Computing Maximum Polygonal Packings in Convex Polygons Using Best-Fit, Genetic Algorithms and ILPs

Alkan Atak
TU Dortmund, Germany

Kevin Buchin
TU Dortmund, Germany

Mart Hagedoorn
TU Dortmund, Germany

Jona Heinrichs
TU Dortmund, Germany

Karsten Hogreve
TU Dortmund, Germany

Guangping Li
TU Dortmund, Germany

Patrick Pawelczyk
TU Dortmund, Germany

Abstract

Given a convex region P and a set of irregular polygons with associated profits, the *Maximum Polygon Packing Problem* seeks a non-overlapping packing of a subset of the polygons (without rotations) into P maximizing the profit of the packed polygons. Depending on the size of an instance, we use different algorithmic solutions: integer linear programs for small instances, genetic algorithms for medium-sized instances and a best-fit approach for large instances. For packing rectilinear polygons we provide a dedicated best-fit algorithm.

2012 ACM Subject Classification Theory of computation → Packing and covering problems

Keywords and phrases Polygon Packing, Nesting Problem, Genetic Algorithm, Integer Linear Programming

Digital Object Identifier 10.4230/LIPIcs.SoCG.2024.83

Category CG Challenge

Supplementary Material *Software (Source Code)*: `https://github.com/traMectory/MPP`
archived at `swh:1:dir:374858a00660cab7de2d2ae6b0e4122a98c52e10`
Software (Source Code): `https://github.com/Alkan-Atak/AtrisBlockPacking`
archived at `swh:1:dir:93cb74626fe1c038bd6e3571e001bfc7e2fd469f`

1 Introduction

Packing problems ask to pack a set of items into a larger container such that no items overlap each other. Packing is a classic and extensively-studied geometric optimization problem. In the *Maximum Polygon Packing Problem* (MPPP), the input is a convex region P in the plane and a set of simple polygons $Q_i, \ldots, Q_n$ that has associated profit π_i with each polygon Q_i. The objective is to find a non-overlapping packing of a subset of the polygons into P such that the sum of the profits of the packed items is maximized.

Most existing practical algorithms for packing simple polygons have been developed for the strip-packing problem, in which all polygons have to be packed into a strip of a certain width, and the objective is to minimize the height [13]. We adapt several of these algorithms to the MPPP and experimentally evaluate them on the instances of the 2024 Computational Geometry Challenge (CG:SHOP 2024).[1] For more details on the MPPP, see the survey paper by Fekete et al. [7] and for other proposed solutions for this challenge we refer to [6, 10, 15].

[1] `https://cgshop.ibr.cs.tu-bs.de`

© Alkan Atak, Kevin Buchin, Mart Hagedoorn, Jona Heinrichs, Karsten Hogreve, Guangping Li, and Patrick Pawelczyk;
licensed under Creative Commons License CC-BY 4.0

2 Algorithmic methods

2.1 Best-Fit Algorithm

A simple heuristic for packing problems is the *bottom-left* method [1], which takes a sequence of items and iteratively packs the items into the bottom-left corner of the remaining space. This approach can be combined with a *best-fit* strategy [3], which aims to pack the item next that best fits into a given corner. This strategy has in particular been successfully used for rectilinear items [12].

We adapt these approaches to our setting. A specific challenge is the trade-off between items that fit well and items with large profit. As measure for how well an item fits into the bottom-left corner, we use the distance d between the bottom-left corner of the bounding box of the item and the bottom-left corner of the container, with a smaller distance corresponding to a better fit. By $d_{\min}$ and $d_{\max}$ we denote the minimum and maximum such distance among all unplaced items. As measure for how profitable an item is we simply use the area-to-value ratio v with a smaller ratio being preferable. By $v_{\min}$ and $v_{\max}$ we denote the minimum and maximum area-to-value ratio among all unplaced items. A crucial component of our scoring system is the introduction of a parameter p (between 0 and 2) that allows for the adjustment of the weight assigned to the area-to-value ratio. Thus, from the remaining items we place the one with the lowest score s, where

$$s = p \times \frac{v - v_{\min}}{v_{\max} - v_{\min}} + \frac{d - d_{\min}}{d_{\max} - d_{\min}}\,, \tag{1}$$

assuming the denominators are non-zero. If $v_{\max} - v_{\min}$ or $d_{\max} - d_{\min}$ equals 0, we simply choose the item with smallest d or largest v, respectively.

The sequential arrangement of items requires a meticulous evaluation of each item within the given problem instance. To further refine the packing configuration, we introduce a preprocessing step wherein we sort the items according to their area-to-value ratio and selectively remove a proportion of the least valuable items. This strategic elimination ensured a greater utilization of more valuable items during the packing process. To adjust the proportion of items removed, we introduced a parameter, which was optimized experimentally.

For problem instances with rectilinear items only, specifically the `atris` instances, we avoid utilizing Minkowski sums, commonly used for calculating nofit polygons (NFPs) [2], which are crucial for determining feasible placements for each item. For these instances we simplified the computation by determining NFPs for rectangles and subsequently combining these. This simplification proved effective in improving the running time without compromising the quality of our packing solutions. In order to also simplify updating a general polygon's feasible placement positions, we first generate a decomposition into convex parts for each polygon. This speeds up the calculation of the NFPs for two given polygons by taking the union of the Minkowski sums of their convex parts.

2.2 Genetic Algorithm

When iteratively packing items, the sequence in which polygons are inserted plays a crucial role in the overall quality of the solution. Finding a good insertion sequence is computationally challenging due to the large number of possible permutations to consider. Consequently, metaheuristics are employed to navigate the complex solution space efficiently. In particular genetic algorithms (GAs) have been used successfully for packing rectangular shapes [9] as well as strip-packing of general polygons [11]. Since the genetic algorithm is only used to determine the order in which the objects are considered, it is easily adapted to our setting.

A genetic algorithm requires defining *individuals*, an initial *population*, a *fitness* function for the selection procedure, a *crossover* and a *mutation* method. The individuals in our GA's population are the permutations of all polygons of a given problem instance. The permutation is used as insertion order. Given such an order, our simple model then uses a bottom-left strategy to pack the items. Since the bottom-left strategy may give suboptimal solutions independent of the insertion order, we extended the solution space by assigning a packing strategy to every single item individually. The additional packing strategies are the bottom-right, top-left, and top-right strategies, which are analogous to the bottom-left strategy. We refer to this strategy as *corner strategy*.

For the initial population we use the following permutations: Firstly, the items sorted according to their values as well as their area in ascending and descending order; secondly, the permutation obtained by running a modified best-fit algorithm from Section 2.1, where the score function minimizes the vertical position of the bottom-left corner of the bounding boxes; thirdly, random permutations. If the corner strategy is used we additionally randomize the placement strategy per item, except for the individual from the modified best-fit algorithm. As fitness function we simply use the final packing score. After each generation's fitness is evaluated, tournament selection is performed to populate the following generation.

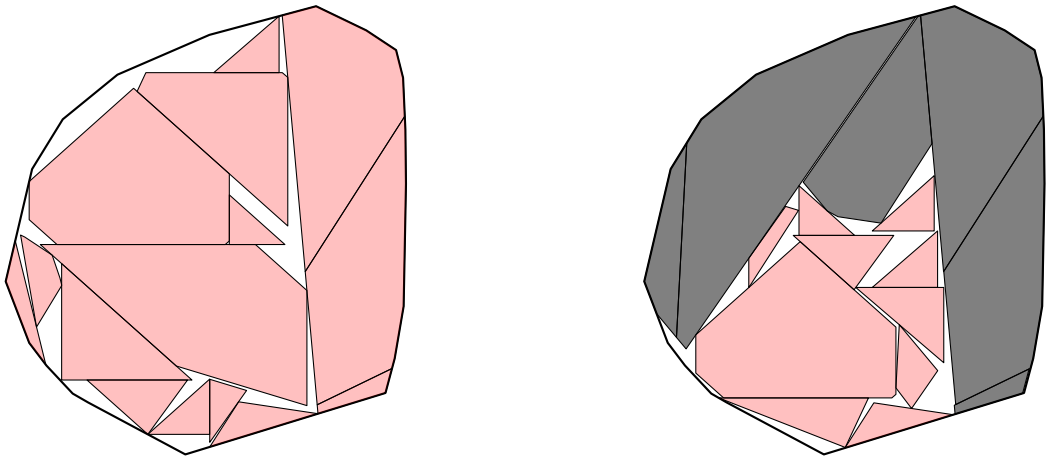
We use two standard crossover methods, namely Order Crossover (OX) and Partially Mapped Crossover (PMX). As mutation of an individual, either the positions of two of its items in the insertion order are swapped or the current packing strategy changes for example from bottom-left to top-right. We conducted experiments with different probability values for mutation ranging from 1% to 40%, and for crossover ranging from 50% to 100%.

For instances with less than 50 items we observed that providing an initial solution by a best-fit algorithm does not guarantee a better overall fitness, even within the initial population. Results varied for different instances regarding crossover and mutation. For instances with more than 50 items, however, providing better initial solutions leads to better overall results with generally better values achieved by higher mutation probabilities.

2.3 Integer Linear Program

For very small instances we use a integer linear programming (ILP) formulation based on similar models for the strip-packing problem by Cherri et al. [4], Rodrigues et al. [16] and Díaz & Ortuño [14]. More specifically, we use non-overlapping constraints based on NFPs, see [4, Constraints (16)-(20)] for details. These models require a convex decomposition of the region outside of a NFP, and the main difference between the models is how this is achieved. The NFP covering model (NFP-CM) [4] has overlapping regions, resulting in symmetrical solutions. To address this, its improved version I-NFP-CM [16] adds an additional constraint for each region to make them disjoint. In contrast, NFP-CM-VS [14] divides the regions into vertical slices, which are also disjoint but require fewer constraints.

We adapted all of these models to our setting. Compared to the strip-packing formulation, we firstly need to incorporate the convex container and secondly need binary variables p_i with $p_i = 1$ representing that we pack the item i. Our objective is then to maximize $\sum_{i=1}^{n} \text{value}_i \cdot p_i$. We use the big-M method to disable constraints for items i when $p_i = 0$ [5]. As the instances used were too large to be solved optimally by our formulation, in Section 3 we report on the best feasible solutions obtained after a certain amount of time. We also used our ILP to complete partial solutions (Figure 1).



(a) No initial partial solution, obtaining a score of 1446.

(b) Initial partial solution (dark grey items), obtaining a score of 1518.

Figure 1 Two solutions obtained by the ILP for `jigsaw_rcf4_45bf920f_31`. In the second solution, the dark grey items were packed manually, the ILP was used to select the remaining items.

3 Practical computation

3.1 System Specification

The best-fit algorithm for rectilinear items is written in Java, whereas the genetic and best-fit algorithms for arbitrary items are written in C++ using the CGAL 5.6[2] [17, 18] and Clipper2[3] libraries, in particular for computing Minkowski sums. Moreover, the ILP models were solved using Gurobi 11 [8]. The computations presented were performed on a set of 8 similarly equipped computers, all with an 8-core AMD 7 3700X processor running at 4.4 GHz.

3.2 Benchmark Data

We tested our implementations using the instances provided for CG:SHOP 2024. We divide the instances into rectilinear instances and arbitrary instances sorted by size. We denote small instances as having less than 100 items, medium instances as having 100 up to but not including 500 items, and large instances as having 500 items or more.

3.3 Experimental Results

The findings presented in this section are the result of systematic experiments and, hence, can slightly vary from the results presented in the official CG Challenge 2024 rankings, which were obtained through exploratory experimentation.

As explained in Section 2.1, the best-fit algorithm utilizes two different means of calculating NFPs depending on the specific instances. As can be seen in Figure 2, the algorithm for rectilinear polygons outperforms the best-fit algorithm in terms of run-time. Furthermore, our algorithm for the rectilinear instances performed especially well, as demonstrated by the fact that this algorithm found the best solution for 21 out of 29 rectilinear instances in the contest, including the example from Figure 3.

[2] `https://www.cgal.org/`
[3] `https://angusj.com/clipper2`

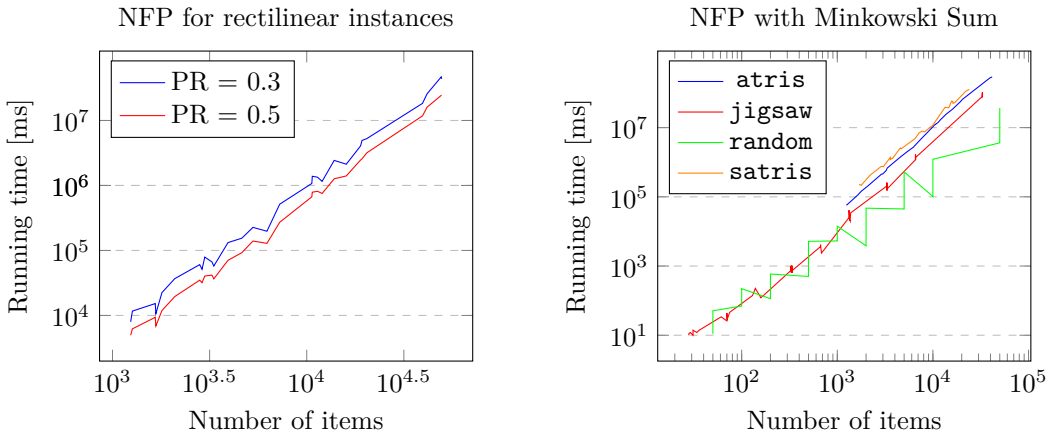


Figure 2 Running times of the best-fit algorithm (here PR stands for percentage removed as described in Section 2.1).

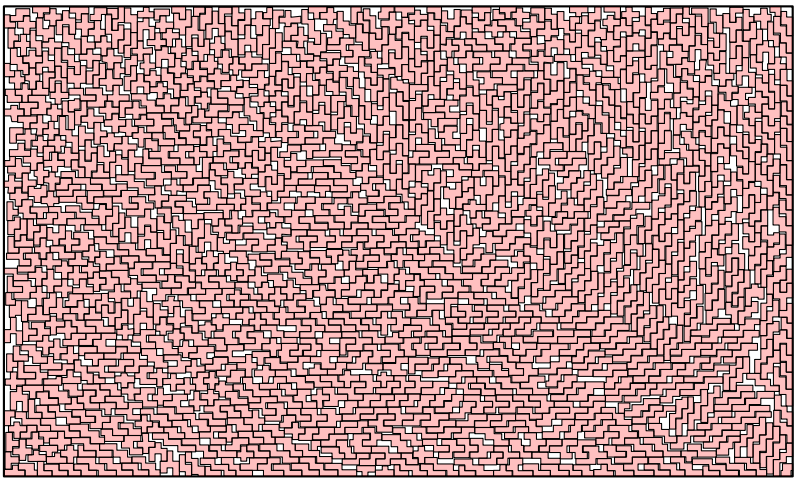


Figure 3 Example solution for `atris2812` (rotated 90 degrees).

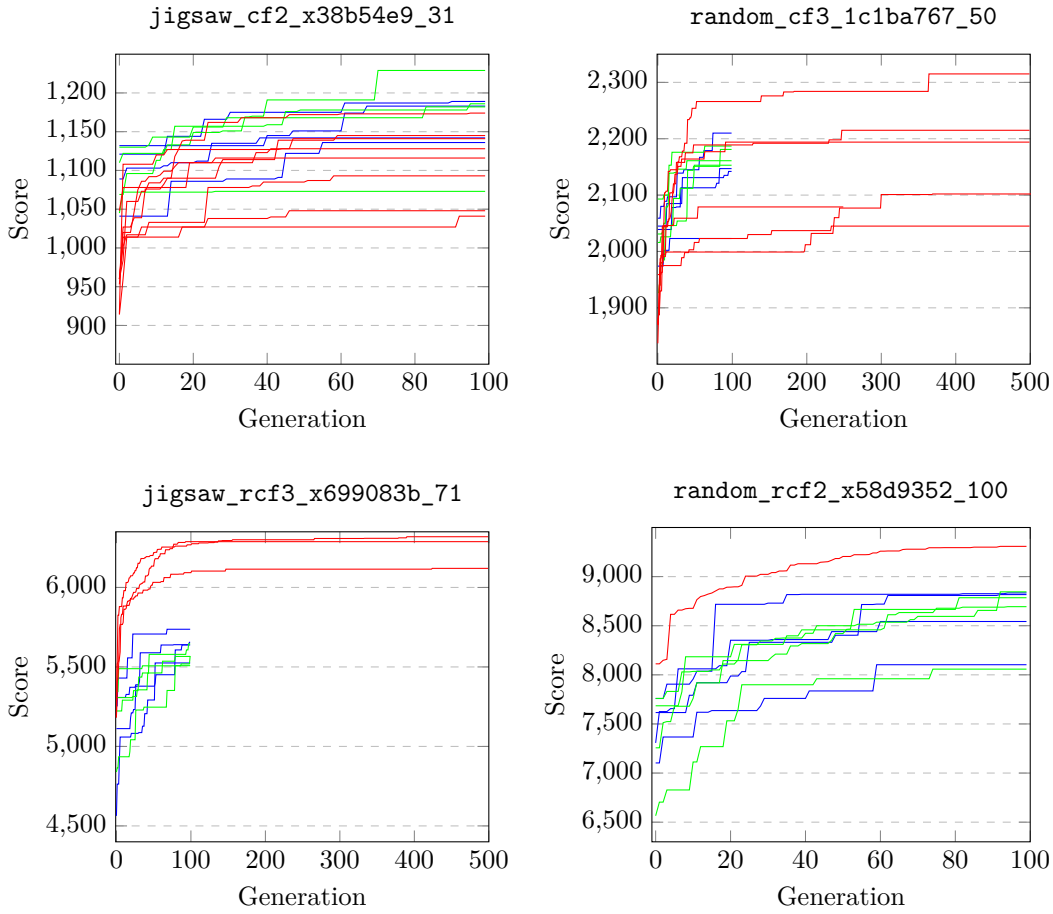


Figure 4 Evolution of maximum score using the genetic algorithms. Red lines indicate a 60% crossover value with swapping items in the mutation phase, green a 100% crossover value with changing placement strategies, and blue a 100% crossover value without mutation (see Section 2.2).

The algorithm we considered next is the genetic algorithm as described in Section 2.2. Figure 4 shows how the maximum score evolves over the number of generations, the graphs indicate that after roughly 100 generations we only obtain diminishing returns. However, due to slower running times, we applied this algorithm only to small and medium-sized instances.

The final algorithm presented in this paper (Section 2.3) is the ILP model. Due to long running times, this model could only be executed on small instances. Even then, exact solutions were not found and the best solution was returned after a computation time of 4 hours, in which for most instances between 10^6 and $5 \cdot 10^6$ branch-and-cut nodes were explored by the ILP solver. Furthermore, as can be seen in Figure 5, there seems to be no significant difference between the results for the different ILP variants.

Finally, we provide a brief comparison between the various methods in Figure 6, where we see that the ILP outperforms the genetic algorithm, which outperforms the best-fit algorithm.

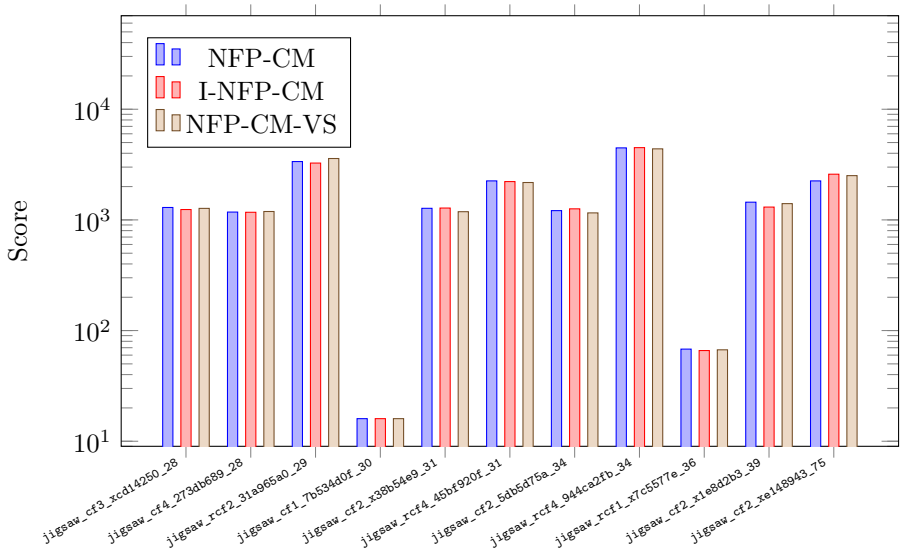


Figure 5 Comparison of results from different ILP models.

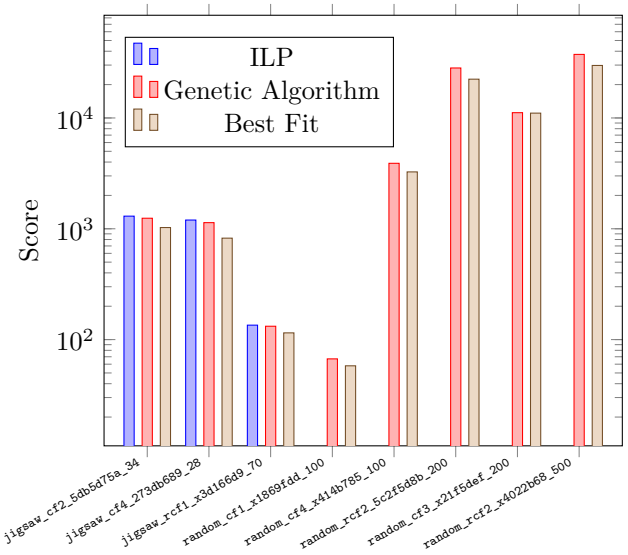


Figure 6 Comparison of results of the genetic algorithm, best-fit algorithm, and ILP.

4 Conclusion

We presented three different approaches to solving the Maximum Polygon Packing Problem (MPPP): an ILP solver, a genetic algorithm, and a best-fit algorithm. We evaluated our approaches on the datasets of the CG Challenge 2024. Solving the ILP resulted in the highest scores for small instances but is the least scalable. Also small instances we did not solve optimally. Finding more scalable ILP formulations for MPPP is a challenging open problem.

For instances too large for the ILP, we used a genetic algorithm. This approach yields good results for medium-sized instances. However, since to compute the fitness of an individual we need to actually pack the items, this approach slows down considerably for large instances. To remedy this situation, an alternative fitness function or a method for quickly estimating the score would be required.

Finally, for the remaining instances, the best-fit algorithm can be used, as it is the most scalable and can find a solution for all instances. In particular for instances with rectilinear polygons, our best-fit algorithm outperformed other approaches in the contest. An interesting direction for further research is to combine the various approaches. For example, an ILP could be used within the best-fit algorithm to solve smaller subproblems optimally.

References

1 Brenda Baker, Ed Coffman, and Ronald Rivest. Orthogonal packings in two dimensions. *SIAM J. Comput.*, 9:846–855, November 1980. doi:10.1137/0209064.

2 Julia A. Bennell and Xiang Song. A comprehensive and robust procedure for obtaining the nofit polygon using minkowski sums. *Computers & Operations Research*, 35(1):267–281, 2008. Part Special Issue: Applications of OR in Finance. doi:10.1016/j.cor.2006.02.026.

3 Edmund K. Burke, Graham Kendall, and Glenn Whitwell. A new placement heuristic for the orthogonal stock-cutting problem. *Operations Research*, 52(4):655–671, 2004. doi:10.1287/opre.1040.0109.

4 Luiz H. Cherri, Leandro R. Mundim, Marina Andretta, Franklina M.B. Toledo, José F. Oliveira, and Maria Antónia Carravilla. Robust mixed-integer linear programming models for the irregular strip packing problem. *European Journal of Operational Research*, 253(3):570–583, 2016. doi:10.1016/j.ejor.2016.03.009.

5 Richard Cottle, Mukund N. Thapa, et al. *Linear and nonlinear optimization*, volume 253. Springer, 2017. doi:10.1007/978-1-4939-7055-1.

6 Guilherme Dias da Fonseca and Yan Gerard. Shadoks approach to knapsack polygonal packing. In *Symposium on Computational Geometry (SoCG)*, volume 293 of *LIPIcs*, pages 84:1–84:9, 2024. doi:10.4230/LIPIcs.SoCG.2024.84.

7 Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Stefan Schirra. Maximum polygon packing: The cg:shop challenge 2024, 2024. arXiv:2403.16203.

8 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL: https://www.gurobi.com.

9 Africa Gómez and Daniel de la Fuente. Resolution of strip-packing problems with genetic algorithms. *Journal of the Operational Research Society*, 51(11):1289–1295, November 2000. doi:10.1057/palgrave.jors.2601019.

10 Martin Held. Priority-driven nesting of irregular polygonal shapes within a convex polygonal container based on a hierarchical integer grid. In *Symposium on Computational Geometry (SoCG)*, volume 293 of *LIPIcs*, pages 85:1–85:6, 2024. doi:10.4230/LIPIcs.SoCG.2024.85.

11 Elizabeth Hopper and Brian C.H. Turton. A review of the application of meta-heuristic algorithms to 2d strip packing problems. *Artificial Intelligence Review*, 16(4):257–300, December 2001. doi:10.1023/A:1012590107280.

12 Yannan Hu, Hideki Hashimoto, Shinji Imahori, and Mutsunori Yagiura. Efficient implementations of construction heuristics for the rectilinear block packing problem. *Computers & Operations Research*, 53:206–222, 2015. doi:10.1016/j.cor.2014.06.021.

13 Yannan Hu, Hideki Hashimoto, Shinji Imahori, and Mutsunori Yagiura. Practical algorithms for two-dimensional packing of general shapes. In Teofilo F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics, Second Edition, Volume 1: Methologies and Traditional Applications*, pages 585–609. Chapman and Hall/CRC, 2018. doi:10.1201/9781351236423-33.

14 Juan J. Lastra-Díaz and María Teresa Ortuño. Mixed-integer programming models for irregular strip packing based on vertical slices and feasibility cuts. *European Journal of Operational Research*, 313(1):69–91, 2024. doi:10.1016/j.ejor.2023.08.009.

15 Canhui Luo, Zhouxing Su, and Zhipeng Lü. A general heuristic approach for maximum polygon packing. In *Symposium on Computational Geometry (SoCG)*, volume 293 of *LIPIcs*, pages 86:1–86:9, 2024. doi:10.4230/LIPIcs.SoCG.2024.86.

16 Rodrigues, Marcos O., Cherri, Luiz H., and Mundim, Leandro R. MIP models for the irregular strip packing problem: new symmetry breaking constraints. *ITM Web Conf.*, 14:05, 2017. doi:10.1051/itmconf/20171400005.

17 The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6 edition, 2023. URL: https://doc.cgal.org/5.6/Manual/packages.html.

18 Ron Wein, Alon Baram, Eyal Flato, Efi Fogel, Michael Hemmer, and Sebastian Morr. 2D minkowski sums. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6 edition, 2023. URL: https://doc.cgal.org/5.6/Manual/packages.html#PkgMinkowskiSum2.