

Paper Type: Original Article

Reinforcement Learning Applied to 2D Strip Packing Problem: A Q-Learning Approach

Shima Shabani¹ Ali Husseinzadeh Kashan¹

¹ Department of Industrial Engineering, Tarbiat Modares University, Tehran, Iran; shima_shabani@modares.ac.ir, a.kashan@modares.ac.ir.

Citation:

Received: 19 April 2026
Revised: 13 June 2026
Accepted: 28 August 2026

Shabani, Sh., & Husseinzadeh Kashan, A. (2026). Reinforcement learning applied to 2d strip packing problem: A Q-learning approach. *Supply Chain and Operations Decision Making*, 3(4), 267-276.

Abstract

This study addresses the two-dimensional rectangular strip packing problem (2D-SPP), a significant combinatorial optimization challenge. A novel approach combining Q-learning with the Bottom-Left-Fill (BLF) algorithm, referred to as QL-BLF, is proposed. Our method aims to enhance packing efficiency by optimizing the sequence and orientation of items. We evaluated the performance of the QL-BLF algorithm against the traditional BLF algorithm using two datasets of 10 and 15 items, each tested across 10 distinct series to ensure comprehensive and reliable results. The computational experiments demonstrated that the QL-BLF algorithm consistently outperformed the BLF algorithm by significantly reducing the strip height and minimizing free space. These findings underscore the effectiveness of reinforcement learning in enhancing traditional heuristic methods for solving complex packing problems.

Keywords: Combinatorial optimization, Reinforcement learning, 2D Strip packing problem, Q-learning, Bottom-left-fill.

1 | Introduction

The two-dimensional Strip Packing Problem (2D-SPP) is a classic combinatorial optimization problem where the goal is to pack a set of 2D rectangular items into a strip of fixed width and infinite height in such a way that the height of the strip used is minimized. The items must be placed without overlapping, and all items must lie entirely within the bounds of the strip [1].

The 2D-SPP is crucial in numerous industries, including textiles, metalworking, and woodworking, where large sheets or rolls of material must be efficiently cut into smaller pieces to minimize waste. Additionally, in logistics and shipping, this problem aids in the efficient packing of items into containers or boxes, optimizing

space usage and reducing shipping costs by determining the optimal arrangement of items within the packaging [2], [3]. Solving the 2D packing problem is computationally difficult because it is NP-hard. To address this challenge, numerous optimization algorithms and techniques have been created, including heuristic, meta-heuristic approaches, and exact algorithms. However, exact algorithms often require extensive computation time and may not find the optimal solution within a reasonable timeframe. As a result, researchers have turned to heuristic algorithms, such as genetic algorithms, to achieve optimal solutions more efficiently [4]. The 2D level strip packing problem has received limited attention. In this paper conducted by Bezerra et al. [5], two mixed-integer linear programming models were adapted, Mrad's 2015 arc flow model for strip packing was rewritten, and valid inequalities were added to the 2004 Lodi et al. model. Computational results show that the enhanced Lodi et al. model outperforms other models in finding optimal solutions.

Additionally, Zhang et al. introduced a Block-Based Heuristic Search Algorithm (BBHSA) [6] for the 2D-SPP with guillotine cut constraints. Extensive experiments demonstrate that BBHSA is computationally effective, particularly achieving optimal solutions in zero-waste benchmark instances. To address the irregular strip-packing problem, which involves placing non-convex polygons on a rectangular strip to minimize length, Lastra-Díaz and Ortuño [7] introduced a new family of continuous MIP models based on vertical slices. These models outperform current MIP models and optimize ten instances in less than one hour. The methodology of Positions and Covering, presented in Reference [8], offers exact solutions for the two-dimensional strip packing problem with non-guillotine constraints. This method generates valid positions and selects the best configuration using a set-covering formulation. Experimental results highlight the method's effectiveness for small and medium-sized instances, successfully achieving optimal solutions for previously unsolved cases. A Genetic Programming (GP) system has been introduced by Burke et al. [9], evolving reusable heuristics for the 2D-SPP by exploring a space of heuristics instead of solutions. The GP hyper-heuristic can automatically generate competitive heuristics, enhancing the design process by intelligently combining good components. Moreover, Rakotonirainy and van Vuuren [10] introduce two enhanced metaheuristic approaches for the 2D-SPP: the first integrates simulated annealing with a heuristic construction algorithm, while the second applies simulated annealing directly to packing configurations. These algorithms outperform existing methods in a set of 1718 benchmark instances with varying characteristics.

In recent times, significant efforts have been directed toward addressing combinatorial optimization problems Bengio et al. [11] through the application of machine learning techniques. Data mining techniques have been employed to obtain reference values for the optimal objective function in challenging combinatorial optimization problems, specifically in the 2D-SPP. A regression model, particularly using random forest, assesses the quality of heuristic solutions by predicting strip height, demonstrating consistency and potential for application in other combinatorial optimization problems [12].

Packing problems often involve complex, dynamic environments where item shapes, sizes, and constraints vary. Reinforcement Learning (RL) algorithms can handle these complexities by continuously learning from interactions and adjusting strategies accordingly (Guerriero and Saccomanno [13]). RSRA, a hybrid heuristic algorithm for the 2D strip packing problem proposed by Zhu et al. [14], combines improved skyline-based scoring rules with a Deep Q-Network (DQN) for sequence optimization and placement. This approach reduces space waste, improves space utilization, and outperforms algorithms like GRASP, SRA, IA, and ISH across eight datasets, showing particularly strong performance on larger datasets with significant improvements in average gap percentages. Furthermore, to address the rectangular strip packing problem, a novel deep RL algorithm has been introduced, utilizing a pointer network and Maxrects-BL for piece placement. This algorithm surpasses or equals classical heuristic methods in most cases, delivering solution times under 1 second across all test instances and demonstrating strong generalization and efficiency [1].

This study bridges this gap by focusing on a comparative analysis of the traditional Bottom-Left-Fill (BLF) algorithm and a Q-learning-enhanced Bottom-Left-Fill (QL-BLF) algorithm. Integrating RL into BLF aims to improve packing efficiency, reduce waste, and optimize computational performance. By evaluating these algorithms, this research contributes to advancing hybrid solutions for 2D-SPP and demonstrates the potential of RL-enhanced heuristics in solving combinatorial optimization problems.

The structure of the present article is as follows: After the introduction, the second section reviews the literature. In the third section, we explain the research methodology. The fourth section presents the computational results, and the final section provides the overall conclusion of this study and discusses future research.

2 | Methodology

To explain our proposed methodology, we present the representation of the interior packing state at each placement step, along with the algorithm designed to optimize the sequence and orientation of items, incorporating the BLF heuristic and the Q-learning algorithm.

2.1 | Bottom-Left-Fill Heuristic

A common constructive algorithm for 2D cutting or packing problems is the BL heuristic, which places pieces in the lowest possible position and then as far left as possible without overlapping, aiming to minimize height, as illustrated by *Fig. 1*. While it is fast with $O(n^2)$ complexity, it cannot fill empty spaces (holes) between placed rectangles. To address this, Chazelle developed the BLF heuristic in 1983, which allows placement in empty spaces. Although BLF is more complex, with $O(n^3)$ complexity, it consistently performs as well as or better than the original BL heuristic by moving rectangles right and up until it finds a feasible position (*Fig. 1*; Oliveira et al. [15]).

The constructive rectangular packing algorithm takes the sheet width W , the number of items n , and the dimensions of the items $\{(w_1, h_1), (w_2, h_2), (w_3, h_3) \dots (w_n, h_n)\}$ as inputs, executes the packing process, and returns the layout along with the packing length L . *Table 1* shows this packing procedure (*Algorithm 1*).

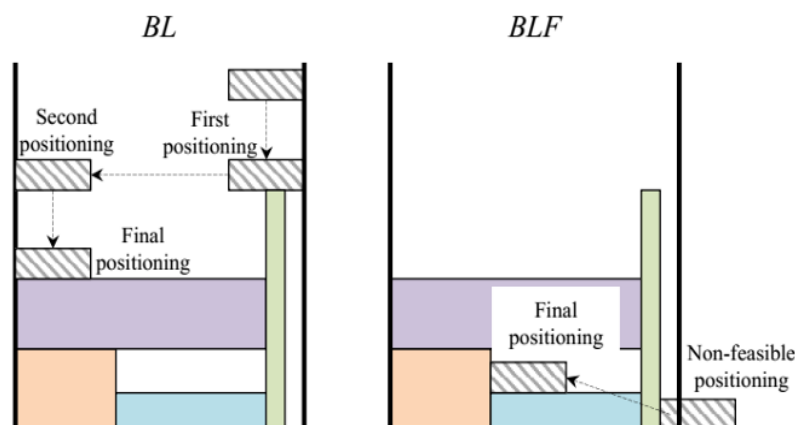


Fig. 1. Examples of application of the Bottom-Left (BL) and Bottom-Left-Fill (BLF) placement strategies.

Table 1. Bottom-Left-Fill algorithm.**Algorithm 1- The BLF algorithm.**

-
1. Input: A rectangular piece sequence (P1,P2,P3,...,Pn) and plate information
 2. Place piece P1 at the position of the plate origin O(0,0)
 3. For each piece P_i in the sequence:
 Move P_i downward until it touches another piece or the bottom of the plate
 Move P_i left until it touches another piece or the left side of the plate
 If there are empty spaces (holes) between already placed pieces, place P_i in the lowest and leftmost feasible position within these spaces
 4. Repeat the process for all pieces until they are packed into the plate
 5. Output: Layout and plate utilization rate
-

2.2| Q-Learning Algorithm

RL is a category of machine learning problems that involve an autonomous agent interacting with an environment. The agent perceives information about its current state and takes actions accordingly. In return, the environment responds with a reward signal, which may be positive or negative. The agent's goal is to maximize the expected cumulative reward signal throughout its interactions Ernst and Louette [16]. Q-learning, a widely recognized RL algorithm, can optimally solve Markov Decision Processes (MDPs). The core components of an MDP are the state, action, and reward. At the i -th stage of each episode, the state is represented by s_i , the action by a_i , and the reward by r_i . Thus, the trajectory of the n -stage MDP is described as follows;

$$s_0, a_1, s_1, r_1, a_2, s_2, r_2, \dots, a_n, s_n, r_n. \quad (1)$$

Fig. 2 illustrates the model of the corresponding n -stage MDP Fang, Rao, Zhao et al. [1]. In this model, s_0 denotes the initial state where no pieces are arranged, while s_n represents the terminal state of the MDP.

At the i -th stage, a_i and s_i refer to the action and state, respectively, where $i \in [1, n]$, and r_i indicates the immediate reward obtained by the piece in state s_i . The evaluation strategy, denoted as $\pi(s | a)$, can adopt methods like the greedy algorithm. The action a_i denotes the piece the agent selects in state s_{i-1} . Upon completing a_i , the environment transitions from state s_{i-1} to s_i , where $s_i = a_i$, representing the current piece number. In this study, the state is the set of items already placed, while the action is the item to be placed along with its orientation. The reward is represented as the reciprocal of the total layout length. *Fig. 2* provides a detailed explanation.

The state s_t is defined as the partial packing history, represented by the ordered sequence of dimensions of the rectangles that have already been placed. Formally, $s_t = [(h_1, w_1), (h_2, w_2), \dots, (h_t, w_t)]$, where (h_k, w_k) denotes the width and height of the k -th placed rectangle in its placed orientation. At each decision step, the action is defined as $a_t = (j_t, o_t)$, where j_t denotes the selected item (rectangle) to be placed, and $o_t \in \{0, 1\}$ is the rotation indicator. Specifically, $o_t = 0$ represents the original orientation (no rotation), while $o_t = 1$ denotes a 90° rotation before placement.

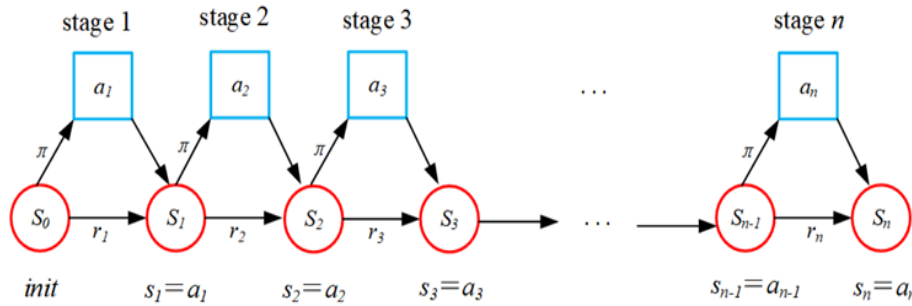


Fig. 2. Choice of items and Markov decision process.

In this paper, the reward r is assigned after completing a packing process, where $r_n = C/H \neq 0$. Here, C is a constant, and H represents the packing height achieved after placing all items. The immediate rewards for intermediate steps are set as $r_1 = r_2 = r_3 = \dots = r_{n-1} = 0$. Consequently, a lower packing height results in a higher packing utilization rate and a greater reward value. Within each episode, the agent traverses all items to be packed once, ensuring the packing process completes in a single run. Thus, each episode consists of n iterations to sequentially select and pack these n items.

In general classification, Fang, Rao, Zhao, et al. [1] categorize RL algorithms into two types: on-policy and off-policy algorithms. Off-policy algorithms use a data-generation strategy that differs from the strategy used for evaluation and improvement. In contrast, on-policy algorithms use the same strategy for both generating data and evaluating and refining it. Q-learning, an off-policy algorithm, updates through continuous interactions with the environment, enabling the agent to autonomously learn the action strategy at each step to maximize the accumulated reward. The long-term cumulative reward is represented by the $Q(s, a)$ value table, which serves as a guide for determining the packing sequence of the next item. The update mechanism of $Q(s, a)$ in Q-learning is expressed in the formula:

$$Q(s, a) = Q(s, a) + \alpha [r(s, a) + \gamma * \text{Max}_{a'} Q(s', a') - Q(s, a)].$$

where α is the learning rate, γ is the discount factor, and s' and a' correspond to the state and action of the subsequent stage, respectively.

Based on *Formula (1)*, we enhance the traditional Q-learning algorithm. Instead of updating the $Q(s, a)$ values at each stage during the episode, the updates are deferred to the end of the episode, where all state-action pairs are updated collectively. This process is carried out in reverse order, following the sequence $Q(s_{n-1}, a_n)$, $Q(s_{n-2}, a_{n-1})$, $Q(s_{n-3}, a_{n-2}) \dots Q(s_0, a_1)$. By adopting this approach, the $Q(s, a)$ values for all state-action pairs are effectively updated, ensuring that each episode's exploration experience is fully leveraged to improve decision-making in subsequent episodes. *Table 2* outlines the Q-learning algorithm.

Table 2. Q-Learning algorithm

Set values for learning rate α , discount rate γ , reward matrix R
Initialize $Q(s, a)$ to zeros
Repeat for each episode, do
Select state s randomly
Repeat for each step of episode, do
Choose a from s using an ϵ -greedy policy or Boltzmann policy
Take action a , obtain reward r from R , and next state s'
Update $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
Set $s = s'$
Until s is the terminal state
End do
End do

4 | Computational Experiments

The computational experiments demonstrate the superiority of the QL-BLF algorithm over the traditional BLF in solving the 2-dimensional rectangular strip packing problem. To evaluate the algorithms' performance, we used datasets with 10 and 15 items, with dimensions varying across 10 different series. Each series represents a unique set of item dimensions, as shown in *Tables 3* and *4*. *Tables 5* and *6* summarize the results for both algorithms.

Table 3. Dimensions of items for the 10-item dataset.

P1		P2		P3		P4		P5		P6		P7		P8		P9		P10	
w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h
3	3	3	1	6	3	9	2	3	2	3	14	3	13	3	2	2	3	3	3
5	3	5	3	5	3	2	10	2	10	2	10	3	10	2	7	5	3	5	11
3	4	2	4	3	14	3	4	2	4	2	6	2	4	2	4	12	4	3	4
4	2	4	12	4	10	4	12	4	12	4	12	4	10	4	12	4	2	4	2
1	5	1	5	2	5	1	5	1	5	1	7	1	5	1	5	11	5	9	5
5	4	5	4	7	4	5	2	5	4	5	4	15	4	5	1	5	4	5	4
3	3	3	3	3	3	3	3	3	3	3	3	3	3	10	3	3	3	3	3
11	2	11	2	11	2	11	2	11	2	10	2	11	2	11	2	2	2	11	2
13	3	13	3	13	3	3	3	6	3	6	3	6	3	6	13	13	3	13	3
1	13	1	13	1	5	3	11	1	12	1	11	1	9	1	11	1	12	1	15

Table 4. Dimensions of items for the 15-item dataset.

P1		P2		P3		P4		P5		P6		P7		P8		P9		P10	
w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h
2	2	2	2	2	12	2	4	12	2	2	9	2	2	2	2	2	2	2	1
5	3	5	3	5	3	5	3	5	2	5	3	5	14	5	3	5	3	5	3
1	6	1	6	10	6	1	12	1	6	1	6	10	6	1	12	1	15	1	6
4	2	4	2	4	1	4	2	4	2	4	11	4	2	4	2	4	2	4	2
1	5	1	5	1	5	1	5	1	5	1	5	1	5	1	5	1	5	1	5
10	4	10	4	10	4	10	4	9	4	10	4	4	4	10	4	10	4	10	4
3	3	3	13	3	13	3	9	3	6	3	13	3	13	3	9	3	10	3	10
11	1	1	1	8	1	1	1	1	1	1	4	1	1	1	1	7	1	5	1
10	3	10	3	10	3	10	2	10	3	10	3	10	3	10	3	10	3	11	3
11	5	11	5	5	5	11	5	11	5	11	5	11	5	11	5	11	1	10	1
2	6	2	6	2	6	2	6	2	7	2	6	2	6	2	6	2	6	2	6
3	8	3	8	3	8	3	8	3	12	3	8	3	8	1	8	3	8	3	8
2	2	2	2	2	2	5	12	2	2	2	5	2	2	3	2	2	2	2	2
5	4	5	4	5	4	5	4	5	4	5	4	15	4	5	4	5	4	5	4
11	2	10	2	10	2	10	5	10	2	13	2	7	2	10	2	9	2	10	2

Across multiple test scenarios, the QL-BLF algorithm consistently outperformed BLF by minimizing total strip height and unused free space. For the 10-item datasets, QL-BLF achieved significant improvements, such as reducing strip height from 20 to 16 units in one case while minimizing free space from 48 to 8 units. Similarly, in the 15-item datasets, QL-BLF not only lowered the total strip height (e.g., from 32 to 31 units in the first series) but also drastically reduced free space (e.g., from 117 to 17 units in the ninth series). These results highlight QL-BLF's adaptive learning capability, which dynamically optimizes packing strategies.

Table 5. Summary of results for both algorithms for the 10-item dataset.

Algorithm		BLF		QL-BLF	
Series	Metrics	Total Height	Free Space	Total Height	Free Space
	1	20	48	16	8
	2	27	86	24	56
	3	26	32	25	22
	4	23	44	21	24
	5	24	72	23	62
	6	24	33	23	23
	7	26	20	26	20
	8	28	53	25	23
	9	24	24	24	24
	10	29	56	26	26

Table 6. Summary of results for both algorithms for the 15-item dataset.

Algorithm		BLF		QL-BLF	
Series	Metrics	Total Height	Free Space	Total Height	Free Space
	1	32	55	31	45
	2	33	47	29	7
	3	37	40	36	30
	4	47	113	42	63
	5	32	33	30	13
	6	40	52	38	32
	7	44	38	44	38
	8	36	97	32	57
	9	36	117	26	17
	10	27	36	25	16

For example, *Fig. 3* and *Fig. 4* illustrate the packing arrangements achieved by both algorithms for the first two datasets for 10 and 15 items. These visualizations clearly demonstrate the superior space utilization of the QL-BLF algorithm, resulting in more compact and efficient packing than traditional BLF.

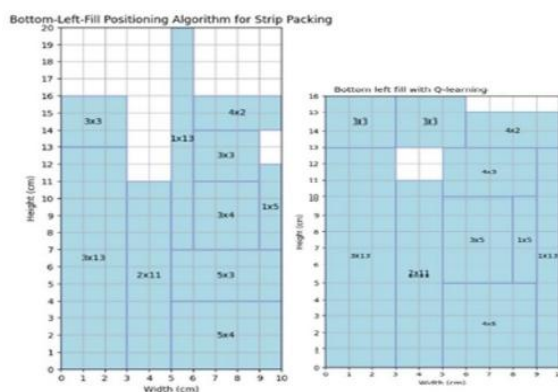


Fig. 3 . Visual Comparison of BLF and QL-BLF Algorithms for the first series of 10-item dataset.

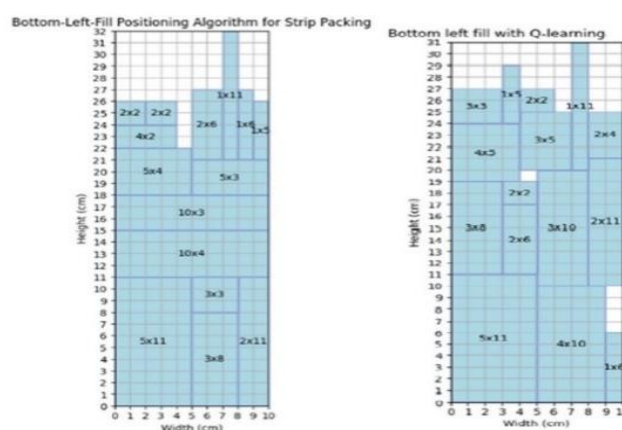


Fig. 4 . Visual Comparison of BLF and QL-BLF Algorithms for the first series of 15-item dataset.

5 | Conclusion

The QL-BLF algorithm exhibits significant improvements over the traditional BLF algorithm in addressing the 2D Strip Packing Problem (2D-SPP). Through the integration of Q-learning, the QL-BLF algorithm demonstrates enhanced adaptability and optimization capabilities, resulting in substantial reductions in strip height and free space. Computational experiments conducted on datasets with 10 and 15 items across multiple episodes consistently highlight the superiority of QL-BLF in achieving more efficient packing solutions. For instance, in several test scenarios, the algorithm not only minimized the strip height but also drastically reduced unused free space, thereby maximizing packing efficiency.

These findings underscore the potential of RL techniques in overcoming the limitations of traditional heuristic methods, offering a dynamic and adaptive approach to combinatorial optimization problems. QL-BLF's success in 2D packing scenarios paves the way for future research to explore its applicability to more complex packing challenges, such as irregular shapes or multi-dimensional packing problems. By further refining and extending the methodology, integrating Q-learning could revolutionize optimization strategies in packing and beyond.

Acknowledgments

The authors sincerely appreciate the valuable contributions and support provided throughout this research.

Funding

No specific funding was received for this research.

Data Availability

The data used in this study are available from the corresponding author upon reasonable request.

References

- [1] Fang, J., Rao, Y., Zhao, X., & Du, B. (2023). A hybrid reinforcement learning algorithm for 2D irregular packing problems. *Mathematics*, 11(2), 327. <https://www.mdpi.com/2227-7390/11/2/327>
- [2] Gilmore, P. C., & Gomory, R. E. (1965). Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13(1), 94–120. <https://doi.org/10.1287/opre.13.1.94>
- [3] Hopper, E., & Turton, B. C. H. (2001). A review of the application of meta-heuristic algorithms to 2D strip packing problems. *Artificial Intelligence Review*, 16(4), 257–300. <https://doi.org/10.1023/A:1012590107280>
- [4] Tian, R., Kang, C., Bi, J., Ma, Z., Liu, Y., Yang, S., & Li, F. (2023). Learning to multi-vehicle cooperative bin packing problem via sequence-to-sequence policy network with deep reinforcement learning model. *Computers & Industrial Engineering*, 177, 108998. <https://doi.org/10.1016/j.cie.2023.108998>
- [5] Bezerra, V. M. R., Leao, A. A. S., Oliveira, J. F., & Santos, M. O. (2020). Models for the two-dimensional level strip packing problem-a review and a computational evaluation. *Journal of the Operational Research Society*, 71(4), 606–627. <https://doi.org/10.1080/01605682.2019.1578914>
- [6] Zhang, H., Yao, S., Zhang, S., Leng, J., Wei, L., & Liu, Q. (2024). A block-based heuristic search algorithm for the two-dimensional guillotine strip packing problem. *Engineering Applications of Artificial Intelligence*, 134, 108624. <https://doi.org/10.1016/j.engappai.2024.108624>
- [7] Lastra-Díaz, J. J., & Ortuño, M. T. (2024). Mixed-integer programming models for irregular strip packing based on vertical slices and feasibility cuts. *European Journal of Operational Research*, 313(1), 69–91. <https://doi.org/10.1016/j.ejor.2023.08.009>
- [8] Cid-Garcia, N. M., & Rios-Solis, Y. A. (2021). Exact solutions for the 2d-strip packing problem using the positions-and-covering methodology. *Plos One*, 16(1), e0245267. <https://doi.org/10.1371/journal.pone.0245267>
- [9] Burke, E. K., Hyde, M., Kendall, G., & Woodward, J. (2010). A genetic programming hyper-heuristic approach for evolving 2-D strip packing heuristics. *IEEE Transactions on Evolutionary Computation*, 14(6), 942–958. <https://www.graham-kendall.com/papers/bhkw2010b.pdf>
- [10] Rakotonirainy, R. G., & van Vuuren, J. H. (2020). Improved metaheuristics for the two-dimensional strip packing problem. *Applied Soft Computing*, 92, 106268. <https://doi.org/10.1016/j.asoc.2020.106268>
- [11] Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: A methodological tour d’horizon. *European Journal of Operational Research*, 290(2), 405–421. <https://doi.org/10.1016/j.ejor.2020.07.063>
- [12] Júnior, A. N., Silva, E., Gomes, A. M., Soares, C., & Oliveira, J. F. (2019). Data mining based framework to assess solution quality for the rectangular 2D strip-packing problem. *Expert Systems With Applications*, 118, 365–380. <https://doi.org/10.1016/j.eswa.2018.10.006>
- [13] Guerriero, F., & Saccomanno, F. P. (2023). A machine learning approach for the bin packing problem. *2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)* (Vol. 1, pp. 436–441). IEEE. <https://doi.org/10.1109/IDAACS58523.2023.10348703>
- [14] Zhu, K., Ji, N., & Li, X. D. (2020). Hybrid heuristic algorithm based on improved rules & reinforcement learning for 2d strip packing problem. *IEEE Access*, 8, 226784–226796. <https://doi.org/10.1109/ACCESS.2020.3045905>

-
- [15] Oliveira, J. F., Neuenfeldt, A., Silva, E., & Carravilla, M. A. (2016). A survey on heuristics for the two-dimensional rectangular strip packing problem. *Pesquisa Operacional*, 36(2), 197–226. <https://doi.org/10.1590/0101-7438.2016.036.02.0197>
- [16] Ernst, D., & Louette, A. (2024). Introduction to reinforcement learning. *Feuerriegel, S., Hartmann, J., Janiesch, C., and Zschech, P*, 111-126. https://damien-ernst.be/wp-content/uploads/2026/02/introduction_to_reinforcement_learning.pdf