_6000	8	7	0	10	9	20.0	8	8	0	8	7	0	8	7	0	8	7	0	**8**	**5**	**0**	**8**	**5**	**0**	8
13_6000	8	8	0	9	8	11.1	8	8	0	9	9	11.1	8	8	0	8	8	0	9	5	11.1	**8**	**6**	**0**	8
14_6000	6	**5**	**0**	7	7	14.3	6	6	0	7	7	14.3	6	6	0	6	6	0	8	3	25	6	6	0	6
15_6000	15	9	0	16	10	6.3	16	12	6.3	15	8	0	15	8	0	15	8	0	18	6	16.7	**15**	**7**	**0**	15
16_6000	4	4	0	5	4	20.0	4	4	0	4	4	0	4	4	0	4	4	0	**4**	**1**	**0**	**4**	**1**	**0**	4
17_6000	14	13	0	15	15	6.7	14	14	0	15	15	6.7	14	14	0	14	14	0	**14**	**5**	**0**	**14**	**5**	**0**	14
18_6000	11	**9**	**0**	12	11	8.3	11	10	0	11	11	0	11	11	0	11	10	0	15	7	26.7	11	10	0	11
19_6000	8	8	0	9	9	11.1	8	8	0	8	8	0	8	8	0	8	8	0	12	3	33.3	**8**	**6**	**0**	8
20_6000	8	8	0	8	7	0	8	8	0	8	8	0	8	8	0	8	8	0	12	3	33.3	**8**	**6**	**0**	8
Total	203	166		227	179		207	181		210	176		203	169		206	168		228	85		**203**	**113**		202

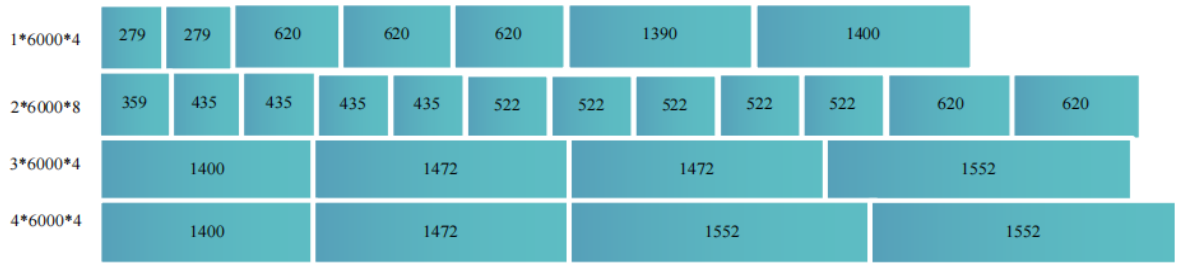

Figure 1. Solution to instance 1_6000 using our method.

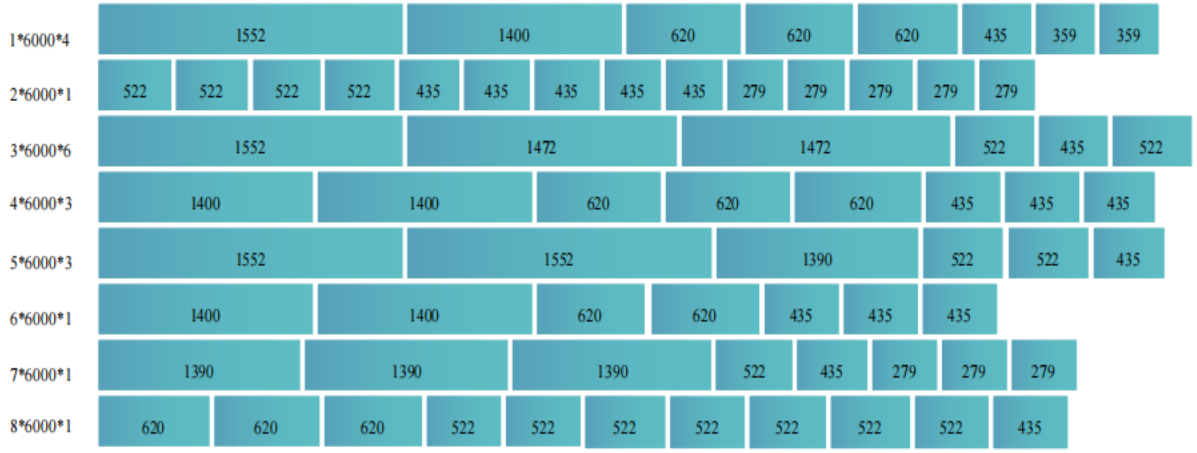

Figure 2. Solution to instance 1_6000 using Chuangying software.

5. Conclusions and future works

In this paper, we reviewed some heuristics and column generation techniques from the literature. A new method called VTC has also been introduced and used to design two other new approaches for solving the integer 1D-CSP by successfully integrating it into two residual heuristics. In the first set of instances, the proposed VTC algorithm showed pattern reduction performance comparable to that of two well-known heuristics. In the second set of instances, a total of 100 well-known instances of different types were tested. The test results for the second set of instances showed an improvement of the average setup cost compared to several classical algorithms. In the third set of instances, 20 real-life instances were tested and the setup cost solved by the designed approaches was significantly less than that of the other published algorithms. In fact, in some practical applications, pattern reduction is essential to reduce the time cost of the setup and adjustment of machinery and equipment. Moreover, in practical applications, the demand needs to be met precisely (overproduction is not allowed), which helps to save material costs. We all know that overproduction tends to generate material waste.

This paper opens up new possibilities for future work. Efforts will be made to extend the VTC algorithm to address the 1D-CSP with multiple stock lengths and further pattern reduction in the future. In addition, the optimization capability and computational time of the new method can continue to be improved.

Acknowledgments

We are grateful for the incoming comments by the reviewers and editors, and we also appreciate the real data provided by Guangdong Weiye Aluminum Factory Co.

This work was supported in part by the Project of the National Key Research and Development Program of China (2021YFC1910402, 2022YFB4703103), National Natural Science Foundation of China (62073129, U21A20490, U22A2059) and Hunan Provincial Natural Science Foundation of China (2022JJ10020).

Conflict of interest

No potential conflict of interest has been reported by the authors.

References

1. P. Gilmore, R. E. Gomory, A linear programming approach to the cutting-stock problem, *Comput. Oper. Res.*, **9** (1961), 849–859. http://doi.org/10.1287/opre.9.6.849
2. H. H. Yanasse, M. S. Limeira, A hybrid heuristic to reduce the number of different patterns in cutting stock problems, *Comput. Oper. Res.*, **33** (2006), 2744–2756. https://doi.org/10.1016/j.cor.2005.02.026
3. A. C. Cherri, M. N. Arenales, H. H. Yanasse, K. C. Poldi, A. C. G. Vianna, The one-dimensional cutting stock problem with usable leftovers-A survey, *Eur. J. Oper. Res.*, **236** (2014), 395–402. https://doi.org/10. 1016/j.ejor.2013.11.026
4. Y. Cui, C. Zhong, Y. Yao, Pattern-set generation algorithm for the one-dimensional cutting stock problem with setup cost, *Eur. J. Oper. Res.*, **243** (2015), 540–546. https://doi.org/10.1016/j.ejor.2014.12.015
5. M. Martin, H. H. Yanasse, L. L. Salles-Neto, Pattern-based ILP models for the one-dimensional cutting stock problem with setup cost, *J. Comb. Optim.*, **44** (2022), 557–582. https://doi.org/10.100 7/s10878-022-00848-z
6. Y. Cui, Z. Liu, C-Sets-based sequential heuristic procedure for the one dimensional cutting stock problem with pattern reduction, *Optim. Methods Software*, **26** (2011), 55–167. https://doi.org/10.1080/10556780903420531
7. H. H. Yanasse, K. C. Poldi, G. R. L. Cerqueira, Modified KOMBI to reduce the different patterns in cutting stock problems, International Federation of Operational Research Societies Melbourne, Australia, 2011.
8. A. Mobasher, A. Ekici, Solution approaches for the cutting stock problem with setup cost, *Comput. Oper. Res.*, **40** (2013), 225–235. https://doi.org/10.1016/j.cor.2012.06
9. L.N. López de Lacalle, Improving the high-speed finishing of forming tools for advanced high-strength steels (AHSS), *Int. J. Adv. Manuf. Technol.*, **29** (2006), 49–63. https://doi.org/10.1016/j.cor.2012.06.007
10. A. I. Hinxman, The trim-loss and assortment problems: a survey, *Eur. J. Oper. Res.*, **5** (2007), 8–18. https://doi.org/10.1016/0377-2217(80)90068-5
11. P. Ongkunaruk, Asymptotic worst-case analyses for the open bin packing problem, Faculty of the Virginia Polytechnic Institute, 2005.

12. A. C. Cherri, M. N. Arenales, H. H. Yanasse, The one-dimensional cutting stock problem with usable leftover: a heuristic approach, *Eur. J. Oper. Res.*, **6** (2009), 897–908. https://doi.org/10.1016/j.ejor.2008.04.039
13. K. C. Poldi, M. N. Arenales, Heuristics for the one-dimensional cutting stock problem with limited multiple stock lengths, *Comput. Oper. Res.*, **36** (2009), 2074–2081. https://doi.org/10.1016/j.cor.2008.07.001
14. Y. Cui, Y. P. Cui, Z. Zhao, Pattern-set generation algorithm for the one one-dimensional multiple stock sizes cutting stock problem, *Eng. Optim.*, **9** (2015), 1289–1301. https://doi.org/10.1080/0305215X.2014.969726
15. G. Cerqueira, S. S. Aguiar, M. Marques, Modified greedy heuristic for the one-dimensional cutting stock problem. *J. Comb. Optim.*, **42** (2021), 657–674. https://doi.org/10.1007/s10878-021-00695-4
16. R. W. Haessler, Controlling cutting pattern changes in one-dimensional trim Problems, *Comput. Oper. Res.*, **23** (1975), 483–493. https://doi.org/10.2307/169698
17. S. Umetani, M. Yagiura, T. Ibaraki, One-dimensional cutting stock problem to minimize the number of different patterns, *Eur. J. Oper. Res.*, **146** (2003), 388–402. https://doi.org/10.1016/S0377-2217(02)00239-4
18. J. Lee, In situ column generation for a cutting-stock problem, *Comput. Oper. Res.*, **34** (2007), 2345–2358. https://doi.org/10.1016/j.cor.2005.09.007
19. R. R. Golfeto, A. C. Moretti, L. L. S. Neto, A genetic symbiotic algorithm applied to the cutting stock problem with multiple objectives, *Adv. Model. Optim.*, **11** (2009),473–501.
20. S. A. Araujo, K. C. Poldi, J. Smith, A genetic algorithm for the one-dimensional cutting stock problem with setups, *Pesqui. Oper.*, **34** (2014), 165–187. https://doi.org/10.1590/0101-7438.2014.034.02.0165
21. A. Mobasher, A. Ekici, Olution approaches for the cutting stock problem with setup cost, *Comput. Oper. Res.*, **40** (2013), 225–235.https://doi.org/10.1016/j.cor.201206.007
22. M. Martin, A. Moretti, M. Gomes-Ruggiero, L. S. Neto, Modifification of Haessler's sequential heuristic procedure for the one-dimensional cutting stock problem with setup cost, *Production*, **28** (2018), e20170105. https://doi.org/10.1590/0103-6513.20170105
23. C. McDiarmid, Pattern minimisation in cutting stock problems, *Discrete Appl. Math.*, **98** (1999), 121–130. https://doi.org/10.1016/S0166-218X(99)00112-2
24. F. Vanderbeck, Exact algorithm for minimizing the number of setups in the one-dimensional cutting stock problem, *Oper. Res.*, **48** (2000), 915–926. https://doi.org/10.2307/222998
25. G. Belov, G. Scheithauer, The number of setups (different patterns) in one-dimensional stock cutting, *Preprint MATH-NM-15-2003, Department of Mathematics, Dresden University of Technology*, 2003
26. C. Alves, J. M. V. De Carvalho, A branch-and-price-and-cut algorithm for the pattern minimization problem, *RAIRO Oper. Res.*, **42** (2008), 435–453. https://doi.org/10.1051/ro:2008027
27. C. Alves, R. Macedo, J. V. D. Carvalho, New lower bounds based on column generation and constraint programming for the pattern minimization problem, *Comput. Oper. Res.*, **36** (2009), 2944–2954. https: //doi.org/10.1016/j.cor.2009.01.008
28. A. Aloisio, C. Arbib, F. Marinelli, On LP relaxations for the pattern minimization problem, *Networks*, **57** (2011), 247–253. https://doi.org/10.1002/net.20422

29. G. Wascher, T. Gau, Heuristics for the integer one-dimensional cutting stock problem: a computational study, *Operations-Research-Spektrum*, **18** (1996), 131–144. https://doi.org/10.1007/BF01539705
30. M. Delorme, M. Iori, Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems, *INFORMS J. Comput.*, **32** (2020), 101–119. https://doi.org/10.1287/ijoc.2018.0880
31. H. Foerster, G. Wäscher, Pattern reduction in one-dimensional cutting stock problems, *Int. J. Prod. Res.*, **38** (2000), 1657–1676. https://doi.org/10.1080/002075400188780
32. T. Gau, G. Wäscher, CUTGEN1: A problem generator for the one-dimensional cutting stock problem, *Eur. J. Oper. Res.*, **84** (1995), 572–579. https://doi.org/10.1016/0377-2217(95)00023-J

Appendix

Table A.1. Random real-life instances in the field.

1_6000									
Length (cm)	1552	1472	1400	1390	620	522	435	359	279
Demand	12	12	12	4	32	40	32	8	8
2_6000									
Length (cm)	1700	1690	1420	1130	1100	1080	760	730	720
Demand	6	2	2	2	8	2	2	2	2
Length (cm)	560	530	520	510	400	330			
Demand	8	2	12	4	2	20			
3_6000									
Length (cm)	3780	2840	2310	2240	2140	2070	1290	1270	1170
Demand	2	2	2	3	2	2	1	1	4
Length (cm)	1150	720	700						
Demand	2	1	1						
4_6000									
Length (cm)	1714	1704	1420	1130	1114	1094	760	730	720
Demand	6	2	2	2	8	2	2	2	2
Length (cm)	560	544	534	524	414	330			
Demand	8	2	12	4	2	20			
5_6000									
Length (cm)	1574	1390	1370	620	514	394	390		
Demand	6	4	4	16	14	8	4		
6_6000									
Length (cm)	4730	4720	3100	3040	2800	2740	1285	1270	1012.5
Demand	2	2	6	4	6	4	4	2	4
Length (cm)	1007.5	630	405	345	305	245			
Demand	4	8	4	4	4	4			
7_6000									
Length (cm)	2410	1820	1810	1750	1740	1500	1440	1400	1100
Demand	2	12	6	3	1	2	2	2	2
Length (cm)	810	740	570	410	340				
Demand	2	1	4	10	5				
8_6000									
Length (cm)	1524	1234	1160	874	765	714	705	620	560
Demand	2	2	4	10	4	4	4	4	4
Length (cm)	370	344							
Demand	2	4							

Continued on next page

9_6000									
Length (cm)	2140	2120	2110	2100	1640	1330	1180	840	820
Demand	2	2	4	2	2	4	2	2	2
Length (cm)	810	800	740	620	500	300			
Demand	4	2	2	10	4	4			
10_6000									
Length (cm)	2154	2134	2124	2114	1640	1330	1180	854	834
Demand	2	2	4	2	2	4	2	2	2
Length (cm)	824	814	740	620	500	314			
Demand	4	2	2	10	4	4			
11_6000									
Length (cm)	1280	1140	1050	1030	900	620	580	570	560
Demand	6	4	4	10	24	2	4	8	2
Length (cm)	450	340							
Demand	8	4							
12_6000									
Length (cm)	3780	2840	2310	2240	2140	2070	1290	1270	1170
Demand	2	2	2	3	2	2	1	1	4
Length (cm)	1150	720	700						
Demand	2	1	1						
13_6000									
Length (cm)	2560	2410	2280	2210	1650	1580	1560	1530	1170
Demand	2	2	4	3	2	1	2	2	1
Length (cm)	750	715	630	570					
Demand	1	2	2	2					
14_6000									
Length (cm)	1510	1220	1160	860	765	705	700	620	560
Demand	2	2	4	10	4	4	4	4	4
Length (cm)	370	330							
Demand	2	4							
15_6000									
Length (cm)	1960	1900	1890	1870	1850	1830	885	875	530
Demand	12	12	2	4	4	2	4	4	16
Length (cm)	460								
Demand	8								
16_6000									
Length (cm)	1271	1151	716	631	571				
Demand	8	4	4	4	4				
17_6000									
Length (cm)	1940	1780	1580	1320	1220	950	910	870	860
Demand	2	4	4	4	12	6	10	4	6
Length (cm)	850	735	710	690	620	550	230	210	
Demand	4	8	2	2	8	4	2	2	
18_6000									
Length (cm)	2090	1810	1780	1770	1750	1740	1680	1670	1170
Demand	4	2	2	2	6	4	2	4	2
Length (cm)	1080	1040	890	860	820	790	630	570	
Demand	1	1	2	2	3	1	2	4	
19_6000									
Length (cm)	1234	1040	830	800	780	760	694	620	514
Demand	8	4	2	4	4	4	2	14	18
Length (cm)	494								
Demand	4								

Continued on next page

20_6000									
Length (cm)	1220	1040	830	800	780	760	680	620	500
Demand	8	4	2	4	4	4	2	14	18
Length (cm)	480								
Demand	4