TOP (2022) 30:194–220
https://doi.org/10.1007/s11750-021-00611-3

ORIGINAL PAPER

A residual recombination heuristic for one-dimensional cutting stock problems

B. S. C. Campello[1] · C. T. L. S. Ghidini[1] · A. O. C. Ayres[1] · W. A. Oliveira[1]

Received: 3 August 2020 / Accepted: 2 June 2021 / Published online: 14 June 2021

Abstract

Cutting stock problems arise in manufacturing industries where large objects need to be cut into smaller pieces. The cutting process usually results in a waste of material; thus, mathematical optimization models are used to reduce losses and take economic gains. This paper introduces a new heuristic procedure, called the Residual Recombination Heuristic (RRH), to the one-dimensional cutting stock problem. The well-known column generation technique typically produces relaxed solutions with non-integer entries, which, in this approach, we associate with a set of residual cutting patterns. The central aspect of this contribution involves recombining these residual cutting patterns in different ways; therefore, generating new integer feasible cutting patterns. Experimental studies and statistical analyses were conducted based on different instances from the literature. We analyze heuristic performance by measuring the waste of material, the number of instances solved to optimality, and by comparing it with other heuristics in the literature. The computational time suggests the suitability of the heuristic for solving real-world problems.

Keywords Cutting stock problem · Multiperiod cutting stock problem · Heuristics

B. S. C. Campello is grateful to CAPES (master's program scholarships, Grant 01P-3717/2017). C. T. L. S. Ghidini wishes to thank FAPESP (Grant 2015/02184-7) and FAEPEX-Unicamp (Grant 519.292-262/15). A. O. C. Ayres is grateful to FAEPEX-Unicamp (master's program scholarships, Grants 1123/15 and 519.292-262/15).

✉ B. S. C. Campello
betania@decom.fee.unicamp.br

C. T. L. S. Ghidini
carla.ghidini@fca.unicamp.br

A. O. C. Ayres
amanda@dca.fee.unicamp.br

W. A. Oliveira
waoliv@unicamp.br

1 School of Applied Sciences, University of Campinas, R. Pedro Zaccaria, 1300, Limeira, São Paulo 13484-350, Brazil

Mathematics Subject Classification 90C59

1 Introduction

One-dimensional cutting stock problems (1D-CSP) have been of great interest in research and practical applications of manufacturing industries for a long time. This problem deals with cutting large objects into smaller demanded pieces. The concept of *cutting patterns* refers to how the object is divided to be cut to generate these pieces. Each cutting pattern produces a specific waste of material. The 1D-CSP aims at choosing the best combination of cutting patterns that minimizes the total waste of material (or the number of objects used), subject to fulfilling the demand for pieces.

Kantorovich (1960) proposed the first 1D-CSP mathematical model, and extensions of the CSP considering new variables and constraints, such as those presented by Dyckhoff (1990), Wäscher et al. (2007), and Melega et al. (2018), design the cutting process more accurately for the production environment. Among these extensions, we highlight the multiperiod one-dimensional cutting stock problem with multiple stock lengths with limited availability (1D-MCSP) (Poldi and Araujo 2016). The 1D-MCSP model comprises objects of different lengths in limited quantities in stock and considering their inventory costs. The planning horizon is divided into periods, and as the demand is known for all periods, it is possible to anticipate the production of pieces for the following periods by considering additional inventory costs. In practice, the anticipation of pieces reduces the total costs due to the selection of better cutting patterns that decrease the overall material waste (Poldi and Araujo 2016).

The cutting stock problem can be formulated as an integer combinatorial linear programming problem, making it difficult to be solved using exact approaches (Chen et al. 2019). Nevertheless, there are some valuable methods in the literature to find the optimal integer solution for the cutting stock and similar bin-packing problems. Degraeve and Schrage (1999) and Degraeve and Peeters (2003) introduced variations of the branch-and-price algorithm aimed to improve its efficiency. Vanderbeck (1999) studied how heuristics can be useful to improve the branch-and-price algorithm. Belov and Scheithauer (2006) proposed a branch-and-cut-and-price algorithm for the 1D-CSP and the two-dimensional version of the problem. Delorme et al. (2016) reviewed and tested mathematical models and exact algorithms for the one-dimensional cutting stock problem. Carvalho (1999) proposed a mixed-integer linear program, known as *arc-flow*, and its exact solution using column generation and branch-and-bound algorithm. Recently, Delorme and Iori (2020) studied pseudo-polynomial formulations for the CSP problems and introduced a new formulation called *Reflect*, which improved the classical arc-flow approach.

Although exact approaches are capable of finding the optimal solution, the computational time required may increase to an unacceptable level depending on the problem's size (Chen et al. 2019). Thus, many heuristic approaches have been proposed, such as the classical family of heuristics of type *Fit Decreasing*:

First Fit Decreasing (FFD), Best Fit Decreasing (BFD), and Next Fit Decreasing (NFD) (Kim and Wy 2010). There also is a type of *evolutionary* heuristics that uses crossover and mutation techniques, such as those proposed by Araujo et al. (2011), and more recently, by Chen et al. (2019). Abdel-Basset et al. (2018) introduced a *meta-heuristic algorithm*, which showed to be efficient in finding the optimal solution and convergence speed. There are *exhaustive repetition* heuristics, as proposed by Hinxman (1980), Stadtler (1990), and Wäscher and Gau (1996), which consist of generate cutting patterns with minor material losses and are used as many times as possible without exceeding the demand. Another type of heuristics is the *residual heuristics*, such as those proposed by Gilmore and Gomory (1963), Stadtler (1990), Wäscher and Gau (1996), and Poldi and Arenales (2009), that, in brief, consist of solving the continuous relaxation and then rounding the fractional entries of the relaxed solution. Wäscher and Gau (1996) and Poldi and Arenales (2009) compared the exhaustive repetition and the residual procedures; they showed that the residual approaches usually present better solutions.

Regarding to the extensions to multiperiod variants of cutting and packing problems, Poldi and Araujo (2016) introduced a generalized arc flow model and proposed a residual-type heuristic for solving the 1D-MCSP. Ma et al. (2019) proposed two heuristics for the 1D-MCSP, which minimize the waste of material and also the patterns' setup. A similar problem to the 1D-MCSP is the one-dimensional cutting stock problem with usable leftovers over multiperiod scenarios (1D-CSPL). In this problem, a leftover of a pattern can be classified as a waste of material or retail. If retail, it is returned to stock to be used in the subsequent cutting process. In the literature, it is possible to find some studies related to this problem (Cherri et al. 2014), such as Cherri et al. (2013), and more recently, Ravelo et al. (2020).

MCSP also arises integrated with the lot-sizing and scheduling problems in many studies, such as Ghidini et al. (2007), Poltroniere et al. (2016), Campello et al. (2020), Ayres et al. (2019), Pitombeira and Athayde (2020), and others reviewed by Melega et al. (2018). The resulting integrated problem generates a complex mathematical model that highlights the importance of developing efficient solution methods for MCSP. For instance, Ghidini et al. (2007) solved the continuous relaxation of the integrated model using the column generation procedure, applying a residual heuristic to find an integer solution. Poltroniere et al. (2016) suggested obtaining the relaxed optimal solution with a column generation procedure and then utilizing the columns (cutting patterns) to solve the integer mathematical model. Campello et al. (2020) considered the integrated problem as multi-objective optimization, solving the integer problem with a fixed number of cutting patterns that present the lowest waste of material. Pitombeira and Athayde (2020) proposed a mathematical model for the 1D-MCSP integrated with the scheduling problem with heterogeneous orders that was solved by a metaheuristic algorithm based on a fix-and-optimize and a random local search.

This paper introduces a new residual-type heuristic named *Residual Recombination Heuristic* (RRH) to address 1D-CSP and 1D-MCSP models. Using the rounded-down solution of the relaxed problem's output as an integer starting point, RRH consists of post-processing the remaining fractional entries by creating a *matrix of* *residual cutting patterns* instead of rounding these fractional entries, as occurs in the previous residual-type heuristics. The RRH procedure reconstructs and generates new integer cutting patterns by recombining the residual cutting patterns from this matrix. The RRH procedure uses the same cutting patterns generated by the relaxed problem as much as possible. We present the RRH for the 1D-CSP model as a prerequisite to understand the RRH for the 1D-MCSP model. Nevertheless, this paper's main contribution is to find feasible solutions for the 1D-MCSP model in a suitable computational time for real-world problem arising in industrial processes.

We point out the difference between our proposal and the idea of recombining cutting patterns introduced by Diegel et al. (1993) and a class of methods called *KOMBI*Foerster and Wäscher (2000). The latter methods aim to minimize the setup cost due to the switch between different cutting patterns. The main idea is to start from an initial solution to the 1D-CSP problem, considering minimal loss, and then identify pairs of different cutting patterns that can be replaced by a single cutting pattern, but using it twice. That is, the number of initial and final objects remains constant. On the other hand, the RRH algorithm aims to recombine the residual cutting patterns to reduce the number of objects used.

The outline of this work is as follows. Section 2 presents the formal descriptions and the mathematical models for the 1D-CSP and the 1D-MCSP. Section 3 describes the proposed residual recombination heuristic. Section 4 introduces the input data and the main parameters used in the computational experiments, whose results are also analyzed. Section 5 closes with the conclusions and the further steps of the research.

2 Mathematical models

In this Section we present the notation and the mathematical models for 1D-CSP and 1D-MCSP.

2.1 Mathematical model for the 1D-CSP

In this model, we consider a unique type of object of length L (of unlimited amount) from which we obtain pieces of type i, $i \in I = \{1, \dots, m\}$, with lengths ℓ_i. We denote by $\mathbf{d} \in \mathbb{Z}_+^m$ the vector of demand, where each element d_i represents the quantity of pieces of type i that must be produced.

The cutting pattern j, $j \in J = \{1, \dots, r\}$, is represented by $\mathbf{a}_j = (a_{1j}, \dots, a_{mj})^T$, where a_{ij} is the number of pieces of type i in the cutting pattern j. Each column j of the matrix $\mathbf{A} \in \mathbb{Z}_+^{m\times r}$ is composed of the cutting patterns $\mathbf{a}_j$. The vector of decision variables is $\mathbf{x} \in \mathbb{Z}_+^r$, and each element x_j represents the number of objects that are cut using the cutting pattern j. Finally, $\mathbf{1}$ is an r-dimensional vector containing ones. The mathematical model for the 1D-CSP is the following:

(**1D-CSP**) $\{\text{Min}\ \mathbf{1}^T\mathbf{x},\ \text{s.t.}\ \mathbf{A}\mathbf{x} = \mathbf{d}, \mathbf{x} \in \mathbb{Z}_+^r\}$.

The 1D-CSP mathematical model aims at minimizing the number of cut objects, fulfilling the demand.

Definition 1 An *maximal cutting pattern* (MCP) is a cutting pattern whose waste of material is smaller than the smallest piece available (Dyckhoff 1990), i.e., $L - \sum_{i=1}^{m} \ell_i a_{ij} < \min_{i \in I}\{\ell_i\}$ for some j.

Definition 2 A *feasible cutting pattern* is a cutting pattern that satisfies the inequality $\ell_1 a_{1j} + \ell_2 a_{2j} + \cdots + \ell_m a_{mj} \leq L$ and $\mathbf{a}_j \in \mathbb{Z}_+^r$.

A relaxed cutting stock problem (R-1D-CSP) is obtained by considering $\mathbf{x} \in \mathbb{R}_+^r$. Each solution of the R-1D-CSP usually provides a starting point for achieving good solutions for the 1D-CSP (Poldi and Arenales 2009).

2.2 Mathematical model for the 1D-MCSP

The 1D-MCSP considers various types of objects, each one with a particular length L_k, $k \in K = \{1, \dots, \varkappa\}$, and under a limited quantity. The objects not cut in a given period remain available in the next periods, and their storage costs are added. Moreover, the demand for pieces is known in a long-term planning horizon, divided into periods. In this sense, the pieces demanded for the future periods can be generated in previous periods. The anticipated pieces are stored until the delivery date, and their storage costs are considered. The 1D-MCSP mathematical model requires two more decision variables concerning the storage of objects and pieces. We consider the following notation:

Indices:

t: periods in the planning horizon; $t \in T = \{1, \dots, \tau\}$;
k: types of objects; $k \in K = \{1, \dots, \varkappa\}$;
j: cutting patterns; $j \in J_k = \{1, \dots, r_k\}$;
i: types of pieces; $i \in I = \{1, \dots, m\}$.

Parameters:

L_k: length of the object of type k;
o_{kt}: number of available objects of length L_k in period t;
ℓ_i: length of the piece of type i;
$\mathbf{d}_t$: vector for demand of pieces; each component d_{it} represents the number of piece of type i, cut in period t;
$\mathbf{a}_{jkt}$: vector of cutting patterns; each component a_{ijkt} represents the number of demanded pieces of type i in the jth cutting pattern cut from the object of type k, in period t;
c_{jkt}: cost of the material waste for the jth cutting pattern for the object of type k in period t;
h_{kt}: storage cost for the object of type k at the end of the period t;
σ_{it}: storage cost for the piece of type i at the end of the period t.

Decision variables:

x_{jkt}: number of objects of type k cut according to the cutting pattern j in period t;
$\mathbf{e}_t$: vector of stored pieces; each component e_{it} represents the number of pieces of type i stored at the end of the period t;
$\mathbf{w}_t$: vector of stored objects; each component w_{kt} represents the number of objects of type k stored at the end of the period t.

The following 1D-MCSP mathematical model was proposed by Poldi and Araujo (2016):

(***1D-MCSP***)

$$\text{Min} \quad \sum_{t\in T}\sum_{k\in K}\sum_{j\in J_k} c_{jkt}x_{jkt} + \sum_{t\in T}\sum_{i\in I}\sigma_{it}e_{it} + \sum_{t\in T}\sum_{k\in K} h_{kt}w_{kt} \tag{1}$$

s.t.

$$\sum_{k\in K}\sum_{j\in J_k} \mathbf{a}_{jkt}x_{jkt} + \mathbf{e}_{t-1} - \mathbf{e}_t = \mathbf{d}_t, \quad \forall t \in T; \tag{2}$$

$$\sum_{j\in J_k} x_{jkt} + w_{k,t-1} - w_{kt} = o_{kt}, \quad \forall k \in K, \quad \forall t \in T; \tag{3}$$

$$e_{i0} = 0, e_{i\tau} = 0, \quad \forall i \in I; \tag{4}$$

$$x_{jkt}, e_{it}, w_{kt} \in \mathbb{Z}_+, \quad i \in I, \quad k \in K, \quad j \in J_k, \quad t \in T. \tag{5}$$

The Objective Function (1) minimizes the costs of material waste and the storage of objects and pieces. The fulfillment of demand in each period is guaranteed by Eq. (2). In Constraints (3), the number of cut objects plus the stock balance in each period is equal to the number of available objects. Constraints (4) define the starting and ending stock levels, and Constraints (5) specify the allowed values for the variables.

If $x_{jkt}, w_{kt}, e_{it} \in \mathbb{R}_+$, $i \in I$, $j \in J_k$, $k \in K$, and $t \in T$, we obtain the corresponding relaxed multiperiod multiple lengths one-dimensional cutting stock problem (R-1D-MCSP).

3 Residual recombination heuristic

In this section, we introduce the new residual type-heuristic proposed in this work, the residual recombination heuristic (RRH). This heuristic starts from the relaxed solution obtained by the column generation procedure, composed of the integer and the fractional parts. Unlike other residual type-heuristics, that round the fractional entries of the relaxed solution, RRH's main idea is to create a matrix of residual cutting patterns from these fractional entries. From this matrix we reconstruct and create new feasible cutting patterns by recombining the residual parts among the patterns, without violating the restriction of meeting the demand.

In Sect. 3.1, we introduce an essential concept of the residual recombination heuristic, the matrix of residual cutting patterns, and the approach to generate integer cutting patterns from this matrix. In Sect. 3.2, we present the RRH algorithm for the 1D-CSP, and in Sect. 3.3, the corresponding extension for the 1D-MCSP.

3.1 Residual recombination heuristic foundations

In general, there are enormous possible cutting patterns in the CSP, which makes finding the optimal solution prohibitive in terms of computational resources required. To deal with this difficulty, Gilmore and Gomory (1961) proposed to solve the CSP using the simplex method with the column generation technique (CG) (Gilmore and Gomory 1961, 1963). When applying the column generation technique, only n cutting patterns are considered explicitly, with $n << r$ (where r is the number of all possible cutting patterns). At each simplex iteration, the best cutting pattern is determined by solving a sub-problem. This procedure is the basis of the RRH algorithm.

Let $\mathbf{x} \in \mathbb{R}^n_+$ and $\mathbf{A} \in \mathbb{Z}^{m \times n}_+$ be the output by solving the R-1D-CSP with the CG, so that $\mathbf{Ax} = \mathbf{d}$. Consider the solution $\mathbf{x}$ comprised of the integer and the fractional parts, that is $\mathbf{x} = \bar{\mathbf{x}} + \tilde{\mathbf{x}}$, where $\bar{\mathbf{x}} \in \mathbb{Z}^n_+$, and $\tilde{\mathbf{x}} \in [0,1)^n$. Thus, $\mathbf{Ax} = \mathbf{d}$ can be represented as $\mathbf{A}\bar{\mathbf{x}} + \mathbf{A}\tilde{\mathbf{x}} = \mathbf{d}$, or equivalently, $\mathbf{A}\tilde{\mathbf{x}} = \mathbf{d} - \mathbf{A}\bar{\mathbf{x}}$. That means an integer feasible solution $\mathbf{A}\bar{\mathbf{x}}$ covers part of the demand $\mathbf{d}$, and a residual demand remains, since $\mathbf{A}\tilde{\mathbf{x}}$ is not a feasible solution, as $\tilde{\mathbf{x}}$ is non-integer. To obtain a vector of integer entries, we rewrite $\mathbf{A}\tilde{\mathbf{x}}$ as follows:

$$\begin{aligned}\mathbf{A}\tilde{\mathbf{x}} &= \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \vdots \\ \tilde{x}_n \end{bmatrix} \\ &= \begin{bmatrix} a_{11}\tilde{x}_1 & \cdots & a_{1n}\tilde{x}_n \\ \vdots & \ddots & \vdots \\ a_{m1}\tilde{x}_1 & \cdots & a_{mn}\tilde{x}_n \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \tilde{\mathbf{A}}\mathbf{1},\end{aligned} \tag{6}$$

where $\tilde{\mathbf{A}}$ is the *matrix of the residual cutting patterns*, and $\mathbf{1}$ is a vector of length m composed by ones. For example, a feasible cutting pattern $\mathbf{a}_j^T = (1,0,2)$ from $\mathbf{A}$ and its respective fractional $\tilde{x}_j = 0.75$ from $\tilde{\mathbf{x}}$ can be represented as $\mathbf{a}_j^T\tilde{x}_j = (1 \times 0.75, 0 \times 0.75, 2 \times 0.75) \equiv \tilde{\mathbf{a}}_j^T = (0.75, 0, 1.5)$, where $\tilde{\mathbf{a}}_j^T$ is the jth column of $\tilde{\mathbf{A}}$.

Note that, because $\mathbf{A}\tilde{\mathbf{x}} = \tilde{\mathbf{A}}\mathbf{1}$, we can rewrite $\mathbf{A}\bar{\mathbf{x}} + \mathbf{A}\tilde{\mathbf{x}} = \mathbf{d}$ as $\mathbf{A}\bar{\mathbf{x}} + \tilde{\mathbf{A}}\mathbf{1} = \mathbf{d}$. However, if there exist non-integer numbers in $\tilde{\mathbf{A}}$, $(\bar{\mathbf{x}}, \mathbf{1})$ is not a feasible solution, and the procedure will aim to find a matrix $\mathbf{B} \in \mathbb{Z}^{m \times n}_+$ such that $\tilde{\mathbf{A}}\mathbf{1} = \mathbf{B}\mathbf{1}$, and then $\mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\mathbf{1} = \mathbf{d}$ becomes a feasible solution for the 1D-CSP.

From the equality $\mathbf{A}\bar{\mathbf{x}} + \tilde{\mathbf{A}}\mathbf{1} = \mathbf{d}$, we obtain $\tilde{\mathbf{A}}\mathbf{1} = \mathbf{d} - \mathbf{A}\bar{\mathbf{x}}$. As $\mathbf{d}$ is a vector of integer numbers, as well as $\mathbf{A}\bar{\mathbf{x}}$, $\mathbf{d} - \mathbf{A}\bar{\mathbf{x}}$ must also be an integer vector. Therefore $\tilde{\mathbf{A}}\mathbf{1}$ is an integer vector too, that we will call $\mathbf{d}^r$. Thus, each row i of $\tilde{\mathbf{A}}\mathbf{1}$,

$\tilde{a}_{i1} + \tilde{a}_{i2} + \cdots + \tilde{a}_{in}$, produces an integer number d_i^r. By the theory of integer partition (Andrews and Eriksson 2004), we have guaranteed that all integer numbers can be written (in many ways) as a sum of positive integer numbers. In this sense, we can rewrite d_i^r in many ways, such that $d_i^r = b_{i1} + b_{i2} + \cdots + b_{in}$ and $(b_{i1}, b_{i2}, \ldots, b_{in})^T \in \mathbb{Z}_+^n$. Algorithm 1 shows how we rewrite each row of $\tilde{\mathbf{A}}$, to obtain $\mathbf{B}$, by recombining the fractional parts of the elements $\tilde{a}_{ij}$. In Algorithm 1, consider $\lceil . \rceil$ the rounding symbol for the ceiling and $\lfloor . \rfloor$ for the floor.

Algorithm 1 Recombination technique

```
Input: $\tilde{\mathbf{A}}$
Initialization $\mathbf{B} = 0_{m \times n}$ (each $\mathbf{b}_i = \mathbf{0}$)

for $i = 1, \ldots, m$ do
    $q \leftarrow 1$
    $p \leftarrow n$
    while $q < p$ do
        $z \leftarrow \lceil \tilde{a}_{iq} \rceil - \tilde{a}_{iq}$
        if $z \leq \tilde{a}_{ip}$ then
            $\tilde{a}_{iq} \leftarrow \lceil \tilde{a}_{iq} \rceil$
            $\tilde{a}_{ip} \leftarrow \tilde{a}_{ip} - z$
            $q + 1$
        else
            $\tilde{a}_{iq} \leftarrow \tilde{a}_{iq} + \tilde{a}_{ip}$
            $\tilde{a}_{ip} \leftarrow \lfloor \tilde{a}_{ip} \rfloor$
            $p \leftarrow p - 1$
        end if
    end while
    $\mathbf{b}_i \leftarrow (\tilde{a}_{i1}, \ldots, \tilde{a}_{in})$
end for
Output: $\mathbf{B}$
```

In Algorithm 1, the same amount removed from an element of matrix $\tilde{\mathbf{A}}$ is added to another in the same row i. The increased element is increased up to achieve its ceiling, while the decreased element can be decreased up to zero. For instance, assuming that row i of $\tilde{\mathbf{A}}$ is $\tilde{\mathbf{a}}_i^T = (0.75, 0.5, 0.6, 0.15)$, $q = 1$, and $p = 4$. As $q < p$ we compute $z = \lceil 0.75 \rceil$ - 0.75 = 0.25. Since $z = 0.25 > \tilde{a}_{14} = 0.15$, the algorithm executes the else statement: $\tilde{a}_{i1} = 0.75 + 0.15 = 0.9$, $\tilde{a}_{i4} = \lfloor 0.15 \rfloor = 0$, and $p = 4 - 1 = 3$ (note that q keeps its value 1). Next, the while statement is true ($1 < 3$), then we compute the new $z = \lceil 0.9 \rceil - 0.9 = 0.1$. Because $z = 0.1 < \tilde{a}_{i3} = 0.60$, the if statement is true, then we proceed to $\tilde{a}_{i1} = \lceil 0.9 \rceil = 1$, $\tilde{a}_{i3} = 0.6 - 0.1 = 0.5$, and $q = 1 + 1 = 2$. Finally, the while statement executes ($2 < 3$), $z = \lceil 0.5 \rceil - 0.5 = 0.5$, $z = 0.5$ is equal to $\tilde{a}_{i3} = 0.5$, then $\tilde{a}_{i2} = 1$, $\tilde{a}_{i3} = 0.5 - 0.5 = 0$, $q = 3$. The while statement is not executed this time because $q > p$ and we finish the procedure with $\mathbf{b}_i = (1, 1, 0, 0)^T$. Thus, we replace the initial vector $\tilde{\mathbf{a}}_i = (0.75, 0.5, 0.6, 0.15)^T$ by $\mathbf{b}_i = (1, 1, 0, 0)^T$, maintaining the sum of $\mathbf{b}_i$ equals to the sum of $\tilde{\mathbf{a}}_i$.

It is important to note that, in Algorithm 1, z represents the value that $\tilde{a}_{iq}$ misses to reach its corresponding upper integer. In the numerical example above, to make $\tilde{a}_{i1} = 0.75$ reaching its upper integer value, it is necessary to add 0.25, which corresponds to the first z value. Therefore, the *if* statement verifies whether the $\tilde{a}_{iq}$ missing value (z) is less than $\tilde{a}_{ip}$. If true, then $\tilde{a}_{iq}$ achieves its upper integer value ($\lceil \tilde{a}_{iq} \rceil$). If false, then, $\tilde{a}_{iq} = \tilde{a}_{iq} + \tilde{a}_{ip}$, which shall always be less than its upper integer value ($\lceil \tilde{a}_{iq} \rceil$). Thus, $\tilde{a}_{iq}$ will achieve at most its ceiling.

Among the many ways for rewriting d_i^r, we developed the approach proposed by Algorithm 1 because it ensures that all $\mathbf{b}_j^T$ meets the inequality necessary to be a feasible cutting pattern, i.e. $\sum_{i=1}^m \ell_i b_{ij} < L, \forall j$.

Consider $\tilde{\mathbf{x}} \in [0,1)^n$ the fractional part of $\mathbf{x}$ and $\mathbf{a}_j, \forall j$, the feasible cutting patterns of $\mathbf{A}$ obtained by the column generation procedure. From Eq. (6), as we multiply a positive integer number a_{ij} by a number between zero and one $\tilde{x}_j$, $\tilde{a}_{ij} = a_{ij}\tilde{x}$, we have that $\tilde{a}_{ij} \leq a_{ij}, \forall i,j$. Then, by rounding up $\lceil \tilde{a}_{ij} \rceil$ or rounding down $\lfloor \tilde{a}_{ij} \rfloor$, we have $\lfloor \tilde{a}_{ij} \rfloor \leq \lceil \tilde{a}_{ij} \rceil \leq a_{ij}, \forall i,j$. If we multiply both sides of the inequality by an integer positive number ℓ_i, we maintain $\ell_i \lfloor \tilde{a}_{ij} \rfloor \leq \ell_i \lceil \tilde{a}_{ij} \rceil \leq \ell_i a_{ij}, \forall i,j$. Since the last inequality is ensured for all i in each j, we can sum all elements of $\lfloor \tilde{\mathbf{a}}_j \rfloor$, $\lceil \tilde{\mathbf{a}}_j \rceil$, and $\mathbf{a}_j$. Since $\mathbf{a}_j, \forall j$, a feasible cutting pattern, we know that $\sum_{i \in I} \ell_i a_{ij} \leq L$. Keeping the inequality

$$\sum_{i \in I} \ell_i \lfloor \tilde{a}_{ij} \rfloor \leq \sum_{i \in I} \ell_i \lceil \tilde{a}_{ij} \rceil \leq \sum_{i \in I} \ell_i a_{ij} \leq L, \forall j.$$

Finally, since the elements of $\mathbf{b}_i$ are obtained from the $\lfloor \tilde{a}_j \rfloor$ or $\lceil \tilde{a}_j \rceil$, we have that $\sum_{i=1}^m \ell_i b_{ij} \leq L, \forall j$.

In summary, the basis of RRH is

Step (1): Solve R-1D-CSP by column generation, obtaining $\mathbf{A}\overline{\mathbf{x}} + \mathbf{A}\tilde{\mathbf{x}} = \mathbf{d}$;
Step (2): Apply Eq. (6) in the solution of **Step (1)**, obtaining $\mathbf{A}\overline{\mathbf{x}} + \tilde{\mathbf{A}}\mathbf{1} = \mathbf{d}$;
Step (3): Apply Algorithm 1 in the solution of **Step (2)**, obtaining a feasible solution for the problem: $\mathbf{A}\overline{\mathbf{x}} + \mathbf{B}\mathbf{1} = \mathbf{d}$.

3.2 Residual recombination heuristic for the 1D-CSP model

By applying **Steps (1)–(3)**, we obtain a feasible solution $\mathbf{A}\overline{\mathbf{x}} + \mathbf{B}\mathbf{1} = \mathbf{d}$ for the 1D-CSP problem. However, Algorithm 1 occasionally produces non-maximal cutting patterns (NMCP): $L - \boldsymbol{\ell}\mathbf{b}_j > \min_{i \in I}\{\ell_i\}$, leading to significant material losses. To improve the generation process of cutting patterns, we sort the elements of $\mathbf{x}$ in descending order (together with the corresponding columns of $\mathbf{A}$ to maintain the equality $\mathbf{A}\mathbf{x} = \mathbf{d}$), before applying Algorithm 1. If, nevertheless, some cutting pattern, e.g. the jth cutting pattern, is still non-efficient, it is removed by assigning the value zero in the jth position of the $\mathbf{1}$ vector, leading to a new vector $\mathbf{y} = (1, \dots, 0, \dots, 1)$ in the place of the $\mathbf{1}$ vector. Since $\mathbf{B}\mathbf{1} \geq \mathbf{B}\mathbf{y}$, the demand $\mathbf{d}$ is no longer fully met, and a residual demand $\mathbf{d}^r = \mathbf{d} - (\mathbf{A}\overline{\mathbf{x}} + \mathbf{B}\mathbf{y})$ arises. Thus, we update the demand $\mathbf{d} = \mathbf{d}^r$; and reapply **Steps (1)–(3)** until the demand is fully met or until the RRH does not find any maximal cutting pattern ($\mathbf{y} = \mathbf{0}$). In the latter case, we apply the greedy exhaustive repetition proposed by Hinxman (1980) to meet the residual demand. The complete RRH steps for obtaining a feasible solution for the 1D-CSP are presented in Algorithm 2.

Algorithm 2 RRH for the 1D-CSP

```
Input: $\mathbf{d} \in \mathbb{Z}_+^m$, $\boldsymbol{\ell} \in \mathbb{Z}_+^m$, $L$
Initialization $\mathbf{y} \leftarrow \mathbf{1}$

  while $\mathbf{d} \neq \mathbf{0}$, do
    $\mathbf{A}, \mathbf{x} \leftarrow$ solve the R-1D-CSP by CG using inputs $\mathbf{d}$, $\boldsymbol{\ell}$ and $L$
    $\mathbf{A}\bar{\mathbf{x}}, \mathbf{A}\tilde{\mathbf{x}} \leftarrow$ split $\mathbf{x}$ into integer and fractional parts
    $\mathbf{A}\tilde{\mathbf{x}} \leftarrow$ sort $\tilde{\mathbf{x}}$ in descending order together with the corresponding columns of $\mathbf{A}$
    $\tilde{\mathbf{A}} \leftarrow$ apply Equation (6) using input $\mathbf{A}\tilde{\mathbf{x}}$
    $\mathbf{B} \leftarrow$ apply Algorithm 1 using input $\tilde{\mathbf{A}}$
    for $j \in J$ do
      if $\mathbf{b}_j$ is a NMCP then
        $y_j \leftarrow 0$
      end if
    end for
    $[\mathbf{A}|\mathbf{B}], (\bar{\mathbf{x}}|\mathbf{y}) \leftarrow$ save current solution
    if $\mathbf{y} = \mathbf{0}$ then
      $\mathbf{G}, \mathbf{x}^g \leftarrow$ solve the problem using the greedy exhaustive repetition (Hinxman [22])
      $[\mathbf{A}|\mathbf{B}|\mathbf{G}], (\bar{\mathbf{x}}|\mathbf{y}|\mathbf{x}^g) \leftarrow$ save the solution
      $\mathbf{d} \leftarrow \mathbf{0}$
    else if $\mathbf{y} = \mathbf{1}$ then
      $\mathbf{d} \leftarrow \mathbf{0}$
    else
      $\mathbf{d} \leftarrow \mathbf{d}^r$
    end if
  end while

  Output: $[\mathbf{A}|\mathbf{B}|\mathbf{G}], (\bar{\mathbf{x}}|\mathbf{y}|\mathbf{x}^g)$
```

3.3 Residual recombination heuristic for the 1D-MCSP model

For the 1D-MCSP, the residual recombination heuristic finds integer solutions for each period. Let $t \in T = \{1, \dots, \tau\}$, the main steps of RRH for the 1D-MCSP are shown in Fig. 1. Following we detail the logic behind each step, necessary to understand the reason why RRH does not violate the constraints of the mathematical model. Algorithm 3 presents the implementation of RRH for 1D-MCSP.

Step (1) Solve the R-1D-MCSP:

For a specific period t, the optimal solution for the R-1D-MCSP employing column generation is $(\mathbf{x}_t, \mathbf{e}_t, \mathbf{w}_t, \mathbf{A}_t)$, where $x_{jkt}, w_{kt}, e_{it} \in \mathbb{R}_+$, $i \in I$, $j \in J_k$, $k \in K$, $\mathbf{x}_t = (\mathbf{x}_{1t}|\mathbf{x}_{2t}| \cdots |\mathbf{x}_{\varkappa t})$, the matrix $\mathbf{A}_t = [\mathbf{A}_{1t}|\mathbf{A}_{2t}| \cdots |\mathbf{A}_{\varkappa t}]$ and each column $\mathbf{a}_{jkt} \in \mathbb{Z}_+^n$ is a feasible cutting pattern.

Step (2) Obtain the matrix of residual cutting patterns:

The demand of period t is met by Eq. (2): $\sum_{k \in K} \mathbf{A}_{kt}\mathbf{x}_{kt} + \mathbf{e}_{t-1} - \mathbf{e}_t = \mathbf{d}_t$, equivalent to $\sum_{k \in K} \mathbf{A}_{kt}\mathbf{x}_{kt} = \mathbf{d}_t - \mathbf{e}_{t-1} + \mathbf{e}_t$. By splitting the solution into the integer and the fractional parts $\mathbf{x}_t = \bar{\mathbf{x}}_{kt} + \tilde{\mathbf{x}}_{kt}$ and $\mathbf{e}_t = \bar{\mathbf{e}}_t + \tilde{\mathbf{e}}_t$, where $\bar{\mathbf{x}}_{kt} \in \mathbb{Z}_+^{n_k}$, $\bar{\mathbf{e}}_t \in \mathbb{Z}_+^m$, $\tilde{\mathbf{x}}_{kt} \in [0, 1)^{n_k}$, and $\tilde{\mathbf{e}}_t \in [0, 1)^m$ we have

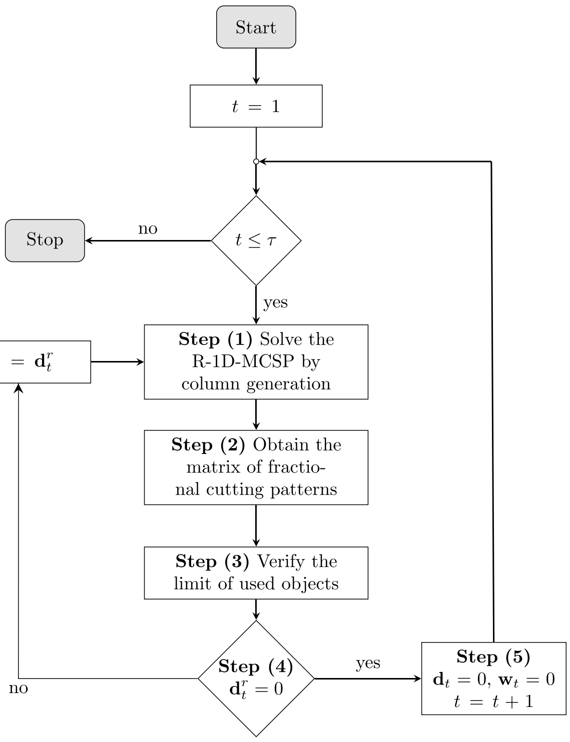

Start
$t = 1$
$t \leq \tau$
no
Stop
yes
Step (1) Solve the R-1D-MCSP by column generation
$\mathbf{d}_t = \mathbf{d}_t^r$
Step (2) Obtain the matrix of fractional cutting patterns
Step (3) Verify the limit of used objects
Step (4) $\mathbf{d}_t^r = 0$
no
yes
Step (5) $\mathbf{d}_t = 0$, $\mathbf{w}_t = 0$ $t = t + 1$

Fig. 1 Flowchart of the main steps of RRH for 1D-MCSP

$$\sum_{k \in K} (\mathbf{A}_{kt}\bar{\mathbf{x}}_{kt} + \mathbf{A}_{kt}\tilde{\mathbf{x}}_{kt}) = \mathbf{d}_t - \mathbf{e}_{t-1} + \bar{\mathbf{e}}_t + \tilde{\mathbf{e}}_t. \tag{7}$$

Note that $\mathbf{e}_{t-1}$ is integer, because, for $t = 1$, $e_{t-1} = e_0 = 0$, from Constraints (4), and for $t > 1$, it was converted to an integer by the heuristic from the previous period.

From Eq. (7), we obtain

$$\sum_{k \in K} \mathbf{A}_{kt}\tilde{\mathbf{x}}_{kt} = \left(\mathbf{d}_t - \sum_{k \in K} \mathbf{A}_{kt}\bar{\mathbf{x}}_{kt} - \mathbf{e}_{t-1} + \bar{\mathbf{e}}_t \right) + \tilde{\mathbf{e}}_t. \tag{8}$$

Let the residual demand $\mathbf{d}_t^r$ be the part of the demand that is not covered by the integer solution, i.e., $\mathbf{d}_t^r = (\mathbf{d}_t - \sum_{k \in K} \mathbf{A}_{kt}\bar{\mathbf{x}}_{kt} - \mathbf{e}_{t-1} + \bar{\mathbf{e}}_t)$. Note that, because $\mathbf{d}_t^r$, $\sum_{k \in K} \mathbf{A}_{kt}\bar{\mathbf{x}}_{kt}$, $\mathbf{e}_{t-1}$, and $\bar{\mathbf{e}}_t$ are vectors of integer elements, $\mathbf{d}_t^r$ is also a integer-valued vector.

Each $\mathbf{A}_{kt}\tilde{\mathbf{x}}_{kt}$ is transformed into $\tilde{\mathbf{A}}_{kt}\mathbf{1}_{kt}$ by applying Eq. (6) to obtain the matrix of residual cutting patterns, i.e., $\sum_{k \in K} \mathbf{A}_{kt}\tilde{\mathbf{x}}_{kt} = \sum_{k \in K} \tilde{\mathbf{A}}_{kt}\mathbf{1}_{kt}$. Then, Eq. (8) is replaced by

$$\sum_{k\in K} \tilde{\mathbf{A}}_{kt}\mathbf{1}_{kt} = \mathbf{d}_t^r + \tilde{\mathbf{e}}_t. \tag{9}$$

Each row i of Eq. (9) is represented as $\sum_{k\in K}(\tilde{a}_{i1})_{kt} + \cdots + \sum_{k\in K}(\tilde{a}_{in_k})_{kt} = d_{it}^r + \tilde{e}_{it}$. Because d_{it}^r is integer, it can be rewritten as a sum of the $(n_k - 1)$ integer elements $d_{it}^r = \sum_{k\in K}(b_{i1})_{kt} + \cdots + \sum_{k\in K}(b_{i(n_k-1)})_{kt}$, $b_{ijkt} \in \mathbb{Z}_+$, plus the fractional $\tilde{b}_{in_kkt} = \tilde{e}_{it}$, $\tilde{b}_{in_kkt} \in [0, 1)$. That means:

$$\begin{aligned}
&\sum_{k\in K}(\tilde{a}_{i1})_{kt} + \cdots + \sum_{k\in K}(\tilde{a}_{in_k})_{kt} \\
&\quad = \sum_{k\in K}(b_{i1})_{kt} + \cdots + \sum_{k\in K}(b_{i(n_k-1)})_{kt} + (\tilde{b}_{in_k})_{kt} \\
&\quad = d_{it}^r + \tilde{e}_{it}.
\end{aligned} \tag{10}$$

The process of rewriting $d_{it}^r + \tilde{e}_{it}$ as in Eq. (10) is by applying Algorithm 1 in $\tilde{\mathbf{A}}_t = [\tilde{\mathbf{A}}_{1t}|\tilde{\mathbf{A}}_{2t}| \cdots |\tilde{\mathbf{A}}_{\varkappa t}]$. Then, we obtain $\mathbf{B}_t = [\mathbf{B}_{1t}|\mathbf{B}_{2t}| \cdots |\mathbf{B}_{\varkappa t}]$, where in each row i, only one element is non-integer. Finally, we proceed to round down the fractional element $\lfloor \tilde{b}_{in_kkt} \rfloor$, which corresponds to rounding down the $\lfloor \tilde{e}_{it} \rfloor$ (note that $\lfloor \tilde{e}_{it} \rfloor = 0$), obtaining the following:

$$\begin{aligned}
&\sum_{k\in K}(\tilde{a}_{i1})_{kt} + \cdots + \sum_{k\in K}(\tilde{a}_{in_k})_{kt} \\
&\quad \geq \sum_{k\in K}(b_{i1})_{kt} + \cdots + \sum_{k\in K}(b_{i(n_k-1)})_{kt} + \lfloor (\tilde{b}_{in_k})_{kt} \rfloor \\
&\quad = d_{it}^r + \lfloor \tilde{e}_{it} \rfloor.
\end{aligned} \tag{11}$$

By rounding down the fractional elements of matrix $\mathbf{B}_t$, we obtain matrix $\overline{\mathbf{B}}_t = [\overline{\mathbf{B}}_{1t}|\overline{\mathbf{B}}_{2t}| \cdots |\overline{\mathbf{B}}_{\varkappa t}]$, where each element $\overline{\mathbf{b}}_{jkt} \in \mathbb{Z}_+^n$. The solution is composed of integer-valued vectors:

$$\sum_{k\in K}(\mathbf{A}_{kt}\bar{\mathbf{x}}_{kt} + \overline{\mathbf{B}}_{kt}\mathbf{1}_{kt}) + \mathbf{e}_{t-1} - \bar{\mathbf{e}}_t = \mathbf{d}_t. \tag{12}$$

Since Algorithm 1 occasionally produces non-maximal cutting patterns (NMCP), we remove the corresponding value of the NMCP in the $\mathbf{1}_{kt}^T$, and represent the new vector as $\mathbf{y}_{kt} = (1, \dots, 0, \dots, 1| \dots |1, \dots, 0, \dots, 1)$.

Step (3) Verify the limit of used objects:

Since the objects are available in limited quantities, it is necessary to verify for the current period if the following equation holds:

$$\sum_{j\in J_k}(\bar{x}_{jkt} + y_{jkt}) > o_{kt},\ k \in K. \tag{13}$$

If true, the heuristic removes the cutting pattern by assigning the value zero in the $\mathbf{y}_{jkt}$, starting from the last pattern $j = J_k$, until the inequality becomes false. Next, it updates the number of available objects in the current period: $\mathbf{w}_t = \mathbf{w}_t - \sum_{k\in K}(\bar{\mathbf{x}}_{kt} + \mathbf{y}_{kt})$. Note that the heuristic finds an integer solution for the variable $\mathbf{w}_t$, because $\bar{\mathbf{x}}_{kt}$ and $\mathbf{y}_{kt}, k \in K$, are integers.

After that, it is necessary to verify if $\mathbf{y}_{kt} = 0, k \in K$. If so, the greedy exhaustive repetition proposed by Hinxman (1980) is applied.

Step (4) calculate the residual demand $\mathbf{d}_t^r$:

The residual demand of the current period is calculated: $\mathbf{d}_t^r = \mathbf{d}_t - (\sum_{k\in K}(\mathbf{A}_{kt}\bar{\mathbf{x}}_{kt} + \overline{\mathbf{B}}_{kt}\mathbf{y}_{kt}) + \mathbf{e}_{t-1} - \bar{\mathbf{e}}_t)$. If $\mathbf{d}_t^r = 0$, **Steps (1)–(3)** must be repeated.

Step (5) Update the parameters:

Save the feasible solution for the current period: $(\mathbf{x}_t, \mathbf{y}_t, \mathbf{A}_t, \overline{\mathbf{B}}_t, \mathbf{e}_t, \mathbf{w}_t)$. We reset the values of the actual demand and the stock of objects, $\mathbf{d}_t = 0$ and $\mathbf{w}_t = 0$, making **Step (1)** stop solving the 1D-MCSP for the previous periods of $t + 1$.

Algorithm 3 presents the complete RRH procedure for the 1D-MCSP.

Algorithm 3 RRH for the 1D-MCSP

```
Input: $\mathbf{d}_t \in \mathbb{Z}_+^m$, $\boldsymbol{\ell} \in \mathbb{Z}_+^m$, $L_k, k \in K$, $o_{kt}$, $h_{kt}$, $\alpha_{it}$
Initialization $\mathbf{y}_t \leftarrow \mathbf{1}$

  for $t \in T$ do
    while $\mathbf{d}_t^r \neq \mathbf{0}$, do
      $\mathbf{A}_t, \mathbf{x}_t, \mathbf{e}_t, \mathbf{w}_t \leftarrow$ solve the R-1D-MCSP by CG using inputs $\mathbf{d}_t$, $\boldsymbol{\ell}$, $L_k$, $o_{kt}$, $h_{kt}$, $\alpha_{it}$
      $\mathbf{A}_t\bar{\mathbf{x}}_t, \mathbf{A}_t\tilde{\mathbf{x}}_t, \bar{\mathbf{e}}_t, \tilde{\mathbf{e}}_t \leftarrow$ split $\mathbf{x}_t$ and $\mathbf{e}_t$ into integer and fractional parts
      $\mathbf{A}_t\tilde{\mathbf{x}}_t \leftarrow$ sort $\tilde{\mathbf{x}}_t$ in descending order together with the corresponding columns of $\mathbf{A}_t$
      $\tilde{\mathbf{A}}_t \leftarrow$ apply Equation (6) using input $\mathbf{A}_t\tilde{\mathbf{x}}_t$
      $\mathbf{B}_t \leftarrow$ apply Algorithm 1 using input $\tilde{\mathbf{A}}_t$
      $\overline{\mathbf{B}}_t \leftarrow \lfloor \tilde{b}_{in_k kt} \rfloor, k \in K$, in $\mathbf{B}_t$
      for $j \in J_k$ do
        if $\mathbf{b}_j$ is a NMCP then
          $y_j \leftarrow 0$
        end if
      end for
      for $k \in K$ do
        $q \leftarrow J_k$
        while $(\bar{\mathbf{x}}_{kt} + \mathbf{y}_{kt}) > o_{kt}$ do
          $\mathbf{y}_{qkt} \leftarrow 0$
          $q \leftarrow q - 1$
        end while
        $\mathbf{w}_t \leftarrow \mathbf{w}_t - \sum_{k\in K} (\bar{\mathbf{x}}_{kt} + \mathbf{y}_{kt})$
      end for
      $[\mathbf{A}_t|\overline{\mathbf{B}}_t], (\bar{\mathbf{x}}_t|\mathbf{y}_t) \leftarrow$ save the current solution
      if $\mathbf{y}_t = \mathbf{0}$ then
        $\mathbf{G}_t, \mathbf{x}_t^g \leftarrow$ solve the problem using the greedy exhaustive repetition (Hinxman [22])
        $[\mathbf{A}_t|\overline{\mathbf{B}}_t|\mathbf{G}_t], (\bar{\mathbf{x}}_t|\mathbf{y}_t|\mathbf{x}_t^g) \leftarrow$ save the solution
        $\mathbf{d}_t^r \leftarrow \mathbf{0}$
      else if $\mathbf{y}_t = \mathbf{1}$ then
        $\mathbf{d}_t^r \leftarrow \mathbf{0}$
      else
        $\mathbf{d}_t^r \leftarrow \mathbf{d}_t - (\sum_{k\in K} (\mathbf{A}_{kt}\bar{\mathbf{x}}_{kt} + \overline{\mathbf{B}}_{kt}\mathbf{y}_{kt}) + \mathbf{e}_{t-1} - \bar{\mathbf{e}}_t)$
      end if
    end while

  end for
  Output: $[\mathbf{A}_t|\overline{\mathbf{B}}_t|\mathbf{G}_t], (\bar{\mathbf{x}}_t|\mathbf{y}_t|\mathbf{x}_t^g)$
```

3.3.1 Numerical example

We explain the RRH procedure for the 1D-MCSP with an illustrative numerical example. We assume $\tau = 3$ (number of periods), $\varkappa = 2$ (object type), and $m = 3$ (piece type). The stock objects and demanded piece data are shown in Tables 1 and 2.

We present the procedure to find the integer solution for the first period ($t = 1$). Consider the R-1D-MCSP solution shown in Table 3.

We have $\sum_{k\in K} \mathbf{A}_{k1}\mathbf{x}_{k1} = (39.2727, 21.1515, 20)^T$, which meet the demand for the first period $\mathbf{d}_1 = (35, 20, 20)^T$, and generated stored pieces $\mathbf{e}_1 = (4.2725, 1.1515, 0)^T$. We split the solution into integer $\bar{\mathbf{x}}_{k1} = (0, 1, 5|10)^T$ and fractional $\tilde{\mathbf{x}}_{k1} = (0.1515, 0.2725, 0.5757|0)^T$, also the stored pieces $\bar{\mathbf{e}}_1 = (4, 1, 0)^T$ and $\tilde{\mathbf{e}}_1 = [0.2725, 0.1515, 0]^T$. We compute $\sum_{k\in K} \mathbf{A}_{k1}\bar{\mathbf{x}}_{k1} = (37, 20, 18)^T$; thus, by Eq. (8):$\sum_{k\in K} \mathbf{A}_{kt}\tilde{\mathbf{x}}_{kt} = (\mathbf{d}_t - \sum_{k\in K} \mathbf{A}_{kt}\bar{\mathbf{x}}_{kt} - \mathbf{e}_{t-1} + \bar{\mathbf{e}}_t) + \tilde{\mathbf{e}}_t$, we have

$$\begin{bmatrix} 2.2725 \\ 1.1515 \\ 2 \end{bmatrix} = \left(\begin{bmatrix} 35 \\ 20 \\ 20 \end{bmatrix} - \begin{bmatrix} 37 \\ 20 \\ 18 \end{bmatrix} + \begin{bmatrix} 4 \\ 1 \\ 0 \end{bmatrix} \right) + \begin{bmatrix} 0.2725 \\ 0.1515 \\ 0 \end{bmatrix}. \tag{14}$$

We call $\mathbf{d}_1^r = (2, 1, 2)^T$ the part of Eq. (14) that is in brackets. We save the solution $\bar{\mathbf{x}}_{k1}^T$ and $\bar{\mathbf{e}}_1^T$. Because $\tilde{\mathbf{x}}_{21} = 0$, in the next steps we deal only with $\tilde{\mathbf{x}}_{11}$. We sort $\tilde{\mathbf{x}}_{11}$ in descending order together with the corresponding columns of $\mathbf{A}_{11}$. Each $\mathbf{A}_{11}\tilde{\mathbf{x}}_{11}$ is transformed into $\tilde{\mathbf{A}}_{11}\mathbf{1}_{11}$ by applying Eq. (6) to obtain the matrix of residual cutting patterns:

$$\mathbf{A}_{11}\tilde{\mathbf{x}}_{11} = \begin{bmatrix} 3 & 2 & 0 \\ 2 & 0 & 0 \\ 1 & 3 & 4 \end{bmatrix} \begin{bmatrix} 0.5757 \\ 0.2725 \\ 0.1515 \end{bmatrix} = \begin{bmatrix} 1.7272 & 0.5450 & 0 & 0 \\ 1.1515 & 0 & 0 & 0 \\ 0.5757 & 0.8175 & 0.6060 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \tilde{\mathbf{A}}_{11}\mathbf{1}_{11}. \tag{15}$$

Note that Eq. (8) holds: $\sum_{k\in K} \tilde{\mathbf{A}}_{11}\mathbf{1}_{11} = \mathbf{d}_1^r + \tilde{\mathbf{e}}_1 \rightarrow (2.2725, 1.1515, 2)^T = (2, 1, 2)^T + (0.2725, 0.1515, 0)^T$.

By applying Algorithm 1, we obtain $\mathbf{B}_{11}\mathbf{1}_{11} = \mathbf{d}_1^r + \tilde{\mathbf{e}}_1$:

$$\mathbf{B}_{11}\mathbf{1}_{11} = \begin{bmatrix} 2 & 0.2725 & 0 & 0 \\ 1.1515 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.2725 \\ 0.1515 \\ 0 \end{bmatrix}. \tag{16}$$

Finally, we proceed to round down the fractional element $\lfloor \tilde{b}_{in_k kt} \rfloor$, which corresponds to rounding down the $\lfloor \tilde{e}_{i1} \rfloor$, to obtain matrix $\overline{\mathbf{B}}_{k1}$ (we exclude the null columns):

$$\overline{\mathbf{B}}_{11}\mathbf{1}_{11} = \begin{bmatrix} 2 & \lfloor 0.2725 \rfloor \\ \lfloor 1.1515 \rfloor & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} + \begin{bmatrix} \lfloor 0.2725 \rfloor \\ \lfloor 1.1515 \rfloor \\ 0 \end{bmatrix} \tag{17}$$

Table 1 Stock object data

Object type (k)	Stock availability		
	Period 1	Period 2	Period 3
1	10	5	4
2	10	3	3

Table 2 Demanded piece data

Piece type (i)	Demand		
	Period 1	Period 2	Period 3
1	35	20	1
2	20	10	10
3	20	5	10

Table 3 Solution for $t = 1$

Object type (k)	x_{jk1}	Cutting pattern ($\mathbf{A}_{k1}^T$)
1	0.1515	(0, 0, 4)
1	1.2725	(2, 0, 3)
1	5.5757	(3, 2, 1)
2	10	(2, 1, 1)

Note that rounding down $\tilde{\mathbf{e}}_1$, does not affect the quantity of pieces produced to meet the demand of the first period ($\mathbf{d}_1$). Thus, Eq. (12) holds for $t = 1$: $\sum_{k\in K}(\mathbf{A}_{k1}\bar{\mathbf{x}}_{k1} + \overline{\mathbf{B}}_{k1}\mathbf{1}_{k1}) + \mathbf{e}_0 - \bar{\mathbf{e}}_1 = \mathbf{d}_1$ (because $\mathbf{e}_0 = \mathbf{0}$ we omit it):

$$\left(\begin{bmatrix}37\\20\\18\end{bmatrix} + \begin{bmatrix}2\\1\\2\end{bmatrix}\right) - \begin{bmatrix}4\\1\\0\end{bmatrix} = \begin{bmatrix}35\\20\\20\end{bmatrix} \tag{18}$$

In summary, it was necessary $x_{11} = 8$, $x_{21} = 10$, which does not exceed the number of objects available for $t = 1$, and it is a feasible solution for the first period.

4 Computational experiments

We conducted some computational experiments to test the efficiency of RRH in terms of material waste, total costs (final value of the objective function) and instances solved to optimality. We implemented the algorithms in Python, and used the solver CPLEX to solve the mathematical models, with the Application Programming Interface (API) *docplex*. The tests were run on a PC with an Intel i5 quad-core, with 24 GB RAM and Linux operating systems.

4.1 Computational results of RRH for the 1D-CSP model

In this section, we analyze RRH's behavior for the 1D-CSP model. For that purpose, we use IRUP (Integer Round-Up Property) and MIRUP (Modified Integer Round-Up Property) optimal criteria (Scheithauer and Terno 1995): let $\mathbf{x}$ be an optimal solution for R-1D-CSP, $f(\mathbf{x})$ the value of the objective function, and $\mathbf{y}$ a feasible integer solution; thus, $\mathbf{y}$ is considered an optimal solution according to IRUP criterion if $f(\mathbf{y}) = \lceil f(\mathbf{x}) \rceil$ and $\mathbf{y}$ is considered to be an optimal solution according to MIRUP criterion if $f(\mathbf{y}) \leq \lceil f(\mathbf{x}) \rceil + 1$. In most cases, the optimal value equals the IRUP criteria, but in some instances, an integer solution fulfills the MIRUP criteria (Scheithauer and Terno 1995; Martinovic and Scheithauer 2016). The results of RRH in terms of MIRUP optimal criteria are shown in this paper since it can be used to measure if the heuristic solution was not higher than the optimal value of the corresponding linear relaxation rounded up plus one.

We used the random generator CUTGEN1 (Gau and Wäscher 1995) to generate instances for the 1D-CSP with the same parameters used by Foerster and Wäscher (2000): object length $L = 1000$; number of type of pieces $m = 10$, 20 or 40; the length of the pieces ℓ_i were treated as random variables, assuming values relative to the size of L in the interval $[v_1 L, v_2 L]$ where, for small pieces size (S), $v_1 = 0.01$ and $v_2 = 0,2$, for wide-spread pieces size (W), $v_1 = 0.01$ and $v_2 = 0.8$, and large pieces size (L) $v_1 = 0.2$ and $v_2 = 0.8$; the average demand $\overline{d}$ was set to 10 or 100; the seed for all classes was set to 1994. We grouped the tests into 18 classes, each with 30 instances, permuting the values of m, ℓ_i (for S, W, and L), and $\overline{d}$, as shown in Table 4.

Table 5 shows the optima reached by the heuristic in absolute and relative terms according to the IRUP and MIRUP criteria. Only in four classes (highlighted bold in the table), the optimum was not achieved by the IRUP criterion, three of which missed one optimum of 30 instances and the other, two optima of 30 instances. For the other classes, the heuristic reached 100% the optimum according to IRUP criteria. Thus, from the 1080 instances tested in total (30 instances ×18 classes ×2 optimal criteria), the optimum has been missed in 5 instances, i.e., 0.4%. Finally, RRH reached 100% the optimum according to MIRUP criteria for all instances. The computational time average was 27 s among the classes.

We implemented three well-known heuristics, described in Poldi and Arenales (2009), to compare their results with RRH: First Fit Decreasing (FFD), Greedy, and Nova 1 (proposed in Poldi and Arenales (2009)). Figure 2 illustrates the percentage of optimal results obtained according to IRUP by RRH and its contenders, applied for the 18 classes. In the first six classes, RRH performed better than FFD and similar to the other two heuristics, as shown in Fig. 2a. From Fig. 2b, one may observe that RRH performed better or similarly than the others heuristics in all but one class, of number 12, where FFD obtained better results. A similar result occurs for the classes of Fig. 2c, where RRH performed worse than FFD and Greedy only in class 18 and better or equivalently for the remaining classes. We infer that RRH performed better than FFD in five of the 18 classes (27,8%) and worse in two (11,1%). Comparing RRH with Greedy, in four of the 18 classes (22,2%), RRH performed better and in one (5,5%) worse. Compared with Nova 1, RRH performed better in eleven of the 18 classes (61%). Taking the total instances, RRH achieved 99.07% of the optimal solutions according to IRUP criterion while Greedy achieved 96.29%; Nova 1, 84.63%, and FFD reached 97.40%.

4.2 Computational results of RRH for the 1D-MCSP model

The CUTGEN1 generator and the IRUP and MIRUP criteria cannot be used for the 1D-MCSP, as they are limited for the cases where there is a unique object length, making it difficult to compare RRH with existing approaches in the literature. Thus, we used four different strategies to analyze the RRH performance:

S1 We compare the RRH results with the *exact approach* results. We emphasize that the analysis's idea is that the optimal solution is an ideal benchmark to be achieved by the RRH. The results and discussion are presented in Sect. 4.2.1;

S2 In this analysis, we used the same parameters and classes as proposed in Poldi and Araujo (2016). We shall verify the waste of material, the objective function, and computational time obtained by RRH. These results are shown in Sect. 4.2.2;

S3 We compare the RRH waste of material, feasible solution, and computational time with the heuristic proposed in Poldi and Araujo (2016). The comparisons are presented in Sect. 4.2.3;

S4 Finally, we present the RRH performance using the same parameters and classes from Cherri et al. (2013) and Ravelo et al. (2020). Results are shown in Sect. 4.2.4.

All data used in the computational tests of the four strategies are available at github.com/BSCCampello/RRH.

Table 4 Parameters used to define the classes for the 1D-CSP

Class	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
m	10	10	20	20	40	40	10	10	20	20	40	40	10	10	20	20	40	40
ℓ_i	S	S	S	S	S	S	W	W	W	W	W	W	L	L	L	L	L	L
$\overline{d}$	10	100	10	100	10	100	10	100	10	100	10	100	10	100	10	100	10	100

Table 5 Optimal results achieved by RRH according to IRUP and MIRUP criteria for the 1D-CSP

Class	IRUP		MIRUP		Average time (s)
	Optimum achievements	Percentage of optimum achievements	Optimum achievements	Percentage of optimum achievements	
1	30	100	30	100	16
2	30	100	30	100	20
3	30	100	30	100	30
4	30	100	30	100	31
5	30	100	30	100	48
6	30	100	30	100	54
7	30	100	30	100	15
8	30	100	30	100	15
9	30	100	30	100	27
10	**29**	**96.7**	30	100	28
11	30	100	30	100	48
12	**29**	**96.7**	30	100	48
13	30	100	30	100	8
14	30	100	30	100	8
15	**29**	**96.7**	30	100	15
16	30	100	30	100	14
17	30	100	30	100	29
18	**28**	**93.3**	30	100	33
Average	29.72	99.07	30	100	27

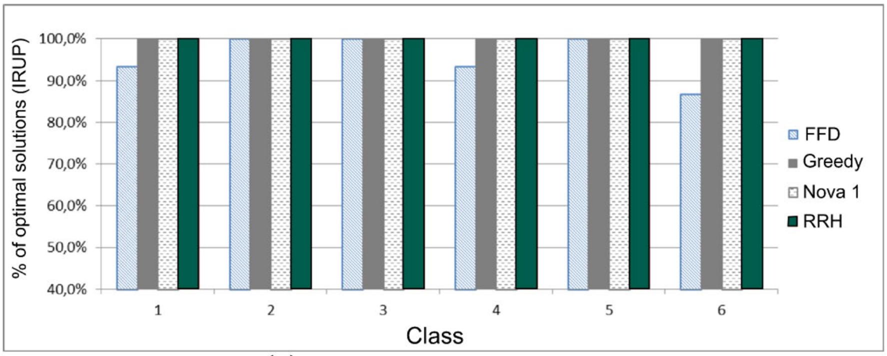
4.2.1 Comparison with the optimal solution

The results of the exact approach (EA) presented in this section are obtained by solving the 1D-MCSP with integer variables. For each instance, we generate all the possible cutting patterns and solve the model by the solver CPLEX. For each example, we set a timeout of 30 min; if the optimal solution was not achieved until this time, the solver returns the best feasible solution found so far.

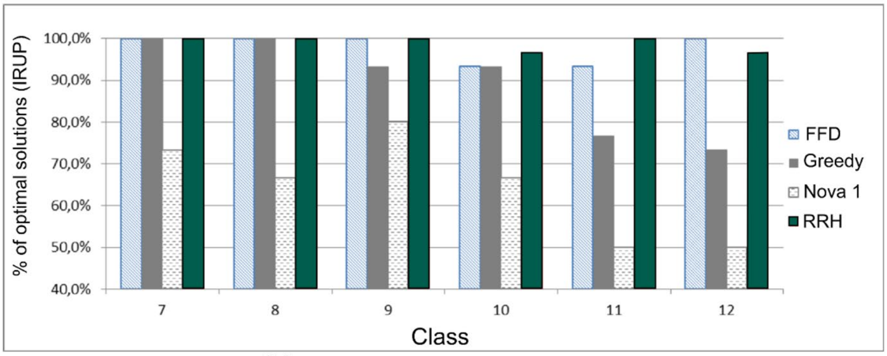
We perform the tests into classes, each with 20 simulations, combining the number of types of pieces m, the number of periods τ, and the number of objects' types $\varkappa$, according to Table 6.

The values highlighted bold in Table 7 are the classes in which the RRH achieved better value than EA. Table 7 reports the averages results (rounded to the nearest integer) of the 20 simulations for each class. We also show the gap between RRH and EA, calculated as **Gap**(%) $= \frac{RRH-EA}{EA} \times 100$. For all classes except Class 6, the gap in the objective function was at most 12.6%. The gap in the waste of material and the value of the objective function for Class 8 were negative, showing for larger simulations, it becomes difficult to test the performance of the heuristics against the optimal solution.

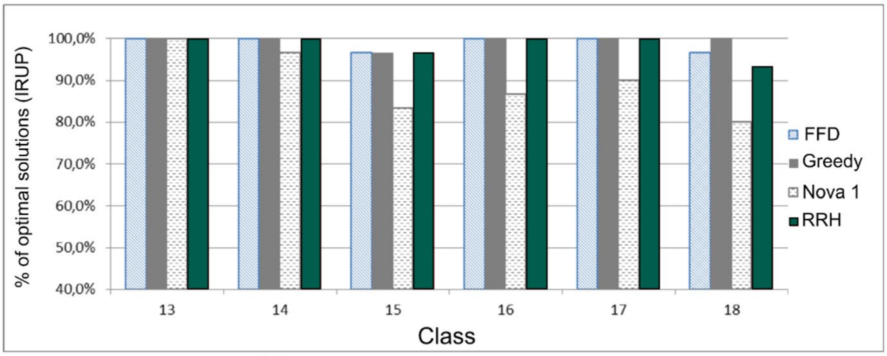
Table 8 presents some information concerning the exact (EA) and column generation approaches. The column labeled "os" expresses the number of optimal



(a) Heuristic results for Classes 1-6.



(b) Heuristic results for Classes 7-12.



(c) Heuristic results for Classes 13-18.

Fig. 2 Comparison of the percentage of optimum achieved according to IRUP criterion by FFD, Greedy, Nova 1, and RRH heuristics

solutions achieved by EA. The last three columns show the size of the samples: the average number of constraints ("nc") of the 1D-MCSP mathematical model; the number of variables, in average, for the EA approach ("nv (EA)"); the total number of variables, in average, employed by the column generation approach ("nv (CG)")

Table 6 Parameters used for the comparison with the exact approach (EA)

Class	1	2	3	4	5	6	7	8
m	3	3	3	3	8	8	8	8
τ	2	2	6	6	2	2	6	6
$\varkappa$	2	3	4	5	2	3	4	5

Table 7 Comparison of the performance obtained by RRH and the exact approach (EA)

Class	Waste of material (cm)			Objective function			Computational time (s)		
	RRH	EA	Gap (%)	RRH	EA	Gap(%)	RRH	EA	Difference (%)
1	4041	3901	3.5	4106	3968	3.5	2	91	− 97.8
2	5122	4912	4.3	5188	4978	4.2	5	90	− 94.4
3	6525	6308	3.3	7016	6799	3.2	29	491	− 94.1
4	4301	4115	4.3	4699	4513	4.1	27	92	− 70.6
5	2505	2232	12.2	2646	2372	11.5	9	482	− 98.1
6	748	535	39.9	938	724	29.5	14	285	− 95.0
7	2366	1955	17.4	3560	3162	12.6	81	740	− 89.0
8	696	972	**− 39.6**	1852	2127	**− 12.9**	131	461	− 71.6
Average	3288	3116	5.7	3751	3580	7.0	37	341	− 88.8

(after all the columns included). EA was not capable of achieving the full quantity of 20 optimal solutions in any class. The worst performance was Class 7, where EA met only 12 optimal solutions of 20 simulations. It is important to remark that, even though these samples are significantly small compared to those used in practice, EA took, on average, 821% more time to obtain results that are not much better than those obtained by RRH. For real-world samples, the need for employing efficient and parsimonious heuristics, as RRH, becomes evident.

4.2.2 Instances from Poldi and Araujo (2016)

The instances used in this section are generated with the same parameters as presented in Poldi and Araujo (2016):

Storage costs of objects: $h_{kt} = \beta L_k$, where $\beta \in [0{,}01; 0{,}1]$;
Storage costs of pieces: $\sigma_{it} = \alpha \ell_i$, where $\alpha \in [0{,}01; 0{,}1]$;
Objects length: $L_k \in [300; 1000]$;
Pieces length: $\ell_i \in [0{,}1; 0{,}4] \times \frac{\sum_{k\in K} L_k}{K}$;
Number of available objects: $o_{kt} \in [\lceil 2av_t \rceil; \lceil 3av_t \rceil]$, where $av_t = \frac{\sum_{i\in I} \ell_i d_{it}}{\sum_{k\in K} L_k}$;
Demand for pieces: $d_{it} \in [10; 200]$,

and the same classes, as shown in Table 9.

Table 10 presents the computational results. The column “Waste of material” show the average waste of material (in centimeters) and the corresponding percentage of waste of material concerning the total length of the object. One may note that the percentage loss reduces as the size of the classes increases. The average computational time was at most 398.6 s. Considering the size of the instances and the degree of difficulty in solving the model, performance of the RRH, both in terms of waste of material and computational time, can be considered as satisfactory.

The last two columns of Table 10 show the average value of the objective function (OF) and the percentage of storage costs over the total amount of the objective function. It is worth noting that the higher the class, the higher the percentage of storage cost over the OF and the lower the waste of material. By increasing m, τ, and $\varkappa$, better cutting patterns are achieved, minimizing the material loss, while increasing the storage cost, since the pieces are cut in advance.

4.2.3 Comparison with the heuristic proposed in Poldi and Araujo (2016)

The 1D-MCSP mathematical model presented in this study is very similar to that used in Poldi and Araujo (2016), called *Generalized Gilmore and Gomory’s* (GGG) model. The difference is that we add Constraint (4), which forces the initial and final stored pieces to be null, being useful in industries that do not desire to stock pieces beyond the known demand.

Poldi and Araujo (2016) proposed a heuristic, denoted here as Poldi and Araujo Heuristic (PAH), to solve the GGG Mathematical Model based on a rolling horizon strategy (Poldi and Araujo 2016). An integer solution for the first period is found, while it remains fractional for future periods. After the first period has started being implemented in practice, it is possible to update future periods’ demand. The heuristic is applied again to solve the new first period; meanwhile, the future periods’ solution remains fractional. Therefore, in Poldi and Araujo (2016), only the first period is indeed implemented, and it is the period whose solution will be rounded.

To measure the heuristic performance, the authors in Poldi and Araujo (2016) proposed two approaches denoted as *rounded first period* + *linear relaxation* and

Table 8 Optimal solutions (os) achieved by EA and size of the samples

Class	os	nc	nv (CG)	nv (EA)
1	19	13	34.5	169.2
2	19	17	63.1	344.7
3	15	39	146.1	716.4
4	19	51	230.8	1274.7
5	15	28	94.9	3954.2
6	17	32	159.5	9696.0
7	12	74	367.2	14854.2
8	16	86	553.2	30648.9
Average	16.5	42.5	206.1	7707.3

Table 9 Parameters used in the real-world experiment

Class	1	2	3	4	5	6	7	8
m	10	10	10	10	20	20	20	20
τ	3	3	6	6	3	3	6	6
$\varkappa$	3	5	3	5	3	5	3	5

rounded first period + *rounded other periods*. In both approaches, the first period is rounded using the PAH. The pieces production of the first period is subtracted from the first period and future periods demand. Thus, the next periods' demand (period 2 up to τ) is summed and called the *super period*. Then, in the first approach (rounded first period + linear relaxation), only the first period is rounded, and the super period is solved by linear programming. The waste of material of the rounded first period plus the waste obtained by linear relaxation is considered the total waste of material. In the second approach (rounded first period + rounded other periods), an integer solution to the super period is determined using the procedure proposed in Poldi and Arenales (2009).

To make a fair comparison between the heuristics, we adopt the same strategy for the RRH. First, we solve the R-1D-MCSP with the column generation procedure, but without Constraint (4), as done in Poldi and Araujo (2016). Then we apply the RRH for the first period, and the super period remains fractional (rounded first period + linear relaxation). For the second approach (rounded first period + rounded other periods), we apply RRH for the first period and again the RRH for the super period. At the end of the super period, we compute the waste of the cutting patterns. If there are some non-maximal cutting patterns, we complete them by applying the Greedy heuristic in the waste of material, allowing the production of pieces beyond the known demand. The input data used are the same as used in Poldi and Araujo (2016).

Table 10 RRH performance for the 1D-MCSP with significantly larger instances

Class	Waste of material (cm)		Computational time (s)	Objective function	Storage costs over the OF (%)
	(cm)	(%)			
1	763	0.12	29.2	936	18.5
2	277	0.06	36.8	433	36.7
3	941	0.08	116.2	1489	36.8
4	472	0.04	171.9	980	51.8
5	350	0.03	83.2	614	43.1
6	256	0.02	97.8	526	51.3
7	420	0.02	351.6	1472	71.4
8	309	0.01	398.6	1217	74.6
Average	473.5	0.05	160.7	958.6	47.9

The values highlighted bold in Tables 11 and 12 are those where the better value was achieved comparing both heuristics. Table 11 shows the RRH results regarding waste of material, feasible solutions, and computational time for the first approach. The results of PAH were taken from Table 12 of Poldi and Araujo (2016); they do not provide the computational time for their first approach. The first remark of Table 11 is that, in Class 5, RRH achieved 19 feasible solutions over 20 instances. In one sample, the number of available objects was not enough to cover the demand using the RRH approach. Then, to give an idea of the average waste of material of Class 5, we add a unit of an object in the sample, and we obtained an average for Class 5, as shown in the table.

From Table 11, it is possible to observe that RRH presents a more reduced waste of material than PAH in six of the eight classes (classes 1, 3, 4, and 6–8). The column **Gap (%)** shows the difference between the heuristics waste of material. The best result of RRH was in Class 7, being 52.70% better than PAH, while the worst was in Class 2, being 14.28% worse than PAH. Finally, the class with the highest computational time was Class 6 (39.03 s), while the lowest was Class 1 (6.21 s).

Table 12 presents the RRH and PAH results (taken from Table 13 of Poldi and Araujo (2016)) for the second approach. A instance in Class 5 was not successfully solved, so we added an object to find a feasible solution. Table 12 shows that for seven of the eight classes, the RRH presented a more reduced waste of material (for Class 5 it was not possible the comparison). The RRH better performance was in Class 4, with a 53.99% gain compared to PAH. Regarding the computational time, PAH needed less time to find an integer solution for classes 1–5 and 7, while the RRH needed less time for Classes 6 and 8.

4.2.4 Instances from Cherri et al. (2013) and Ravelo et al. (2020)

In these tests, we aim to solve harder instances using the same parameters and classes used in Cherri et al. (2013) and Ravelo et al. (2020). For this purpose, we consider objects of two lengths with unlimited availability in the stock in all the periods: $\varkappa = 2$, $L_1 = 1000$, $L_2 = 1100$; The number of types of pieces is $m = 50$, with the length of the pieces in the interval $\ell_i \in [10.5, 262.5]$; and the demands in the interval $d_i \in [200, 500]$. We varied the number of periods from 2 to 12: $\tau = 2, \dots, 12$. The storage cost of objects is considered null, since the stock availability is unlimited, and we tested tree different values for the storage cost of the pieces: $\alpha = 0$, $\alpha = 0.5$ and $\alpha = 1$. For each period and each α, we solve 20 instances and calculate the average results considering the **waste of material** $(\%) = \frac{\text{total waste}}{\text{total object (cm) used}}$; **OF for period** $= \frac{\text{OF}}{\text{number of periods}}$; and **computational time** (in seconds).

Figure 3 summarizes the computational results for this set of instances. Figure 3a shows that the higher the piece's storage cost, the higher the material loss. For instance, when $\tau = 2$ and $\alpha = 0$, the waste of material is, on average, 0.536×10^{-4},

Table 11 Rounded first period + linear relaxation solution for the super period

Class	Waste of material (cm)			Feasible solution		Time (s)
	PAH	RRH	Gap (%)	PAH	RRH	RRH
1	155.99	**147.92**	5.17	20	20	6.21
2	**120.71**	137.95	− 14.28	20	20	17.15
3	181.28	**107.90**	40.47	20	20	7.13
4	176.22	**103.05**	41.52	20	20	15.96
5	294.42	183.91*	–	19	20	11.10
6	178.50	**120.40**	32.54	20	20	39.03
7	381.01	**180.18**	52.70	20	20	15.37
8	235.24	**144.60**	38.53	20	20	33.02

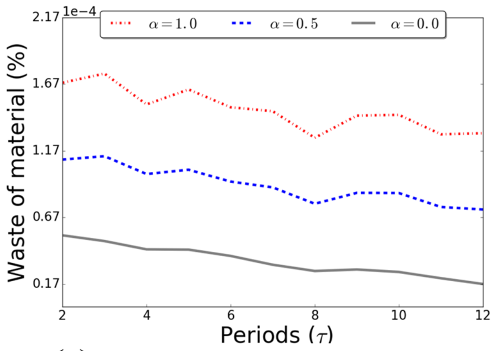
and when $\tau = 2$ and $\alpha = 1$, the waste of material is 1.68×10^{-4}, which means a 213% increase. Figure 3a also shows that the higher the number of periods, the smaller the waste of material. A similar analysis can be made from Fig. 3b. Another remark in Fig. 3a, b is that the number of periods influences a more reduced waste of material and OF value per period. For instance, when $\tau = 2$ and $\alpha = 0$, the waste of material is, on average, 0.536×10^{-4}, and when $\tau = 12$ and $\alpha = 0$, it is 0.170×10^{-4}, i.e., a 215% increase. This analysis suggests that it is possible to minimize the waste of material and the OF value by reducing the storage costs or solving the problem considering more periods.

Finally, Fig. 3c provides the RRH computational time, on average. For $\alpha = 0$, we observe a nearly linear growth of the time; for $\alpha = 1$, this growth is closer to an exponential. For $\alpha = 0$ and $\tau = 2$, the computational time was 21.45 s, and for $\tau = 12$, it was 309.78 s. For $\alpha = 1$ and $\tau = 2$, the RRH obtained 64.09 s, and for $\tau = 12$, it was 1420.21 s (about 23.66 min). Given the complexity of solving the 1D-MCSP using the exact method with harder instances, we consider that the RRH obtained an acceptable performance.

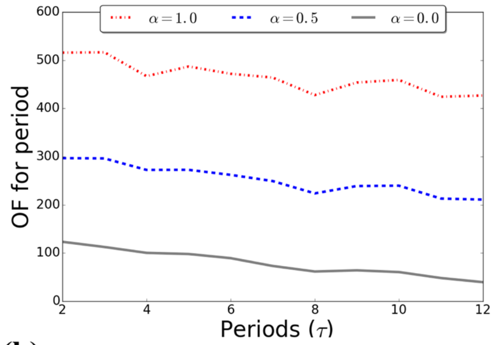
Table 12 Rounded first period + rounded super period

Class	Waste of material (cm)			Feasible solution		Time (s)	
	PAH	RRH	Gap (%)	PAH	RRH	PAH	RRH
1	411.35	**205.50**	50.04	20	20	**1.83**	4.51
2	289.25	**215.25**	25.58	20	20	**13.80**	14.99
3	335.50	**185.50**	44.64	20	20	**3.44**	6.30
4	345.10	**158.75**	**53.99**	20	20	21.14	**15.00**
5	482.90	272.20*	43.63	19	20	**4.48**	8.16
6	375.30	**193.85**	48.34	20	20	38.67	**32.12**
7	573.60	**283.35**	50.60	20	20	**11.46**	28.35
8	425.80	**216.85**	49.07	20	20	93.55	**28.72**

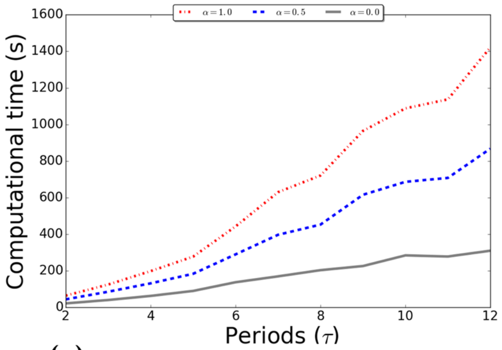
Fig. 3 Average values for waste of material, objective function for period, and computational time



(a) Average values for waste of material.



(b) Average values for objective function for period.



(c) Average values for computational time.

5 Conclusions

This work introduces the residual recombination heuristic (RRH) to solve the 1D-CSP and 1D-MCSP models. From the solution of the column generation procedure, RRH generates the matrix of residual cutting patterns and recombines them to obtain a feasible complementary solution. Computational results showed that RRH, when applied to solve the 1D-CSP model, achieves, on average, 99.07% of the optimum according to the IRUP criterion and 100% according to the MIRUP. We also compared RRH performance with other relevant contenders in the literature, and RRH performed satisfactorily.

Concerning the 1D-MCSP model, RRH achieved values of the objective function 7.0% above the exact solution. For instances using the same parameters as used in Poldi and Araujo (2016), the waste of material achieved by RRH was, on average, at most 0.12% over the total material used. By comparing the RRH with the heuristic proposed in Poldi and Araujo (2016), we observed that the RRH obtained a more reduced waste of material for the samples solved. Finally, we tested the RRH over the harder instances used in Cherri et al. (2013) and Ravelo et al. (2020) and the results showed an acceptable performance in terms of computational time.

We can conclude that the RRH obtained high-quality solutions within a reasonable computational time for both 1D-CSP and 1D-MCSP models. For future study, it may be interesting to adopt new methods of rounding the matrix of residual cutting patterns, which could lead to better cutting patterns.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Abdel-Basset M, Manogaran G, Abdel-Fatah L, Mirjalili S (2018) An improved nature inspired metaheuristic algorithm for 1-D bin packing problems. Pers Ubiquitous Comput 22(5–6):1117–1132

Andrews GE, Eriksson K (2004) Integer partitions. Cambridge University Press, Cambridge

Araujo SA, Constantino AA, Poldi KC (2011) An evolutionary algorithm for the one-dimensional cutting stock problem. Int Trans Oper Res 18(1):115–127

Ayres AOC, Campello BSC, Oliveira WA, Ghidini CTLS (2019) A bi-integrated model for coupling lot-sizing and cutting-stock problems. arXiv preprint arXiv:191202242

Belov G, Scheithauer G (2006) A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. Eur J Oper Res 171(1):85–106

Campello BSC, Ghidini CTLS, Ayres AOC, Oliveira WA (2020) A multiobjective integrated model for lot sizing and cutting stock problems. J Oper Res Soc 71(9):1466–1478

Carvalho J (1999) Exact solution of bin-packing problems using column generation and branch-and-bound. Ann Oper Res 86:629–659

Chen Y, Huang H, Cai H, Chen P (2019) A genetic algorithm approach for the multiple length cutting stock problem. In: 2019 IEEE 1st global conference on life sciences and technologies (LifeTech). IEEE, pp 162–165

Cherri AC, Arenales MN, Yanasse HH (2013) The usable leftover one-dimensional cutting stock problem—a priority-in-use heuristic. Int Trans Oper Res 20(2):189–199

Cherri AC, Arenales MN, Yanasse HH, Poldi KC, Vianna ACG (2014) The one-dimensional cutting stock problem with usable leftovers—a survey. Eur J Oper Res 236(2):395–402

Degraeve Z, Peeters M (2003) Optimal integer solutions to industrial cutting-stock problems: part 2, benchmark results. INFORMS J Comput 15(1):58–81

Degraeve Z, Schrage L (1999) Optimal integer solutions to industrial cutting stock problems. INFORMS J Comput 11(4):406–419

Delorme M, Iori M (2020) Enhanced pseudo-polynomial formulations for bin packing and cutting stock problems. INFORMS J Comput 32(1):101–119

Delorme M, Iori M, Martello S (2016) Bin packing and cutting stock problems: mathematical models and exact algorithms. Eur J Oper Res 255(1):1–20

Diegel A, Chetty M, Van Schalkwyck S, Naidoo S (1993) Setup combining in the trim loss problem-3-to-2 & 2-to-1. Business Administration, Durban

Dyckhoff H (1990) A typology of cutting and packing problems. Eur J Oper Res 44(2):145–159

Foerster H, Wäscher G (2000) Pattern reduction in one-dimensional cutting stock problems. Int J Prod Res 38(7):1657–1676

Gau T, Wäscher G (1995) CUTGEN1: a problem generator for the standard one-dimensional cutting stock problem. Eur J Oper Res 84(3):572–579

Ghidini C, Alem D, Arenales M (2007) Solving a combined cutting stock and lot-sizing problem in small furniture industries. In: Proceedings of the 6th international conference on operational research for development (VI-ICORD)

Gilmore P, Gomory RE (1961) A linear programming approach to the cutting-stock problem. Oper Res 9(6):849–859

Gilmore P, Gomory RE (1963) A linear programming approach to the cutting stock problem part II. Oper Res 11(6):863–888

Hinxman AI (1980) The trim-loss and assortment problems: a survey. Eur J Oper Res 5(1):8–18

Kantorovich L (1960) Mathematical methods of organizing and planning production. Manag Sci 6(4):366–422

Kim B, Wy J (2010) Last two fit augmentation to the well-known construction heuristics for one-dimensional bin-packing problem: an empirical study. Int J Adv Manuf Technol 50(9–12):1145–1152

Ma N, Liu Y, Zhou Z (2019) Two heuristics for the capacitated multi-period cutting stock problem with pattern setup cost. Comput Oper Res 109:218–229

Martinovic J, Scheithauer G (2016) Integer rounding and modified integer rounding for the skiving stock problem. Discrete Optim 21:118–130

Melega GM, de Araujo SA, Jans R (2018) Classification and literature review of integrated lot-sizing and cutting stock problems. Eur J Oper Res 271(1):1–19

Pitombeira AR, Athayde BP (2020) A matheuristic algorithm for the one-dimensional cutting stock and scheduling problem with heterogeneous orders. TOP 28:178–192

Poldi K, Araujo SA (2016) Mathematical models and a heuristic method for the multiperiod one-dimensional cutting stock problem. Ann Oper Res 238(1–2):497–520

Poldi KC, Arenales MN (2009) Heuristics for the one-dimensional cutting stock problem with limited multiple stock lengths. Comput Oper Res 36(6):2074–2081

Poltroniere SC, Araujo SA, Poldi KC (2016) Optimization of an integrated lot sizing and cutting stock problem in the paper industry. TEMA (São Carlos) 17(3):305–320

Ravelo SV, Meneses CN, Santos MO (2020) Meta-heuristics for the one-dimensional cutting stock problem with usable leftover. J Heuristics 26:586–618

Scheithauer G, Terno J (1995) The modified integer round-up property of the one-dimensional cutting stock problem. Eur J Oper Res 84(3):562–571

Stadtler H (1990) A one-dimensional cutting stock problem in the aluminium industry and its solution. Eur. J. Oper. Res. 44(2):209–223

Vanderbeck F (1999) Computational study of a column generation algorithm for bin packing and cutting stock problems. Math. Program. 86(3):565–594

Wäscher G, Gau T (1996) Heuristics for the integer one-dimensional cutting stock problem: a computational study. Oper. Res. Spektrum 18(3):131–144

Wäscher G, Haußner H, Schumann H (2007) An improved typology of cutting and packing problems. Eur. J. Oper. Res. 183(3):1109–1130