São José dos Campos - SP
LV SBPO
LV Simpósio Brasileiro de Pesquisa Operacional
São José dos Campos, SP - 6 a 9 de novembro de 2023

A hybrid approach via GRASP meta-heuristic with reinforcement learning for the guillotine cutting stock problem

Alan Douglas C. Rebouças, Jonathan L. Silva, Thiago Henrique F. Oliveira, Dario José Aloise
Universidade do Estado do Rio Grande do Norte
Rua Prof. Antônio Campos, Mossoró - RN, 59610-090
{alandolglasicapui, nathan13joker, thiagohfo}@gmail.com,
darioaloise@uern.br

Hugo Alexandre D. do Nascimento, Humberto J. Longo
Universidade Federal de Goiás
Av. Esperança, Goiânia - GO, 74690-900
{hadn, longo}@ufg.br

ABSTRACT

The guillotine cutting stock problem (GCSP) refers to the problem of finding the best layout for cutting sheets of 2D raw material using a guillotine cutting procedure in order to satisfy a specific demand of rectangular items. This problem belongs to the $\mathcal{NP}$-hard class of combinatorial optimisation problems. Therefore, it is not possible to guarantee optimal solutions in a viable computational time to large instances using exact methods. The present work proposes a hybrid meta-heuristic approach to the GCSP: a Greedy Randomised Adaptive Search Procedure (GRASP) combined with a Q-learning algorithm, denoted here as GRASP-learning. The Q-learning algorithm, in the hybrid GRASP, replaces the constructive phase of solutions. It aims to improve the quality of the initial solutions for the meta-heuristic. Computational tests performed with known benchmark instances demonstrated that the hybrid methods are competitive and promising when compared to other results reported in the specialised literature.

KEYWORDS. Combinatorial optimisation. Guillotine cutting. Sequential decision problem. Hybrid meta-heuristic. Q-learning.

Paper topics: MH - Meta-Heuristics, CO - Combinatorial Optimisation

DOI: 10.59254/sbpo-2023-175114
http://live-proceedings.pantheonsite.io/p/175114

1. Introduction

The cutting stock problem is the well known problem of cutting standard-sized pieces of stock material, such as paper rolls or metal sheets, into pieces of specified sizes while minimising material wasted. In the present article, the stock material is assumed to be identical rectangular sheets of raw material, that are successively cut by guillotine, into rectangles of the required sizes, with usable leftovers. There is no restriction on the quantity or on the order in which horizontal and vertical guillotine cuts can be applied. Furthermore, any rectangular item to be cut can be rotated 90 degrees if necessary. This special case is called the guillotine cutting stock problem (GCSP).

Many scientific works report on investigations into the GCSP, such as the studies carried out by [Lodi et al., 2002], [Macedo et al., 2010], [Fleszar, 2013] and [Furini et al., 2016]. However, despite being a widely explored research topic, this problem remains challenging, as increasingly larger data instances, that arise from practical industry scenarios and in benchmark databases, have been proposed. The development of new methods based on combining existing techniques with different characteristics can be advantageous for solving the GCSP, with great potential to be more efficient in dealing with instances of challenging dimensions. One possibility in this direction is to combine machine learning techniques with meta-heuristics, which has already been done for many other combinatorial optimisation problems [de Lima Junior et al., 2007], [Lopes Silva et al., 2019], [Talbi, 2021], [Bengio et al., 2021], [Li et al., 2021] but is still incipient for guillotine cutting problems. Therefore, there is an opportunity to expand research in this area.

The current work presents a hybrid approach that aims to find high quality solutions for GCSP instances in reasonable computational time. The proposed method combines a meta-heuristic (*GRASP*) with a *Reinforcement Learning* (*RL*) technique, more specifically a Q*-learning* algorithm customised to work on a representation of the GCSP as a sequential decision problem. In the proposed approach, the Q-learning algorithm generates initial solutions that, in general, are of good quality. These solutions are then passed on to the meta-heuristic, that further improves them. The guillotine cutting results achieved with this hybrid approach proved to be superior to those obtained by the aforementioned algorithms alone.

The remainder of the paper is organised as follows. Section 2 formally describes the GCSP. In Section 3, all elements of the proposed hybrid approach are presented in detail. This includes: a constructive heuristic for producing a guillotine cutting solution, called the Adapted HHD-Heuristic; a customised Q-learning method that uses the constructive heuristic; and the new hybrid method, called GRASP-learning. An evaluation of the hybrid methods is presented in Section 4. Our final considerations and conclusions are drawn in Section 5.

2. The GCSP

Consider a list $\mathcal{S}$ of large rectangular sheets of raw material with standard size $W \times H$ (width $\times$ height). Let $\mathcal{D} = \langle d_i, i = 1, \ldots n \rangle$ be a list of n individual rectangular items to be cut, with item d_i having size $w_i \times h_i$ ($w_i \leqslant W$ and $h_i \leqslant H$). For simplicity, from now on we assume the label of the item $d_i \in \mathcal{D}$ to be i, for $i = 1, \ldots n$. We refer to the items using their labels in most tables and figures, when this information is unambiguous. Other scientific works in the field usually group items of same size into a single item demand and specify their quantity. However, the given definition of $\mathcal{D}$ is necessary to support the modelling of the cutting problem as a sequential problem. Furthermore, assume $|\mathcal{S}| \geqslant n$, in order to satisfy the total cutting demand.

A *cut layout* is a specific arrangement of how to cut items on a single larger sheet. The *Guillotine Cutting Stock Problem* (GCSP) aims to find cut layouts that optimally use sheets in $\mathcal{S}$ for producing all items in $\mathcal{D}$. As previously mentioned, we assume that there is no restriction on the

DOI: 10.59254/sbpo-2023-175114
http://live-proceedings.pantheonsite.io/p/175114

order in which the guillotine cuts (either horizontal or vertical) can be applied, implying in a "stage-unrestricted", and more general, version of the problem. Another relevant aspect is the possibility of rotating any rectangular item in 90 degrees before cutting it on the sheet, what makes the problem even more complex due to the larger number of cutting combinations. We call *wasted piece* every rectangular piece from a sheet that remained non-used after cutting all demanded items. During the cutting process, if a rectangular piece can still be used for further producing a demanded item, then it is named *remaining area* or *remaining piece*.

Usually, the objective of the GCSP is to minimise the waste of raw materials, what can be interpreted in many different ways. For instance, we could minimise the total amount of used sheets, or the sum of the area of the wasted pieces. The objective function adopted in this work, to be minimised, seeks to incorporate both goals and is formally defined as:

$$f(\mathcal{S}, \mathcal{D}) = \|\mathcal{L}\| + \omega, \tag{1}$$

where $\mathcal{L}$ is the list of constructed cut layouts ($\|\mathcal{L}\| \leqslant \|\mathcal{S}\|$). The term ω is equal to $1 - a/(W \times H)$, with a the area of the largest wasted piece in sheet with highest total waste, after the cutting process.

Let $\sigma(\mathcal{D}) = \langle d_{k_1}, d_{k_2}, \ldots, d_{k_n} \rangle$, with $d_{k_i} \in \mathcal{D}$ for $1 \leqslant k_i \leqslant n$, be a particular permutation of the demanded items. Also, assume the existence of a procedure that, given a permutation $\sigma(\mathcal{D})$, deterministically builds a set of stage-unrestricted guillotine cut layouts that fulfils the complete item demands and provides its objective function value. The adapted HHD-Heuristic described in Section 3.1 acts as such a procedure.

2.1. The GCSP as a sequential decision problem

Using the list $\mathcal{D}$ with n demanded rectangular items, presented in the previous section, and the concepts introduced in [Puterman, 2014] for defining a sequential decision problem, we formulate the GCSP as a quintuple $M = \langle \mathcal{T}, \mathcal{E}, \mathcal{A}, R, \Pi \rangle$, where:

- $\mathcal{T} = \{t_0, t_1, \ldots, t_{n-1}\}$ is the set of decision times;
- $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$ is the set of decision states, with each state $e_i \in \mathcal{E}$ corresponding to a unique item $d_i \in \mathcal{D}$ to be embedded in cut layouts, $i = 1, \ldots n$;
- $\mathcal{A}$ is the set of all actions a_{ij}, with $i, j = 1, \ldots n$ and $i \neq j$, that specify that an item d_j must be cut after the item d_i. The set is denoted by:

$$\begin{aligned}\mathcal{A} &= \{a_{12}, a_{13}, \ldots, a_{1n}\} \cup \{a_{21}, a_{23}, a_{24}, \ldots, a_{2n}\} \cup \cdots \cup \{a_{n1}, a_{n2}, \ldots, a_{nn-1}\} \\ &= \{a_{12}, a_{13}, \ldots, a_{1n}, a_{21}, a_{23}, a_{24}, \ldots, a_{2n}, \ldots, a_{n2}, a_{n3}, \ldots, a_{nn-1}\}.\end{aligned}$$

- $R : \mathcal{E} \times A \to \mathbb{R}^+$ is a function that computes the expected return (expected gain) for performing an action in a given state. For the GCSP, the expected return depends only on the ratio of the area of the second item involved in the action and the area of the raw material sheet. It is expressed, for an action a_{ij}, as:

$$r_j = \frac{w_j \cdot h_j}{W \cdot H}. \tag{2}$$

  The expected return value can be used for choosing the item d_j that fits in the available area and occupies the largest possible area of the sheet.
- $\Pi : \mathcal{E} \times \mathcal{A} \times \mathcal{E} \to [0, 1]$ is a function that defines transition probabilities between states $e_i, e_j \in \mathcal{E}$, where $\pi_{ij}(e_j \mid e_i, a_{ij})$ corresponds to the probability of reaching the state e_j from state e_i, by choosing the action a_{ij}. In the modelling adopted in the current work, it is assumed that such choices are deterministic and, therefore, $\pi_{ij}(e_j \mid e_i, a_{ij}) = 1; i, j = 1, \ldots, n$.

DOI: 10.59254/sbpo-2023-175114
http://live-proceedings.pantheonsite.io/p/175114

During the sequential construction of a cut layout, some actions may no longer be available. This occurs when an item entirely fills the available area of the current raw material sheet or when the sheet does not have the areal capacity to cut any of the remaining items. In the latter case, corresponding states are available, but actions related to those items are no longer allowed. So, when there are no more permitted actions, but there are remaining uncut items, a new sheet is added and the remaining demands become available states. Decisions regarding item rotation and whether the initial cut is horizontal or vertical are not considered in this modelling. These particulars are treated by the adapted HHD-Heuristic, described in Section 3.1.

Figure 1 illustrates the above model for a GCSP instance, consisting of a set $\mathcal{D}$ with the first five rectangular items given in Table 1 and standard raw material sheets with dimension $W \times H = 4.2 \times 4.2$. The decision process is modelled as a graph $G = (V(G), E(G)) = (\mathcal{E}, \mathcal{A})$, with set $\mathcal{A}$ containing the possibilities of consecutive cuts of two items. For example, a transition from state e_1 to state e_2 (action a_{12}) means cutting item d_2 in the raw material sheet immediately after this has been done for the item d_1. At decision time t_0, the state e_1, representing the demand d_1, is chosen. At t_1, e_2 is chosen, and so on, as highlighted in red. At t_2, the actions referring to e_3 are discarded, because d_3 does not fit in the sheet after the previous actions. Therefore, e_4 is chosen as the next state to be processed. At the next decision time, e_5 is chosen. Finally, a new sheet must be added to the decision process and one more iteration is needed to fit the item d_3 in it.

Tabela 1: Size of the seven rectangular items for cut.

Item (i)	1	2	3	4	5	6	7
Width (w_i)	2.0	4.2	3.2	1.6	1.2	0.8	1.2
Length (h_i)	2.5	1.8	1.3	1.4	0.7	1.5	2.8
Area ($w_i \cdot h_i$)	5.00	7.56	4.16	2.24	0.84	1.20	3.36

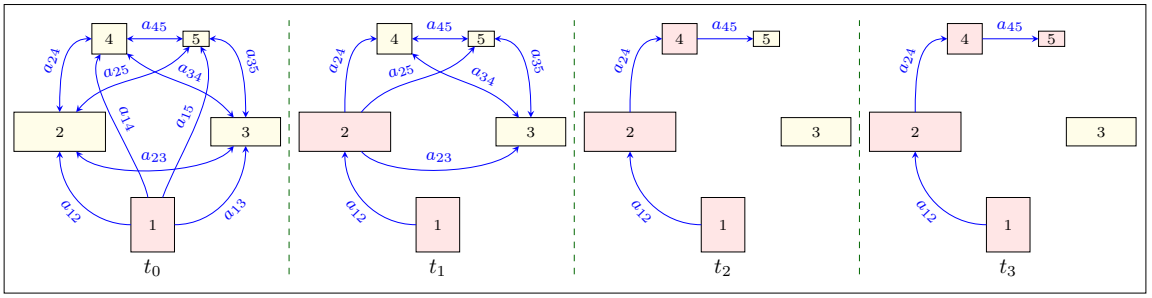

Figura 1: The decision graph of a GCSP instance.

3. The hybrid approach

Figure 2 shows the internal elements of the hybrid approach. As previously mentioned, it consists of a classical meta-heuristic combined with a customised Q-learning method. In addition, instead of adopting a new (and possibly complex) data structure for describing a solution to an instance of the guillotine cutting problem, our methods use the simple demanded-item permutation $\sigma(\mathcal{D})$ for representing any solution. Another algorithm, called here *adapted HHD-Heuristic*, is embedded in the approach with the aim of generating the associated cutting layouts for any given $\sigma(\mathcal{D})$, whenever needed. These elements work as follows:

DOI: 10.59254/sbpo-2023-175114
http://live-proceedings.pantheonsite.io/p/175114

1. The customised Q-learning algorithm is employed repeatedly for creating initial solutions (permutations $\sigma(\mathcal{D})$). While generating a solution (what involves composing a permutation item by item), the Q-learning calls the adapted HHD-Heuristic in order to produce its associated cutting layouts and, based on their quality, to enhance an internal decision process.
2. The "Meta-heuristic core" in Figure 2 is usually the part of a classical meta-heuristic responsible for improving initial solutions. Many meta-heuristics can be used here, and their core can be detached and inserted in the approach for computing higher quality results based on the set of initial solutions. For every new permutation $\sigma(\mathcal{D})$ produced by the core, the adapted HHD-Heuristic is called for generating the associated guillotined cutting layouts.

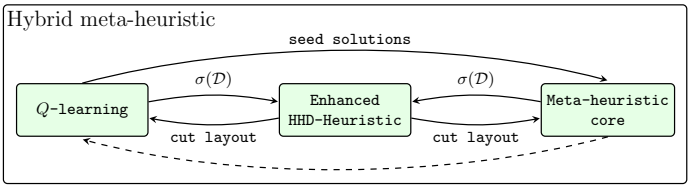

Figura 2: Architecture of the hybrid approach.

Overall, our hybrid approach explores the "space of permutations", with each permutation $\sigma(\mathcal{D})$ representing a possibly different solution. For every such solution, the adapted HHD-Heuristic is capable of producing its associated cutting layouts. The cost of a solution is based on its cutting layouts, as defined in Eq. (1). This strategy keeps the core meta-heuristic independent of the particulars of the optimisation problem. All the complex tasks related to constructing a guillotined cutting solution, evaluating it, and ensuring that the constraints of the problem are not violated, are performed by the adapted HHD-Heuristic. Also, there is no need for complex data structures to represent the solutions. In the following subsections, the adapted HHD-Heuristic and the customised Q-learning method for the proposed approach are presented.

3.1. The Adapted HHD-Heuristic

The *adapted HHD-Heuristic* is based on the constructive heuristic HHD-Heuristic, proposed by [Nascimento et al., 1999]. This former technique was designed to generate good quality feasible solutions for instances of the 2D guillotine cutting problem. Given a list $\mathcal{D}$ of rectangular items and a list $\mathcal{S}$ of raw-material sheets for cutting, the HHD-Heuristic works as follows:

I. A particular permutation $\sigma(\mathcal{D})$ is defined as a copy of $\mathcal{D}$, but having all items sorted in non-increasing order of area. A permutation $\sigma(\mathcal{S})$ is also defined as a list of rectangular pieces sorted in non-decreasing order of area containing, initially, all sheets in $\mathcal{S}$ (with standard size) but that, during the cutting processing, can also hold remaining pieces of sheets (for later reused). Let $\mathcal{C}$ be a list of guillotine cut actions, with each *cut action* represented by a quadruple $\langle d, s, rot, orient \rangle$, where d is an item from $\sigma(\mathcal{D})$, s is a sheet or a piece of a sheet taken from $\sigma(\mathcal{S})$, and $rot \in \{True, False\}$ is a flag indicating if d should be rotated or not in order to be cut from s. Finally, $orient \in \{none, h, v, hv, vh\}$ defines the cut orientation to be applied: $none$ means that no cut is necessary since d fits precisely in s without loss; h (or v) means that just a horizontal (or a vertical) guillotined cut is needed to extract d from s; and hv (or vh) means that a horizontal cut followed by a vertical cut (or the opposite) are necessary in order to obtain d. Let also $\mathcal{Z}$ be a list of wasted pieces. Initially, $\mathcal{C} = \langle\,\rangle$ and $\mathcal{Z} = \langle\,\rangle$, that is, these lists are empty.

II. If $\sigma(\mathcal{D}) = \langle\,\rangle$ or $\sigma(\mathcal{S}) = \langle\,\rangle$, stop and return $\mathcal{C}$ and $\mathcal{Z}$.

DOI: 10.59254/sbpo-2023-175114
http://live-proceedings.pantheonsite.io/p/175114

III. Remove the smallest sheet s in $\sigma(\mathcal{S})$ (always at the beginning of this list).
IV. Find the largest item d in $\sigma(\mathcal{D})$ that fits in s. If there is no such item, s is simply discarded as a wasted piece (added to $\mathcal{Z}$), and the algorithm goes to Step II. Otherwise, d is removed from $\sigma(\mathcal{D})$ and the algorithm continues below.
V. Cutting process: evaluate all possible scenarios for cutting d on the lower-left corner of s and choose one of them, according to a predefined strategy as further explained. Let $\langle d, s, rot, orient\rangle$ be the *cut action* associated with the chosen scenario. Add $\langle d, s, rot, orient\rangle$ to $\mathcal{C}$. Furthermore, if $orient \in \{h, v\}$, then a smaller section of s remains and it must be added to the beginning of $\sigma(\mathcal{S})$ (since it is smaller in area than the current first element in $\sigma(\mathcal{S})$). If $orient \in \{hv, vh\}$, then two smaller sections of s remain. They have to be sorted in non-decreasing order of area and added to the beginning of $\sigma(\mathcal{S})$. Note that $\sigma(\mathcal{S})$ always stays sorted in non-decreasing order according to the area of its elements.
VI. Go to Step II.

In every execution of Step V, the cut scenarios must consider the possibility of rotating or not the item d by 90 degrees and all valid cut orientations. In order to guide the decision towards the best cut scenario, three strategies were proposed by [Nascimento et al., 1999] such that one of them should be previously selected and applied during the run of the HHD-heuristic. They consist of choosing the scenario that maximises:

1) The area of the smallest remaining section of s (if not null);
2) The area of the largest remaining section of s (if not null); or
3) The smallest side-dimension (width or height) of any remaining section of s (if not null).

The algorithm iterates until all items are cut ($\sigma(\mathcal{D}) = \langle\,\rangle$) or when there is no piece available for satisfying the complete cut demand ($\sigma(\mathcal{S}) = \langle\,\rangle$). The idea behind the heuristic is to cut the smallest sheets (or the smallest remaining pieces of a sheet) first, what reflects a desire to reuse small remaining pieces and leave the larger ones for future demands. The decision to choose the largest item that fits on a piece is guided by the intuition that larger items are more important and should be cut and supplied first. Moreover, the above three strategies aim at providing remaining pieces of raw material that are still large enough to be reused in further cut actions.

The cut layouts generated by the HHD-Heuristic are dependent on the order in which the items and the raw material sheets (and pieces) are kept in their respective permutations, as defined in Step I of the algorithm previously presented. Thus, different permutations of item and/or raw material sheets (and pieces) may result in different cut layouts. Such a particular characteristic allows for the usage of the HHD-Heuristic as a generic generator of guillotine cut layouts for a given permutation of demanded items. This idea was previously explored in a related work by [Silva, 2017] and is extensively utilised here. It consists of choosing first an item and then looking for the best suitable piece of raw material to cut it. Thus, a given permutation $\sigma(\mathcal{D}) = \langle d_{k_1}, d_{k_2}, \ldots, d_{k_n}\rangle$ is processed sequentially from d_{k_1} to d_{k_n}, without changing the order of its elements. For each item d in $\sigma(\mathcal{D})$, the smallest piece/sheet s in $\sigma(\mathcal{S})$ that allows a feasible cut is chosen and used in the cutting process.

3.2. Customised Q-learning method

As explained before, the *Q-learning* is employed in the present work to allow a metaheuristic to start its processing from high quality initial solutions. The Q-learning, introduced in [Watkins, 1989], is a model-free method based on a discrete, finite-time Markov decision process and on mappings from states to actions, i.e., on policies or rules to decide what to do with given knowledge about a particular state. A policy should be defined over the entire state space in order to specify the action to be taken in any given situation. The concepts and notations adopted from now

DOI: 10.59254/sbpo-2023-175114
http://live-proceedings.pantheonsite.io/p/175114

on to describe the Q-learning method are the same used in Section 2.1, when modelling the GCSP as a sequential decision problem.

Following that formalisation, each demanded item d_i is represented by a state e_i to be visited by a learning agent, $i = 1, \ldots n$. An action characterises the choice of cutting an item from a sheet and implies the changing from the current state to the state associated with the particular item. The Q-values matrix always has size $n \times n$. Furthermore, the Q-learning iteratively updates the Q-values matrix for state-action pairs using the expression:

$$Q[e_i, a_{ij}] = (1 - \alpha) \cdot Q[e_i, a_{ij}] + \alpha \cdot \left(r_j + \gamma \cdot \left(\max_{\substack{1 \leqslant k \leqslant n \\ k \neq i,j}} Q[e_j, a_{jk}] \right) \right), \qquad (3)$$

where e_i is the current state, a_{ij} is the performed action from state e_i to state e_j, r_j is the received immediate reward for executing a_{ij}, a_{jk} for several values of k are the possible actions at the state e_j, γ is a discount factor $(0 \leqslant \gamma < 1)$ and α is a learning factor $(0 < \alpha < 1)$.

A customised version of the Q-learning method to deal with the GCSP was developed for the present work (its pseudocode is presented in Algorithm 1). It takes as input the list $\mathcal{D}$ and the size of the raw material sheets, as well as its own configuration parameters, and computes a Q-values matrix as other Q-learning methods do. The initial state in the Q-learning is always e_1 (representing the first item d_1). Furthermore, the adaptive policy ε-greedy [Lima Júnior, 2009] is used to select the next actions and to define the behaviour of the learning agent. This technique consists of choosing the highest value of Q with probability $1 - \varepsilon$ and a random action with probability ε.

Algorithm 1: Customised Q-learning $(\alpha, \varepsilon, \gamma, \mathcal{D}, W, H)$

Input: Learning factor α, probability ε, discount factor γ, the list of demanded items $\mathcal{D}$ and the size W and H of the raw material sheets.
Output: The computed Q-values matrix.

```
Initialise the Q-values matrix with zeros;
while (episodes are not over) do
    Initialise σ(S) = ⟨(W, H, W · H)⟩ for the adapted HHD-Heuristic, D' = D, and the list of
      not-yet cut items U = {};
    i ← 1;
    Use the adapted HHD-Heuristic to find a suitable way to cut σ = ⟨d_i⟩ from σ(S);
    Remove d_i from D';
    while (D' ∪ U is not empty) do
        Select a_ij in accordance with the ε-greedy rule, with d_j ∈ D';
        Use the adapted HHD-Heuristic to find a suitable way to cut σ = ⟨d_j⟩ from σ(S);
        Remove d_j from D';
        if (d_j could not be cut in σ(S)) then
            Add d_j to U;
            if (D' is empty) then
                σ(S) = ⟨(W, H, W · H)⟩;
                Move all items from U to D';
        else
            Compute the reward r_j for d_j;
            Update Q[e_i, a_ij] using Eq. (3), with e_i associated to d_i;
            i ← j;
return Q.
```

DOI: 10.59254/sbpo-2023-175114
http://live-proceedings.pantheonsite.io/p/175114

The customised Q-learning method works as follows. Firstly, all entries of the Q-values matrix are initialised to zero (Step 1). Next, for every episode (an iteration of Step 2), the permutation $\sigma(\mathcal{S})$ is initialised with just one sheet for the cutting process; a temporary list $\mathcal{D}'$ is defined containing the elements in $\mathcal{D}$; and an empty list of items processed but not yet cut is defined (Step 3). Then, the first item d_i, with $i = 1$, is selected (Step 4), and a suitable way of cutting it is evaluated (Step 5). A version of the adapted HHD-Heuristic is used for this later task, with the modification that it deals with only one demanded item at a time and that it preserves the current state of its data structures for supporting the next item evaluation. Overall, the HHD-Heuristic finds a suitable s in $\sigma(\mathcal{S})$ and the best way of cutting the item d_i in s, possibly resulting in the inclusion of smaller pieces of s in $\sigma(\mathcal{S})$.

By definition, d_1 can be cut from the initial piece in $\sigma(\mathcal{S})$ and therefore, it is removed from $\mathcal{D}'$ after been processed (Step 6). Next, while there are still items to be cut, the algorithm processes each of them sequentially (Step 7). An action a_{ij} with $d_j \in \mathcal{D}$ is selected according to a ε-greedy rule [1], thus inducing a transition from state e_i to e_j (Step 8). The selected action a_{ij} involves a new item d_j, whose possible cut is now evaluated (Step 9). Next, the item d_j is removed from $\mathcal{D}'$, in order to indicate that it has been processed (Step 10). If no cutting was possible using any of the pieces in $\sigma(\mathcal{S})$, as verified in Step 11, then d_j is added to the list of processed but not-yet cut items (Step 12). At this moment, if there is no other item to cut (Step 13), $\sigma(\mathcal{S})$ is initialised with a new sheet of standard size (Step 14) and all items in U are re-inserted into $\mathcal{D}'$ (Step 15). In contrast, if d_j could be cut from $\sigma(\mathcal{S})$, then the reward for d_j is computed, the Q-values matrix is updated and e_j becomes the new current state (Steps 17-19). The operations in the innermost while loop are repeated until a complete cut layout is formed, that is, when all items have been cut. This entire process is repeated several times (each time consisting of a new episode), always starting with the item d_1 and sequentially adjusting the Q-values matrix. At the end of the running cycles, the Q-learning method has found a policy (the Q-values matrix) for producing cutting layouts that includes all demanded items.

The Q-learning algorithm provides a balance between using the knowledge already obtained and looking for new spaces not yet explored. This is due to the ε-greedy rule, that occasionally allows visits to other states randomly chosen. Such a characteristic makes the Q-learning suitable to be integrated with other techniques, in order to create a more robust and effective way of exploring the search space of an optimisation problem. The next two sections expand this idea, by introducing a hybridisation of the GRASP method with the Q-learning algorithm, followed by the permutation extraction procedure. From now on, we use only the term *customised Q-learning* method to refer to the customised Q-learning algorithm coupled with the permutation extraction procedure.

3.3. GRASP-learning

Our GRASP-learning method is an enhancement of a GRASP presented in [Silva, 2017] for the GCSP. Both of the GRASP methods work on a sequential representation of the solutions and depend on the adapted HHD-Heuristic for constructing the associated cutting layouts. The difference is that the GRASP-learning method employs the customised Q-learning method in the place of a constructive phase, as a greedy-random algorithm for generating initial solutions of good quality. The Q-values matrix is initialised just once, when creating the first starting solution. Then, this solution is improved repeatedly by local search. When iterating to produce a new starting

[1] The ε-greedy rule is a stochastic policy that is based on a predefined value $0 \leq \varepsilon \leq 1$. In order to use the rule with regard to a row i of the Q-values matrix, a number z is randomly chosen in the interval $[0, 1]$. If $z < \varepsilon$, then the algorithm randomly chooses any column j of the row that represents a not-yet cut item (d_j); Otherwise, the column j with maximum value in the row, that represents a not-yet cut item (d_j), is chosen.

DOI: 10.59254/sbpo-2023-175114
http://live-proceedings.pantheonsite.io/p/175114

solution, the latest Q-values matrix is reused and updated. The fact that the Q-values matrix is never reset but just reused at each GRASP iteration allows it to function as an adaptive memory that saves past experiences to be used in the future. This process raises the chance of producing higher quality initial solutions as the GRASP algorithm progresses. Furthermore, the ε parameter of the Q-learning maintains the balance between "greediness" and "randomness". The number of episodes in the customised Q-learning is a parameter that can be empirically defined.

The 2-*Opt* descent technique is employed in the local search phase of both the GRASP and GRASP-learning methods. This technique allows to move to a neighbouring solution only if it improves the objective function value. The local search method receives an item permutation list (in the GRASP-learning method, generated by the customised Q-learning method), performs item position exchanges and sends, at each change, the new permutation to the HHD-Heuristic for assembling the cut layouts and evaluating the related solution. If the new solution has a lower objective value, it becomes the best solution found so far.

4. Computational evaluation

We now evaluate the effectiveness of the new hybrid methods in solving the GCSP. All computational tests were performed on a personal computer with a 64-bit Windows 10 operating system, 8 GB of RAM memory at a frequency of 1333 MHz and an Intel Core i7-7660U 2.5 GHz 2-core processor. For the experiment, we employed the numerical instances compiled by [Lodi et al., 1999] from previous works, that are widely used benchmark instances from the literature on two-dimensional stock cutting problems. The entire dataset has 500 instances divided into ten classes, each with five groups of 10 instances containing 20, 40, 60, 80 or 100 items.

The GRASP-learning method, as well as their classical version, were implemented in Python 3.8. Preliminary tests were performed with a subset of 20% of the instances, in order to find the best calibration for the parameters of the methods. Consequently, the classical GRASP method was set to run with $\alpha = 0.2$ and $N_{max} = 50$. When applicable, the same parameter values were used for both classical and hybrid (learning) versions of these two methods. Additionally, the following parameters of the customised Q-learning were employed in the hybrid meta-heuristic: $N_{\text{Ep}} = 200$ (number of episodes), $\alpha = 0.90$ (learning rate), $\gamma = 0.90$ (discount factor) and $\varepsilon = 0.90$ (balance between greediness and randomness). A decay rate $\tau = 0.97$ was applied to the epsilon parameter for each episode. All methods were run 50 times for each of the 500 instances.

Let $\|\mathcal{L}_{i,j}\|$ be the number of used standard sheets in the cutting solution produced by an method for the i-th instance, $i = 1, 2, \ldots 500$, at the j-th execution, $j = 1, 2, \ldots 50$. Also let $t_{i,j}$ be the time required by the method for producing that particular solution. We define the following five quality metrics for the method: Sum Best, Sum Worst, Average Sum, Average Time and Average σ. Sum Best, Sum Worst and Sum Mean define the sum of the best, worst and average, respectively, of the results obtained in the 50 executions. The Average σ define the mean standard deviation ($\sqrt{\frac{\sum_{i=1}^{500}\sigma_i^2}{500}}$) and the Average Time defines the mean elapsed time at all executions.

We note that the above quality metrics are defined on the standard sheets list, while internally the methods presented in this paper employ the complete objective function described in Eq. (1). This was done for maintaining compatibility with the quality metric already adopted in the literature for this dataset, which consists of the sum of the used sheets in all instances.

In order to enable an additional evaluation, the methods were compared with 10 other methods from the literature, including the standard HHD-Heuristic. Such comparisons were made in terms of the "Sum Best" metric.

DOI: 10.59254/sbpo-2023-175114
http://live-proceedings.pantheonsite.io/p/175114

4.1. Results

Table 2 provides the results of the GRASP and the GRASP-Learning methods, as well as of the other methods for comparison. The GRASP-learning excelled in the worst and in the best cases, demonstrating that the application of the Q-learning was positive. This was obtained at a cost of a higher computational time (not included in the table), as the GRASP-Learning took 33.79 seconds in average to process all GCSP instances while the GRASP demanded only 16.89 seconds. The Average σ obtained for GRASP was 0.10 and for GRASP-Learning 0.06.

Table 2 also provides a comparison of the new hybrid method to other methods: TSKP – Tabu Search Knapsack Problem [Lodi, 1999], GLS – Guided Local Search [Faroe et al., 2003], GBL/A-B – Guillotine Bottom Left constructive heuristic/agent-based implementation [Polyakovsky e M'Hallah, 2009], CH – Constructive Heuristic [Charalambous e Fleszar, 2011], CHBP – Constructive Heuristic with Biased sufficiency limit and Postoptimization [Charalambous e Fleszar, 2011], CFIH – Critical-Fit Insertion Heuristic [Fleszar, 2013] and CFIH+J4 – CFIH with Justification with four iterations [Fleszar, 2013], are deterministic heuristics reported in the literature. The Adapted HDD-Heuristic, described in Section 3.1, is also included in the table. The eleventh line of the table refers to the Asynchronous Team (ATeam) constructed by [Silva, 2017], followed by the lower bound proposed by [Clautiaux et al., 2007].

The GRASP-learning method matched the total number of sheets in the best solution produced by the *Asynchronous Team* (*ATeam*) of [Silva, 2017], which is currently the best known result. Furthermore, the GRASP-learning method had a lower value of *Sum Worst* and of *Average* σ and is much simpler to describe and to implement than the ATeam method. The software architecture of the ATeam consists of six initialisation agents and five improvement agents, some of them being complete meta-heuristics such as GRASP, a Tabu Search, a Genetic Algorithm, a BRKGA, and also some combinations of those methods such as a hybrid GRASP-GA method [Silva, 2017]. Another point to note is that the GRASP-learning method could itself be embedded in the ATeam in order to further improve its output.

Tabela 2: Results obtained by the methods for the entire set of instances.

Method	**Type**			**Sum Best**	**Sum Worst**	**Average Sum**	**%LB Dist.**	**Year**
GRASP	H	M		7051	7112	7058.00	2.307	
GRASP-learning	H	M	RL	7045	7068	7057.65	2.220	–
TSKP	H	M		7101	–	–	3.032	1999
GLS	H			7367	–	–	6.982	2003
A-B	H	Hp		7063	–	–	2.481	2009
CH	H			7161	–	–	3.903	2011
CHBP	H	I		7064	–	–	2.495	2011
CFIH	H			7100	–	–	3.018	2013
CFIH+J4	H	I		7080	–	–	2.728	2013
Adapted HHD-Heuristic	H			7349	–	–	6.631	2017
ATeam	H	M	MA	7045	7117	7085.14	2.220	2017
Lower Bound				6892			0.000	2007

H: Heuristic Hp: Hyper-heuristic M: Meta-heuristic
I: Improvement RL: Reinforement Learning MA: Multi-agent

DOI: 10.59254/sbpo-2023-175114
http://live-proceedings.pantheonsite.io/p/175114

5. Conclusion

This work proposed a new approach for the Guillotine Cutting Stock Problem (GCSP) based on a hybridisation of a GRASP method with a reinforcement learning technique, in particular, a Q-learning algorithm. To validate the approach, the new hybrid method GRASP-learning as well as their non-hybrid version were implemented and tested on a well-known numerical benchmark. The analysis of the obtained results showed that the combination of meta-heuristics with the Q-learning algorithm is promising. Using this strategy, the hybrid meta-heuristic produced better results than its classical version. The GRASP-learning generated the best overall results. The new hybrid method also compare favourably with other algorithms found in the specialised literature.

The adoption of a simple sequential representation for the guillotine cut solutions combined with a heuristic for providing the associated cutting layouts allowed the non-conventional use of the Q-learning algorithm, which greatly enhanced the power of the proposed methods. This strategy can be generalised to tackle other non-trivial optimisation problems. Some aspects of the present research might be further developed through more detailed and in-depth studies. For instance, the quality performance of the GRASP-learning method could be evaluated for other GCSP datasets. Furthermore, the new hybrid method can be tailored to solve other optimisation problems, considering that the Adapted HHD-Heuristic is replaced by a suitable equivalent algorithm.

Referências

Bengio, Y., Lodi, A., e Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour dâhorizon. *European Journal of Operational Research*, 290(2):405–421.

Charalambous, C. e Fleszar, K. (2011). A constructive bin-oriented heuristic for the two-dimensional bin packing problem with guillotine cuts. *Computers & Operations Research*, 38 (10):1443–1451.

Clautiaux, F., Jouglet, A., e El Hayek, J. (2007). A new lower bound for the non-oriented two-dimensional bin-packing problem. *Operations Research Letters*, 35(3):365–373.

de Lima Junior, F. C., de Melo, J. D., e Neto, A. D. D. (2007). Using Q-learning Algorithm for Initialization of the GRASP Metaheuristic and Genetic Algorithm. In *2007 International Joint Conference on Neural Networks*, p. 1243–1248. IEEE.

Faroe, O., Pisinger, D., e Zachariasen, M. (2003). Guided local search for the three-dimensional bin-packing problem. *Informs journal on computing*, 15(3):267–283.

Fleszar, K. (2013). Three insertion heuristics and a justification improvement heuristic for two-dimensional bin packing with guillotine cuts. *Computers & Operations Research*, 40(1):463–474.

Furini, F., Malaguti, E., e Thomopulos, D. (2016). Modeling Two-Dimensional Guillotine Cutting Problems via Integer Programming. *INFORMS Journal on Computing*, 28(4):736–751.

Li, K., Zhang, T., Wang, R., Wang, Y., Han, Y., e Wang, L. (2021). Deep reinforcement learning for combinatorial optimization: Covering salesman problems. *IEEE transactions on cybernetics*, 52 (12):13142–13155.

DOI: 10.59254/sbpo-2023-175114
http://live-proceedings.pantheonsite.io/p/175114

Lima Júnior, F. C. (2009). *Algoritmo Q-learning como estratégia de exploração e/ou explotação para metaheurísticas GRASP e algoritmo genético*. PhD thesis, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Universidade Federal do Rio Grande do Norte – UFRN, Natal, RN, Brazil.

Lodi, A., Martello, S., e Vigo, D. (1999). Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems. *INFORMS Journal on Computing*, 11(4):345–357.

Lodi, A. (1999). *Algorithms for Two-Dimensional Bin Packing and Assignment Problems*. PhD thesis, Dottorato di Ricerca in Ingegneria dei Sistemi, Università degli Studi di Bologna, Bologna, Italy.

Lodi, A., Martello, S., e Monaci, M. (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252.

Lopes Silva, M. A., de Souza, S. R., Freitas Souza, M. J., e Bazzan, A. L. C. (2019). A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems. *Expert Systems with Applications*, 131:148–171.

Macedo, R., Alves, C., e De Carvalho, J. V. (2010). Arc-flow model for the two-dimensional guillotine cutting stock problem. *Computers & Operations Research*, 37(6):991–1001.

Nascimento, H. A. D., Longo, H. J., e Aloise, D. J. (1999). Uma Heurística $\mathcal{O}(mn)$ para o Corte Bidimensional Guilhotinado. In *Anais do XXXI Simpósio Brasileiro de Pesquisa Operacional, SBPO 1999*, p. 1197–1206, Juiz de Fora-MG, Brazil.

Polyakovsky, S. e M'Hallah, R. (2009). An agent-based approach to the two-dimensional guillotine bin packing problem. *European Journal of Operational Research*, 192(3):767–781.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Silva, J. L. d. (2017). Contribuições em otimização combinatória para o problema de corte bidimensional guilhotinado não-estagiado. Master's thesis, Mestrado em Ciência da Computação, Universidade Federal Rural do Semi-Árido, Mossoró-RN, Brazil.

Talbi, E.-G. (2021). Machine Learning into Metaheuristics: A Survey and Taxonomy. *ACM Computing Surveys (CSUR)*, 54(6):1–32.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK.

DOI: 10.59254/sbpo-2023-175114
http://live-proceedings.pantheonsite.io/p/175114