57º SBPO
SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
Gramado • RS
DE 5 A 9 DE OUTUBRO DE 2025

A Modular Random-Key Framework for the Two-Dimensional Variable Sized Bin Packing Problem

Felipe Silvestre Cardoso Roberto
Universidade Federal de São Paulo
São José dos Campos/SP
felipe.silvestre@unifesp.br

Lucas Mendes Santos Silva
Instituto Tecnológico de Aeronáutica
São José dos Campos/SP
mendeslmss@ita.br

Nilton Carlos Santos Araújo
Instituto Tecnológico de Aeronáutica
São José dos Campos/SP
nilton.araujo.101642@ga.ita.br

Rafael da Costa Azevedo
Instituto Tecnológico de Aeronáutica
São José dos Campos/SP
azevedorca@ita.br

Sanderson Lincohn Gonzaga de Oliveira
Universidade Federal de São Paulo
São José dos Campos/SP
sanderson.oliveira@unifesp.br

ABSTRACT

We propose a modular metaheuristic framework based on random-key encoding to solve both the one- and two-dimensional Variable-Sized Bin Packing Problem (VSBPP). The VSBPP generalizes the classical bin packing problem by allowing bins of different sizes and associated costs to minimize total packing cost. Our method separates the optimization engine from problem-specific constraints via dedicated decoders and incorporates the No-Fit Polygon and Bottom-Left placement rules to address geometric feasibility and item rotation cuts in the 2D case. We benchmark the approach against state-of-the-art algorithms using four standard datasets. The proposed framework achieved average cost gaps of 0.15% to 0.40% for 1D instances and of 9.4% to 19.4% for 2D instances—closely approaching the performance of exact methods while using only 600 seconds of runtime. These results confirm the framework's flexibility, computational efficiency, and applicability to complex real-world packing problems.

KEYWORDS. Bin Packing Problem, Random-Key Optimizers, Modular Metaheuristics

Paper topics: Metaheuristics, Cutting and Packing Problems, Combinatorial Optimization

DOI: 10.59254/sbpo-2025-212127
https://proceedings.science/p/212127?lang=pt-br

1. Introduction

Cutting and packing problems are classical combinatorial optimization challenges with widespread applications in industries such as aerospace, automotive, and manufacturing. One of the most studied variants is the Bin Packing Problem (BPP), where the objective is to pack items into a minimum number of fixed-size bins. The BPP is NP-hard [Garey and Johnson, 1979].

A generalization of the BPP is the Variable-Sized Bin Packing Problem (VSBPP), in which bins have different capacities and associated costs. The objective becomes minimizing the total computational cost rather than the number of bins used. In its two-dimensional version (2D-VSBPP), both items and bins are rectangles, with additional spatial constraints such as item rotation [Pisinger and Sigurd, 2005].

Traditional methods often couple optimization logic with problem-specific constraints, limiting adaptability [Lodi et al., 2002b; Pisinger and Sigurd, 2007; Bortfeldt, 2013; Gomes and Resende, 2015]. This paper introduces a modular framework based on the Random-Key Optimizer (RKO) architecture [Chaves et al., 2024], which decouples the metaheuristic engine from decoding logic, enabling reuse and flexibility. Our approach integrates geometric feasibility through No-Fit Polygon (NFP) [Albano and Sapuppo, 1980], Inner-Fit Polygon (IFP)[Bennell and Oliveira, 2008], and Bottom-Left (BL) placement [Baker et al., 1980], while leveraging biased random-key genetic algorithms [Gonçalves and Resende, 2011].

Exact methods such as branch-and-price [Pisinger and Sigurd, 2005] are effective for small instances but face scalability issues. Metaheuristics like GRASP and VNS [Hemmelmayr et al., 2012; Resende and Ribeiro, 2010] scale better but rarely handle complex constraints or real-world feasibility.

This work contributes:

- A modular RKO framework for 1D and 2D VSBPP;
- Integration of rotation and efficient placement in geometric decoders;
- A comparative evaluation with exact and heuristic methods on benchmark datasets;
- An open-source implementation in Python.

To the best of our knowledge, this is the first modular random-key-based approach applied to 2D-VSBPP with realistic constraints.

2. Background

2.1. Variable Sized Bin Packing Problems

The classical Bin Packing Problem (BPP) aims to pack a set of items into the minimum number of fixed-size bins, minimizing unused space. It is well-known to be NP-hard [Garey and Johnson, 1979]. A generalization of this problem is the Variable-Sized Bin Packing Problem (VS-BPP), where bins have heterogeneous capacities and associated costs. Instead of minimizing the number of bins, the goal is to reduce the total cost of the bins used [Coffman Jr et al., 1984; Kellerer et al., 2004; Epstein et al., 2006].

For the one-dimensional VSBPP (1D-VSBPP), we define items by their lengths, and bins by their capacities and costs. The task is to assign each item to a bin without exceeding its capacity and while minimizing the overall bin cost. In the two-dimensional case (2D-VSBPP), both items and bins are rectangular, introducing spatial layout constraints in addition to the cost minimization objective [Pisinger and Sigurd, 2005]. The packing of items must prevent any overlap, ensuring they remain entirely within the boundaries of the bin. These additional constraints significantly increase the complexity.

DOI: 10.59254/sbpo-2025-212127
https://proceedings.science/p/212127?lang=pt-br

Both versions of the VSBPP generalize the classical BPP and appear in practical applications such as cutting stock in manufacturing, packaging of irregularly shaped products, warehouse storage optimization, and logistics planning [Dyckhoff, 1990; Wäscher et al., 2007].

2.2. Random Key Optimizers

Random Key Optimizers (RKO) are continuous-domain search methods designed to tackle hard combinatorial optimization problems. The core idea is to encode a solution as a vector of real-valued keys in the interval $[0, 1)$, known as random keys. A problem-specific procedure, called a decoder, decodes these keys into a feasible discrete solution [Chaves et al., 2024]. This decoupling of the optimization engine from the problem structure allows the use of generic metaheuristics across multiple problem domains.

Various metaheuristics have been adapted to operate over random-key representations. In this work, we employ three widely used approaches:

- **The Biased Random-Key Genetic Algorithm (BRKGA) [Gonçalves and Resende, 2011]** is one of the most prominent RKO variants. It introduces a structured mating strategy that favors elite solutions and maintains diversity via biased parent selection.
- **Variable Neighborhood Search (VNS)** is a trajectory-based method that explores multiple neighborhoods with key-specific perturbations.
- **Iterated Local Search (ILS)** is a method that alternates local improvement and global perturbations.

2.3. Overlap Analysis

In two-dimensional cutting and packing problems, overlap detection is essential for constructing feasible solutions. The literature commonly employs two key techniques: the No-Fit Polygon (NFP), which checks for overlaps between pieces, and the Inner-Fit Polygon (IFP), which verifies that a piece fits entirely within the bin.

2.3.1. No-Fit Polygon (NFP)

The No-Fit Polygon (NFP), introduced by Art Jr [1966] and later formalized by Albano and Sapuppo [1980], is particularly effective for handling irregular (non-rectangular) items. Given two polygons A and B, the NFP, denoted as NFP(A, B), defines the region where placing A would result in an overlap with B, assuming the reference point of A lies within the NFP. This transformation reduces the overlap detection problem to a simple point-in-polygon test.

2.3.2. Inner-Fit Polygon (IFP)

While the NFP addresses inter-item overlaps, it does not guarantee that an item remains within the bin's boundaries. To ensure containment, the Inner-Fit Polygon (IFP), denoted IFP(A, S), is defined for item A against the container polygon S.

The Inner-Fit Polygon (IFP) extends the concept of the No-Fit Polygon (NFP) introduced by Art Jr [1966], offering a geometric method for determining feasible internal placements in packing and nesting problems. Bennell and Oliveira [2008] provided a key tutorial on NFPs and IFPs, while Burke et al. [2007] later introduced robust algorithms for NFP generation. These techniques are widely used in industries such as textiles, sheet metal, and shipbuilding to optimize material usage.

To ensure the feasible placement of a new item A within a container without overlapping previously placed items $\{B_i\}$, one must consider both the spatial constraints imposed by the container and the existing objects. This is effectively modeled by computing the geometric difference of two key geometric constructs: the *Inner Fit Polygon* IFP(A, S), which delineates all

DOI: 10.59254/sbpo-2025-212127
https://proceedings.science/p/212127?lang=pt-br

positions where A fits entirely within the container S, and the complement of the *No-Fit Polygons* $\text{NFP}(A, B_i)$, which represent all positions where A would overlap with an already placed item B_i. The set difference

$$\text{Feasible}(A) = \text{IFP}(A, S) \setminus \bigcup_i \text{NFP}(A, B_i) \tag{1}$$

thus defines the complete set of positions where item A may be placed such that it is both fully contained and does not overlap with any B_i. This geometric difference is a non-trivial geometric operation but has been rigorously studied and applied in industrial contexts. A detailed exposition of this technique and its geometric underpinnings is provided by Mahadevan and Daniels [1991], who demonstrate the practical use of IFP and NFP difference for irregular part placement.

2.4. Placement Rules

Once the feasible region for placing an item within a bin has been identified, many valid positions may remain. Evaluating all these positions is computationally infeasible, as the underlying problem is NP-hard. It motivates the use of heuristic placement rules to reduce the search space and computation time [Coffman et al., 1996]. The most classical rule is Bottom–Left (BL) [Baker et al., 1980], which selects the lowest and leftmost feasible position. This rule tends to fill the bin in an organized manner, aiming to maximize the occupied area of each placement.

3. Related Works

The Variable-Sized Bin Packing Problem (VSBPP) has been extensively studied across different domains. Existing approaches can be broadly categorized into exact methods, heuristics, and metaheuristics, each with varying degrees of scalability and adaptability.

3.1. One-Dimensional VSBPP

In the one-dimensional case, Epstein and Levin [2007] introduced a classification and approximation framework that has become a reference. Later, Belov and Scheithauer [2006] proposed a robust exact algorithm using the branch-and-bound technique.

Haouari and Serairi [2009] developed six heuristics based on set covering and subset sum, demonstrating strong performance, particularly in large-scale instances. Their set-covering-based heuristic offered near-optimal results at low cost. Hemmelmayr et al. [2012] extended this line by proposing a Variable Neighborhood Search (VNS) that incorporated dynamic programming and multiple neighborhood structures, yielding competitive results under nonlinear cost settings.

3.2. Two-Dimensional VSBPP

For the 2D-VSBPP, exact methods include the branch-and-price algorithm by Pisinger and Sigurd [2005], which applies Dantzig–Wolfe decomposition and column generation to solve moderate-size instances to optimality.

Heuristic and metaheuristic contributions include works by Lodi et al. [2002a] and Cung et al. [2000]. Other hybrid strategies have also been proposed by researchers such as Falkenauer [2005], Wei et al. [2016], and Zhang et al. [2014]. The latter addressed guillotine cuts with a combination of simulated annealing and binary search.

Despite this progress, many solutions lack modularity and adaptability to complex constraints such as rotation or realistic spatial feasibility. Classical approaches such as GRASP [Resende and Ribeiro, 2010] and traditional VNS require extensive tuning and are often tailored to specific configurations. Moreover, their reliance on synthetic datasets limits generalization.

Random-key genetic algorithms offer a promising alternative. The seminal work by Bean [1994] introduced random-key encodings, later refined by Gonçalves and Resende [2011] into the Biased Random-Key Genetic Algorithm (BRKGA). Gonçalves and Resende [2012] applied BRKGA to large-scale packing and layout problems. Chaves et al. [2024] proposed the Random-Key Optimizer (RKO), combining multiple metaheuristics under a modular architecture. While

DOI: 10.59254/sbpo-2025-212127
https://proceedings.science/p/212127?lang=pt-br

RKO has shown success in general combinatorial settings, its use in 2D-VSBPP remains unexplored.

To our knowledge, this is the first work to apply a modular random-key-based framework to the 2D-VSBPP, incorporating constraints such as item rotation and guillotine cuts. This novel approach aims to bridge the gap between algorithmic flexibility and practical feasibility.

4. Methodology

This section follows the main steps of our RKO approach. This study executed all code on an Intel i7-9700K CPU (3.6 GHz, 8 cores) with 16 GB of RAM (2400 MHz).

4.1. RKO

We implemented the Random-Key Optimizer (RKO) framework in Python, following the architecture proposed by Chaves et al. [2024]. This framework provides a modular and problem-independent design in which solutions are encoded as vectors of N real-valued keys in the interval $[0, 1)$. A problem-specific decoder interprets these vectors to produce feasible solutions in the discrete domain.

Our implementation integrates three metaheuristics: Biased Random-Key Genetic Algorithm (BRKGA), Variable Neighborhood Search (VNS), and Iterated Local Search (ILS). Each operates over the same continuous solution space but uses different search dynamics and perturbation strategies. The RKO framework entirely separates the optimization logic from the problem constraints, which the decoder encapsulates.

To enhance performance and promote diverse search behaviors, we adopt a parallel execution model. The three metaheuristics run concurrently, each using two CPU cores (totaling six cores). Although they operate independently — each with its search state, population (or trajectory), and parameter settings — all metaheuristics asynchronously contribute candidate solutions to a shared solution pool, as proposed in Chaves et al. [2024].

This shared solution pool serves as a lightweight communication mechanism that enables indirect cooperation between the metaheuristics. Promising solutions discovered by one method can influence the search trajectories of others through periodic updates or replacements. At the end of the execution, the best solution stored in the pool is returned as the final output.

This design combines exploration diversity with parallel efficiency, leveraging both algorithmic complementarity and multicore hardware. We describe the decoders used for the 1D and 2D versions of the Variable-Sized Bin Packing Problem (VSBPP) in the following subsections.

4.2. Decoder for 1D-VSBPP

In our implementation, the random-key vector has a length of $2N$, where N is the number of items to be packed. The vector is split into two equal parts: the first half defines the *insertion order* of the items, and the second half determines *which bin type to open* when needed.

To establish the insertion order, we sort the first N keys in ascending order. Each key is associated with its original index in the vector, which identifies the corresponding item. The sequence of original indices after sorting defines the packing order.

The second half of the vector — the remaining N keys — determines the selection of the type of the bin. For each item, we first attempt to pack it into one of the currently open bins. If multiple feasible bins are available, we apply a *best-fit* rule, selecting the bin that maximizes the utilization ratio:

$$\text{utilization} = \frac{\text{current load} + \text{item size}}{\text{bin capacity}}.$$

If no open bin can accommodate the item, a new bin is opened according to the corresponding key in the second half of the vector. We identify all bin types that can hold the item,

DOI: 10.59254/sbpo-2025-212127
https://proceedings.science/p/212127?lang=pt-br

multiply the key by the number of valid types, and take the integer part as the index of the bin type to open. This deterministic process ensures feasibility.

This decoding procedure transforms a continuous random-key vector into a valid packing solution for the one-dimensional VSBPP, allowing the optimizer to evaluate solution quality based on total bin cost.

4.3. Decoder for 2D-VSBPP

The 2D decoder follows the same high-level structure as in the 1D case, but incorporates geometric feasibility checks to handle spatial constraints. The random-key vector has a length of 2N, where the first half defines the *insertion order* and the second half controls bin type selection.

Insertion order is determined by sorting the first N keys, as in the 1D case. However, when placing each item I, the decoder must also evaluate geometric feasibility. For each open bin and each allowed rotation angle (0° and 90°), we compute the set of feasible positions:

$$\text{Feasible}(I) = \text{IFP}(I, S) \setminus \bigcup_i \text{NFP}(I, B_i),$$

where IFP ensures containment within the bin and NFP prevents overlaps with already placed items.

If feasible positions exist, we select the bin that maximizes space utilization:

$$\text{utilization} = \frac{\text{occupied area} + \text{item area}}{\text{bin area}},$$

and place the item at the bottom-leftmost feasible coordinate following the *Bottom-Left* rule.

If the item cannot be placed in any open bin, the corresponding key from the second half of the vector determines the bin type to open. We identify bin types that can hold the item in at least one allowed rotation, multiply the key by the number of valid types, and take the integer part to select the bin type and rotation. Placement then follows the same *Bottom-Left* strategy.

This decoding procedure generates geometrically feasible packings for the two-dimensional VSBPP, explicitly enforcing containment, non-overlap, and rotation constraints — making it a direct extension of the 1D logic to the spatial domain.

5. Experimental Methodology

5.1. Datasets

This research employed four datasets for experimental evaluation. The first three were used to test the 1D-VSBPP, while the fourth dataset was applied to the 2D-VSBPP. As there are few published instances available for the 2D variation, we adopted the dataset proposed by Pisinger and Sigurd [2005]. This paper describes the datasets below.

5.1.1. 1D-VSBPP

- **Set 1**: This dataset comprises 10 instances for each $n \in 25, 50, 100, 200, 500$, where n denotes the number of items. Instances follow a linear cost model, defined as $W_i = c_i, \forall i \in 1, \ldots, m$, with the number of bin types (m) fixed at 3 in all cases. Item weights are uniformly distributed in the interval $[1, 100]$. These instances are those presented in Monaci [2003]. It should be noted that optimal solutions for all problem cases are known and have been obtained through exact methods.

- **Set 2**: The second dataset, proposed by Correia et al. [2008], consists of 150 instances with a fixed number of bin types ($m = 7$). The bin capacities are defined as $W_1 = 70$, $W_2 = 100$, $W_3 = 130$, $W_4 = 160$, $W_5 = 190$, $W_6 = 220$, and $W_7 = 250$. Item weights are randomly drawn from the interval $[1, 250]$. As in Set 1, a linear-cost function is applied. The optimal solutions for these instances are not known.

DOI: 10.59254/sbpo-2025-212127
https://proceedings.science/p/212127?lang=pt-br

- **Set 3**: Also introduced in Correia et al. [2008], this dataset consists of 150 instances with $m = 7$ bin types and the same capacity values as in Set 2. Item weights are randomly selected from $[1, 250]$. These instances differ in that a concave cost function is defined as $c_i = \lfloor 10\sqrt{W_i} \rfloor$, for $i = 1, \ldots, 7$. Optimal solutions are not known for this set.

- **Set 4**: The fourth dataset, also from Correia et al. [2008], shares the same configuration as Sets 2 and 3 regarding the number and capacities of bin types. Item weights are randomly drawn from the interval $[1, 250]$. This set uses a convex cost function given by $c_i = \left\lfloor 0.1 \cdot W_i^{3/2} \right\rfloor$. As with Sets 2 and 3, optimal solutions are not available.

5.1.2. 2D-VSBPP

This dataset includes instances with five bin types each. The bin sizes are generated uniformly according to $[W/2, W] \times [H/2, H]$ where $W, H \in \{10, 30, 40, 100\}$, and the cost for each bin type c_k is defined as a function of its area $W_k H_k$, as specified by

$$c_k = \left\lceil 10 \cdot \frac{1.2 \cdot W_k H_k - 0.2 \cdot WH}{WH} \right\rceil.$$

For this work, we used the first four classes presented in Pisinger and Sigurd [2005], detailed in Figure 1.

Figure 1: Instances considered for Set 4

Class 1	W_k and H_k uniformly random in $[1, 10]$, $W = H = 10$
Class 2	W_k and H_k uniformly random in $[1, 10]$, $W = H = 30$
Class 3	W_k and H_k uniformly random in $[1, 35]$, $W = H = 40$
Class 4	W_k and H_k uniformly random in $[1, 35]$, $W = H = 100$

5.2. Baselines

For the one-dimensional variable-sized bin packing problem (1D-VSBPP), we adopt as a baseline the set covering (SC) method evaluated by Hemmelmayr et al. [2012] and proposed by Haouari and Serairi [2009]. This approach generates a pool of feasible packings using randomized heuristics and then solves a restricted set covering model to select the best combination. In Hemmelmayr's experiments, it showed strong performance, particularly on nonlinear cost instances. In particular, only the Variable Neighborhood Search outperformed the method. In our experiments, we compare the RKO-based heuristic directly with this SC method on the same benchmark instances.

For the two-dimensional variable-sized bin packing problem (2D-VSBPP), we use as a baseline the exact branch-and-price algorithm of Pisinger and Sigurd [2005]. This method applies Dantzig–Wolfe decomposition with delayed column generation, and its pricing subproblem is a specialized two-dimensional knapsack formulation. The authors also introduced the widely used 2D-VSBPP benchmark set. Their algorithm remains a state-of-the-art exact method, solving instances with up to 100 items to optimality or providing strong lower bounds. In our experiments, we compare the solutions of our heuristic with those reported by Pisinger and Sigurd on their benchmark set.

DOI: 10.59254/sbpo-2025-212127
https://proceedings.science/p/212127?lang=pt-br

5.3. Metrics

The present study evaluated the quality of the solutions using different metrics for the one- and two-dimensional variants of the problem.

For the 1D-VSBPP, we evaluated our method based on the *gap* to the best known solutions reported in the literature [Hemmelmayr et al., 2012]. Let C be the total cost obtained by our algorithm, and C^* be the best known cost for the instance; then the gap is defined as:

$$\text{Gap} = \frac{C - C^*}{C^*} \times 100\%.$$

For the 2D-VSBPP, we evaluate solution quality using the continuous lower bound L_{2c} proposed by Pisinger and Sigurd [2005]. This bound estimates the minimum packing cost by assuming items can be fractionally assigned to bins. It is computed as:

$$L_{2c} = \left\lceil \frac{c_{k_0}}{W_{k_0} H_{k_0}} \sum_{i=1}^{n} w_i h_i \right\rceil_c,$$

where $c_{k_0}/(W_{k_0} H_{k_0})$ is the lowest cost per unit area among all bin types. Despite being a relaxation, L_{2c} offers a tight and efficiently computable benchmark for evaluating heuristic solutions. This bound is derived from a Dantzig–Wolfe decomposition and solved via delayed column generation, providing tight estimates of the optimal cost. The gap is computed as:

$$\text{Gap}_{L_{2c}} = \frac{C - L_{2c}}{L_{2c}} \times 100\%.$$

These metrics allow for the evaluation of our metaheuristic's absolute quality and efficiency against state-of-the-art methods under different computational budgets.

6. Experiments

This section presents a comparative analysis of the RKO framework on both 1D and 2D VSBPP.

6.1. 1D-VSBPP

To assess the performance of our implementation in the one-dimensional case, we evaluated the Random-Key Optimizer (RKO) on three standard benchmark datasets (Sets 1, 2, and 3) and one adapted two-dimensional set (Set 4), treated as 1D instances. The present study compared performance against the Set Covering (SC) approach proposed by Haouari and Serairi [2009], which has been widely adopted as a strong baseline, including in the comparative study by Hemmelmayr et al. [2012].

Table 1 summarizes the average and standard deviation of the cost gap obtained by both methods across the four datasets. Results show that our RKO implementation, developed entirely in Python, achieved highly competitive performance, especially since both algorithms were evaluated under an identical time constraint of 150 seconds per instance.

Table 1: Average gap (%) and standard deviation for RKO and SC on 1D-VSBPP datasets

Dataset	RKO (avg)	RKO (std)	SC (avg)	SC (std)
Set 1	0.23	0.22	0.12	0.25
Set 2	0.22	0.50	0.17	0.50
Set 3	0.40	0.12	0.16	0.10
Set 4	0.15	0.14	0.10	0.12

DOI: 10.59254/sbpo-2025-212127
https://proceedings.science/p/212127?lang=pt-br

Despite its general-purpose structure and modular design, RKO yielded near-optimal results across all datasets. In particular, the gap differences compared to SC were modest, demonstrating the effectiveness of the random-key approach even in highly constrained scenarios. Notably, for Sets 2 and 4, RKO matched or approached the SC baseline, even when facing nonlinear cost functions and a greater variety of bin configurations.

These results confirm that our approach is not only flexible but also efficient, delivering performance comparable with state-of-the-art heuristics from the literature. Furthermore, the fact that RKO relies on a unified and reusable metaheuristic engine without problem-specific Hardcoding reinforces the strength of its design as a viable alternative for scalable and extensible VSBPP solvers.

6.2. 2D-VSBPP

To evaluate the performance of our method in the two-dimensional variant of the problem, we conducted a comparative analysis with the exact branch-and-price algorithm proposed by Pisinger and Sigurd [2005]. Their approach is the most established exact method for the 2D-VSBPP, employing Dantzig–Wolfe decomposition, delayed column generation, and integer programming to obtain tight lower bounds. However, this level of precision comes at the cost of significantly higher runtime, with their evaluations limited to 3600 seconds per instance.

Table 2 summarizes the average gap (%) and standard deviation achieved by our Random-Key Optimizer (RKO) with a 600-second time limit, compared to the results reported for the branch-and-price method in Pisinger and Sigurd [2005].

Table 2: Average and standard deviation of gaps (%) for RKO (600s) vs. Branch-and-Price (3600s)

Dataset	RKO (avg)	RKO (std)	Exact (avg)	Exact (std)
Class 1	9.40	3.00	7.87	2.37
Class 2	15.76	6.11	10.57	2.93
Class 3	18.53	3.93	14.14	1.70
Class 4	19.38	3.92	15.49	3.54

Despite being a general-purpose metaheuristic implemented in Python, RKO achieved highly competitive results, with average gaps consistently within 5–6 percentage points of the exact approach, and in some cases even closer. These results were obtained in only one-sixth of the computation time used by the exact method.

The branch-and-price algorithm leverages a robust mathematical programming framework and is designed to solve instances to proven optimality. However, the overhead associated with its setup and solution time makes it less practical for large-scale or time-constrained applications.

In contrast, RKO provides a flexible and efficient alternative. Its modular metaheuristic design, paired with a robust decoding strategy, allowed it to handle complex constraints such as rotation and guillotine cuts while maintaining solution quality. The relatively low standard deviations observed across instances also indicate that the method is stable and reliable across different problem classes.

These findings confirm that RKO can serve as a viable heuristic alternative to exact methods, particularly when solution time is a critical constraint. Its ability to consistently deliver near-optimal results under limited runtime suggests strong potential for real-world applications in packing, cutting, and layout optimization scenarios.

DOI: 10.59254/sbpo-2025-212127
https://proceedings.science/p/212127?lang=pt-br

7. Conclusion

This work presented and evaluated a modular Random-Key Optimizer (RKO) framework tailored for both the one-dimensional (1D) and two-dimensional (2D) variants of the Variable-Sized Bin Packing Problem (VSBPP). The VSBPP is a significant generalization of the classical Bin Packing Problem, shifting the objective from minimizing the number of fixed-size bins to reducing the total cost of bins with varying capacities and associated costs.

A key motivation for this research was addressing the structural rigidity of many existing solution methods for the VSBPP, where optimization logic is strongly coupled with problem-specific constraints. This coupling hinders the adaptability and extensibility of these methods to new variants or the inclusion of realistic constraints such as item rotation or guillotine cuts, often requiring substantial algorithmic rewrites. The RKO framework directly tackles this limitation. It achieves this by decoupling the optimization engine from problem-specific constraints through a random-key encoding and a dedicated decoder, which enhances the approach's adaptability and extensibility.

The proposed framework was validated through comprehensive experimental studies on standard benchmark datasets for both 1D and 2D VSBPP instances.

In the one-dimensional setting, our RKO implementation, using a decoder that manages item insertion order and bin type selection based on random keys and a best-fit heuristic, demonstrated highly competitive performance. Compared to the Set Covering (SC) method, a robust baseline from the literature, RKO achieved average cost gaps ranging from 0.15% to 0.40% across various datasets, closely approaching the SC's performance of 0.10% to 0.17%. The results confirmed the RKO's efficiency and robustness, even when handling complex scenarios with nonlinear cost functions and diverse bin types.

For the two-dimensional VSBPP, the RKO framework employed a decoder that incorporates essential geometric computations, such as the No-Fit Polygon (NFP) for overlap detection and the Inner-Fit Polygon (IFP) for containment verification. It supports realistic constraints such as item rotation (0°and 90°) and utilizes the Bottom-Left placement rule to select feasible positions within the bin. Benchmarked against an established exact branch-and-price algorithm, the RKO, despite being given significantly less computation time (600s vs. 3600s), delivered competitive results. Average cost gaps for RKO were between 9.4% and 19.4%, compared to the exact solver's 7.9% to 15.5%. The results highlight RKO's capability to handle the complex spatial and geometric constraints of the 2D problem while providing high-quality solutions in a practical timeframe.

The RKO framework presented in this work combines performance, generality, and extensibility. Its ability to deliver near-optimal results within limited computational budgets positions it as a viable and efficient alternative to traditional heuristics and exact methods for VSBPP research and real-world applications in packing, cutting, and layout optimization. Furthermore, the provision of an open-source implementation in Python supports reproducibility and encourages future extensions of the framework to other complex packing variants and constraints.

References

Albano, A. and Sapuppo, G. (1980). Optimal allocation of two-dimensional irregular shapes using heuristic search methods. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-10(5): 242–248.

Art Jr, R. C. (1966). *An approach to the two dimensional irregular cutting stock problem.* PhD thesis, Massachusetts Institute of Technology.

Baker, B. S., Coffman, E. G., Jr., and Rivest, R. L. (1980). Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4):846–855.

DOI: 10.59254/sbpo-2025-212127
https://proceedings.science/p/212127?lang=pt-br

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160.

Belov, G. and Scheithauer, G. (2006). An exact algorithm for orthogonal packing problems using a branch-and-bound approach. *European Journal of Operational Research*, 172(2):505–523.

Bennell, J. A. and Oliveira, J. F. (2008). The geometry of nesting problems: A tutorial. *European Journal of Operational Research*, 184(2):397–415.

Bortfeldt, A. (2013). A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints. *Computers Operations Research*, 39(11):2248–2257.

Burke, E. K., Hyde, M. R., Kendall, G., O'Brien, F., and Woodward, J. (2007). A robust and adaptive approach to the nesting of irregular shapes. *INFORMS Journal on Computing*, 19(4): 517–529.

Chaves, A. A., Resende, M. G. C., and Silva, R. M. A. (2024). A continuous-grasp random-key optimizer. In Sevaux, M., Olteanu, A.-L., Pardo, E. G., Sifaleras, A., and Makboul, S., editors, *Metaheuristics*, p. 15–20. Springer Nature Switzerland, Cham.

Coffman, E. G., Garey, M. R., and Johnson, D. S. (1996). Approximation algorithms for bin packing: A survey. In Hochbaum, D. S., editor, *Approximation Algorithms for NP-hard Problems*, p. 46–93. PWS Publishing Co.

Coffman Jr, E. G., Garey, M. R., and Johnson, D. S. (1984). Approximation algorithms for bin packing: A survey. *Approximation Algorithms for NP-hard Problems*, p. 46–93.

Correia, I., Gouveia, L., and da Gama, F. S. (2008). Solving the variable size bin packing problem with discretized formulations. *Computers Operations Research*, 35(6):2103–2113. ISSN 0305-0548. Part Special Issue: OR Applications in the Military and in Counter-Terrorism.

Cung, V.-D., Hifi, M., and Le Cun, B. (2000). A new branch-and-bound method for the two-dimensional bin packing problem with due dates. *European Journal of Operational Research*, 127(3):549–567.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, 44(2):145–159.

Epstein, L., Favrholdt, L. M., and Levin, A. (2006). Online variable-sized bin packing. *Theoretical Computer Science*, 361(2–3):256–265.

Epstein, L. and Levin, A. (2007). A survey on algorithms for the bin packing problem. *SIAM Journal on Optimization*, 17(3):681–701.

Falkenauer, E. (2005). Hybrid genetic algorithms for bin-packing and related problems. *Hybrid Metaheuristics*, p. 1–26.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series. W. H. Freeman. ISBN 9780716710448.

Gomes, F. A. and Resende, M. G. (2015). A parallel grasp algorithm for the three-dimensional bin packing problem. *Computers Operations Research*, 63:29–42.

DOI: 10.59254/sbpo-2025-212127
https://proceedings.science/p/212127?lang=pt-br

Gonçalves, J. F. and Resende, M. G. (2011). Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525.

Gonçalves, J. F. and Resende, M. G. C. (2012). A biased random-key genetic algorithm for a 2d and 3d bin packing problem. Technical report, AT&T Labs Research.

Haouari, M. and Serairi, M. (2009). Heuristics for the variable sized bin-packing problem. *Computers & Operations Research*, 36(10):2877–2884.

Hemmelmayr, V., Schmid, V., and Blum, C. (2012). Variable neighbourhood search for the variable sized bin packing problem. *Computers & Operations Research*, 39(2):1097–1108.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer. ISBN 978-3-540-40286-7.

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

Lodi, A., Martello, S., and Vigo, D. (2002b). Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 14(3):302–317.

Mahadevan, S. and Daniels, K. M. (1991). On the placement of irregular parts in industrial applications. *Computer-Aided Design*, 23(1):44–52.

Monaci, M. (2003). Algorithms for packing and scheduling problems. *4OR*, 1:85–87.

Pisinger, D. and Sigurd, M. (2005). The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, 2(2):154–167.

Pisinger, D. and Sigurd, M. (2007). Using decomposition techniques and constraint programming for solving the 2d bin-packing problem. *INFORMS Journal on Computing*, 19(1):36–51.

Resende, M. G. and Ribeiro, C. C. (2010). Grasp: Greedy randomized adaptive search procedures. In *Handbook of Metaheuristics*, p. 287–319. Springer.

Wei, L., Kang, R., and Zhou, Y. (2016). A variable neighborhood search algorithm for the variable-sized bin packing problem. *Computers & Operations Research*, 67:150–160.

Wäscher, G., Haußner, H., and Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183(3):1109–1130.

Zhang, D., Hong, S., Lau, H. C., Zeng, X., and Si, Y.-W. (2014). A hybrid heuristic algorithm for the 2d variable-sized bin packing problem. *European Journal of Operational Research*, 238(1): 95–103.

DOI: 10.59254/sbpo-2025-212127
https://proceedings.science/p/212127?lang=pt-br