Combining Local Search and Genetic Algorithm for Two-Dimensional Guillotine Bin Packing Problems with partial sequence constraint

Filipe Souza and Diarmuid Grimes

Department of computer science
Cork Institute of Technology
Cork, Ireland
{filipe.luca-desouza@mycit.ie,diarmuid.grimes@cit.ie}

Abstract. In this work we consider a variant of the standard bin packing problem, known as the the two-dimensional guillotine cutting problem. This is a widely occurring problem in industry, for example for reducing the economic and environmental costs of glass production. This variant can also incorporate an additional partial sequencing constraint on batches of sub-items in comparison with the traditional bin packing problem. We propose a hybrid metaheuristic approach combining a dedicated genetic algorithm with a local search for solution refinement. The genetic algorithm has a number of dedicated aspects for the problem at hand, such as a complex phenotype representation with multi gene-types, a partial fitness function and different mutation operators for different parts of the chromosome tuple. Empirical results on public instances demonstrate the effectiveness of the hybrid algorithm and provide insight into interleaving local search with population-based approaches.

Keywords: Bin Packing Problem · Genetic Algorithm · Local Search · Phenotype · Genotype · Fitness Function

1 Introduction

The Two-Dimensional Bin Packing Problem (2BP) aims to allocate several rectangular items to a set of rectangular bins, in a way that the number of required bins is minimized. Most of the literature related to Cutting and Packing problems are based on heuristic [2, 8, 13, 14] or Branch-and-Bound [3, 12, 18, 19] approaches.

Cutting glass optimization is an important industrial specialization of the 2BP. The initial constraint in this problem is that the glass jumbo (bin) must be cut in guillotine cuts, i.e. orthogonal cuts going from one side of the sheet component to the other. In this work we consider the variant where the items can be rotated by 90 degrees and with partial sequencing constraints introduced by the ROADEF/EURO challenge 2018[1], where the items from the same order (stack) have to be cut in a specific sequence.

[1] https://www.roadef.org/challenge/2018/en/index.php

To address this issue a novel technique combining a dedicated genetic algorithm interleaved with a local search operator is proposed, denoted as GALS-2BP. To the best of our knowledge, the use of such a hybrid algorithm to solve the 2BP, is still insufficiently explored. The method proposed is investigated not only to generate good performance for the variant studied but also to provide further insight into the behavior of such hybridized techniques.

Investigating the potential of combining Local Search and Genetic Algorithm can produce impressive results, due to the complementary nature of the approaches. Genetic algorithms look at combining good solutions, but without looking at why a solution is good or bad. On the other hand, local search methods only consider one solution, but they specifically look for where the solution can be improved and make changes based on this.

Furthermore, this work digs deep into many components of Genetic Algorithms such as a complex phenotype representation of a chromosome with multiple gene-types, a partial fitness function, and the application of different mutation operators for different parts of a chromosome. Additional aspects investigated include crossover operators, elite survival, and periodic rediversification of the population to avoid stagnation.

The performance of the GALS-2BP approach was evaluated on the instances proposed in the ROADEF/EURO challenge 2018: Cutting Optimisation Problem, and shown to be competitive with the winner of that challenge: the *MBA** algorithm [12]. The latter is an anytime tree search algorithm with some simplistic bounds and pseudo-dominance properties. The evaluation also provided opportunity to develop a number of insights into the behavior of a hybrid approach such as GALS-2BP.

2 Problem definition

The guillotine cutting problem considered here can be defined as follows:

Definition 1 (Item). *An item i is a glass piece to cut, characterized by a pair (w_i, h_i) representing its width w and its height h.*

Definition 2 (Stack). *A stack $S = \{i_1, i_2, ..., i_j\}$ is an ordered sequence of items such that $i_1 <cut\ i_2 <cut\ ... <cut\ i_j$, basically meaning that item i_1 must be cut before item i_2, which is one of the constrains related to deliveries and item processing. For example in an instance with three stacks $s_1 = \{i_1, i_2\}$, $s_2 = \{i_3, i_4, i_5\}$, and $s_3 = \{i_6, i_7\}$, two feasible sequence of cutting can be define as below:*

- *The sequence $s_1, s_2, s_3, s_3, s_2, s_1, s_2$ means $i_1, i_3, i_6, i_7, i_4, i_2, i_5$.*
- *The sequence $s_2, s_3, s_2, s_1, s_3, s_1, s_2$ means $i_3, i_6, i_4, i_1, i_7, i_2, i_5$.*

Definition 3 (Batch). *The set of items to cut is called batch I. It corresponds to a customer order using the stack notation, the item set I can be partitioned into n stacks.*

$$I = \bigcup_{k=1}^{n} s_k$$

Definition 4 (Bin). *A bin b is characterized by a pair (W_b, H_b) representing its width W and its height H. The set of bins $B = \{b_1, b_2, ..., b_q\}$, where $b_1 < cut$ b_2 means that bin b_1 has to be used before the bin b_2. In addition the bin size is standardised.*

$$\forall a, b \in B : W_a = W_b$$

$$\forall a, b \in B : H_a = H_b$$

Definition 5 (Guillotine cut). *A guillotine cut is the cut that divides a nonempty subset p into two disjoint rectangular subsets p'_1 and p'_2 such that no item $i \in I$ is intersected by the cut between p'_1 and p'_2.*

Definition 6 (Cutting pattern). *A set cutting pattern $P = \{p_1, p_2, ..., p_t\}$ is a feasible solution, where $\forall p \in P$, p is a guillotine cutting pattern that divides a rectangular glass piece p_t in two new rectangular piece p_{t+1} and p_{t+2}. Figure 1 depicts an example of guillotine pattern for a bin and the order of the cuts.*

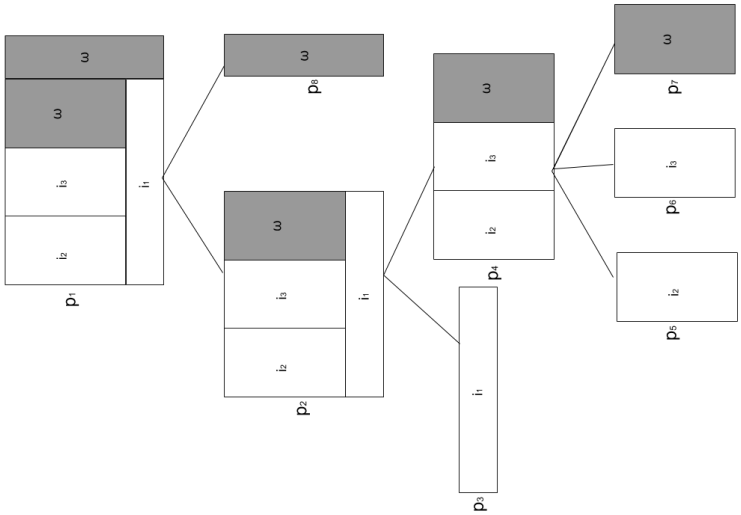

Fig. 1. Example of feasible cutting pattern diagram

Definition 7 (Waste). *A waste ω is a p, where $\forall i \in I : i \cap p = 0$. In Figure 1, p_7 and p_8 are examples of waste.*

$$\omega = p \Rightarrow \forall i \in I : i \cap p = 0$$

Definition 8 (Residual). *A residual R is the last element of P.*

$$R = p_{|P|}$$

The objective function is to minimize the geometrical loss of the cutting patterns applied to bins. In order to compute the loss area, glass leftover must

not be taken into account, since they can be reused after cutting. Only the residual in the last cutting pattern will be considered in order to simplify the residual management. This residual is represented by the waste at right of the last 1-cut performed in the last cutting pattern in a solution as described in definition 8. So that, the problem can thus be modeled as:

$x_{ib} \in \{0,1\}$ *should be* 1 *iff item* i *is packed into bin* b.

$y_b \in \{0,1\}$ *should be* 1 *iff bin* b *is used.*

$$\text{minimize} \quad \sum_{b \in B} H_b W_b y_b - (H_R W_R) - \sum_{i \in I} w_i h_i \tag{1}$$

$$\text{subject to:} \quad \sum_{i \in I} w_i h_i <= \sum_{b \in B} H_b W_b \tag{2}$$

$$\sum_{b \in B} x_{ib} = 1 \qquad \forall i \in I \tag{3}$$

$$\sum_{i \in I} w_i h_i x_{ij} \leq W_b H_b \qquad \forall b \in B \tag{4}$$

$$s \cap q = 0 \qquad \forall s, q \in S, s \neq q \tag{5}$$

$$y_{B_j} \geq y_{B_{j+1}} \qquad \forall j \in \{1, 2, ..., |B| - 1\} \tag{6}$$

$$j \notin p \ni i \qquad \forall i, j \in I, i \neq j, \exists p \in P \tag{7}$$

$$x_{ib} \in \{0,1\} \qquad \forall i \in I, \forall b \in B \tag{8}$$

$$y_b \in \{0,1\} \qquad \forall b \in B \tag{9}$$

Note that in our work, we consider a slight simplification of two-dimensional cutting glass problem proposed in the ROADEF/EURO challenge 2018. Namely the defects constraint of the challenge problem, where a bin may contain defects at given locations and that must not be included in our cut items, is not considered in this work.

3 Related Work

The 2BP has been a problem of interest for decades [6]. One of the most popular heuristic approaches to tackle the two-dimensional bin packing problem was proposed by Baker *et al.*[2], the well-known *bottom-left* (BL) heuristic. It tries to minimize the height of each piece in a rectangular bin, focusing on orthogonal and oriented packing.

Numerous variants of the bottom-left heuristic have been proposed since then (e.g. [8, 13]). Even though the recent literature have recommended block-based placement[14] for guillotine problems, due to the additional ordered sequence constraint in our problem, we chose item-based placement based on BL, similar to [10].

In terms of relevant GA approaches to the 2BP, Goncalves and Resende in [7] proposed a biased random-key genetic algorithm (BRKGA) approach. In their work a novel fitness function was also proposed which we have incorporated for

our evolutionary process fitness function. While, Hwang *et al.*in [9] used Genetic Algorithms as an approach to solving a variant of 2BP with guillotine constraint for retail shelf space design, in this variant the size of the items can be mutated in order to maximize the demand rate of the items, which is a function of the quantity displayed and the location in the shelf space.

To the best of our knowledge, the idea of combining Genetic Algorithm with local search approach to solve a specific problem was introduced in the late 80s with the term of Memetic Algorithms [16], and since then it has been the focus of many research works (See [17, 15, 5]). While in the matter of 2BP, Kierkosz and Luczak in [11] proposed an interesting approach that combines an evolutionary algorithm and a tree search, where the GA solution is used as an input for the tree search step. The same idea is implemented in this paper's approach, where the LS uses the GA solution to generate the final solution.

Finally, the current State of the art algorithm to solve the variation of 2BP presented in this work comes from [12], where they proposed the algorithm MBA*. The approach involves a tree search algorithm with aggressive pruning which diminishes over time.

4 Proposed Hybrid Algorithm

Figure 2 shows the macro process of the GALS-2BP algorithm developed in this work. As can be observed, the macro-structure is almost the same as the general GA, with only the addition of two new steps. The first one is the Replacement strategy that aims to avoid homogeneous population and the second one is a Local Search step that refines the GA solutions in order to improve the final solution. Thus, every GA step is guided by the cost of the solutions refined by the Local Search.

Chromosome Representation Using the standard GA terminology, a solution to a problem is given by a set of genes encoded in a chromosome. This solution can be represented directly or indirectly by the chromosome. When it is a direct representation, the chromosome will be the solution of the problem, referred to as a Genotype [20]. An indirect representation on the other hand, has the chromosome as the input to a particular procedure to generate the solution, that representation is known as a Phenotype [7]. In this work, the indirect representation was used given that the problem solution is too computationally expensive.

For this reason, the chromosome consists of three subgroups of genes that are combined to compute the fitness of the chromosome. Each chromosome is composed of $3N$ genes, where N is the number of items in the instance problem. The first N genes represents the sequence in which the items will be cut (this part of the chromosome works the same as a permutation problem encoding). The next N genes represent whether the item will be rotated or not. The final N genes represent the orientation of the cutting for the item (horizontal or vertical).

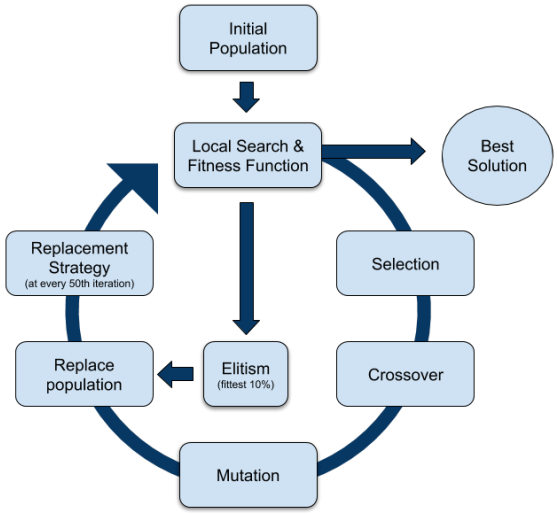

Fig. 2. Hybrid Algorithm diagram

Fitness Function The fitness function presented in equation 1 was implemented for the Genetic Algorithm. That function gives the final cost for the solution. However, it is necessary to place each item in its bin to calculate or update this fitness function, and this requires a high computational effort. Therefore this is impracticable for the Local Search step in computing the cost of each neighborhood move. Instead, the *occupation rate* of the bin is used to guide search. Equation 10 formally describes how the occupation rate is calculated for a given bin.

$$\frac{\sum_{i \in I} w_i h_i x_{ib}}{W_b H_b} \tag{10}$$

Initial Population A high level of diversity in the initial population is essential to avoid early stagnation in GAs. Thus, the initial population was created with a random set of item sequences and also randomly assigning the rotation position and the cutting orientation. Hence, the diversity of the population is ensured, while the Local Search step aids in the intensification.

Selection Method In order to achieve a balance between biasing towards fitter individuals (i.e. better solutions) and the necessity of being computationally efficient, the chosen selection method was Tournament Selection[1]. In particular, binary tournament selection was used, where two individuals are randomly chosen and the fitter of the two is selected.

Crossover Operator Due to the specificity of this problem, the chromosome is divided into three parts as previously stated. Hence, different crossover operators were developed for each part. For the permutation part two different permutation-based crossover operators were tested - uniform order-based crossover and PMX. While the two parts with Boolean valued genes followed the permutation operator chosen. In other words, when the PMX is used for permutation, two-point crossover is used for the Boolean parts, and uniform crossover was used for the uniform order-based operator case.

Mutation Operator The mutation operator works in the following way: First, a subset of offspring population is selected based on the mutation rate probability. Then, for each offspring, it is decided randomly which part (sequence, rotation, cut orientation) of the chromosome will be mutated. If the sequence part is chosen, the algorithm will apply a Reciprocal Exchange Mutation on that part of the chromosome, where two random genes are selected and their positions are swapped. Otherwise, Boolean genes are selected based on the mutation rate and their values are flipped.

Elite survival Elite survival was included such that the top x percent of the best solutions of the previous population are copied to the new population without modifications. Then, the rest of the new population is generated by the processes of crossover and mutation of the individuals from the previous population (including the elite chromosomes).

Local Search The Local Search is the last step to generate a new population. That step tries to optimize the wasted space left by the Genetic Algorithm solution. In order to do that, for each wasted space in the bins the Local Search algorithm searches in the whole neighbourhood composed by the next item of each stack. If there is more than one neighbour suitable for the waste space, it is randomly selected between the three best fit items and the selected item will be moved to fill the wasted space. The quality of the neighbor move is defined by the size of the smallest new wasted space that it creates after placing the item.

5 Experimental results

5.1 Experimental design

The configuration presented in Table 1 below was used to assess the performance of the GALS-2BP algorithm comparing it to the MBA* algorithm. Both algorithms were run under the same conditions and on the same machine (MacBook Pro 2.4 GHz Intel Core i5 machine with 8GB 1600 MHz DDR3 on macOS High Sierra), with a runtime cutoff of 10 minutes per instance. Furthermore, as the proposed approach has stochastic components, the presented results are the average of 10 runs with different seeds.

Table 1. Configurations parameters for the benchmark experiment.

Parameter	value
Iterations	10,000
Population Size	100
Selection	Tournament
Crossover	PMX
Mutation	Reciprocal Exchange
Mutation Rate	30%
Elitism Rate	10%
Local Search	100%

5.2 Preliminary Experiments

Replacement Strategy During initial experimentation, it was observed that the approach was not able to improve the solutions after the initial population for some instances. This is illustrated in Fig. 3, where for the instance X2 the algorithm was not able to improve the solution after the initial population even when Local Search was applied for more than 40% of the population. Analysing the problem, it was discovered that the population was rapidly becoming homogeneous due to the strong specificity of the Local Search Process. To overcome this issue, a Replacement Strategy component was developed.

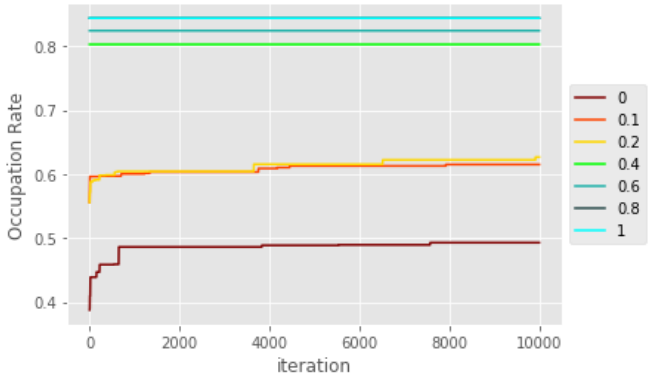

Fig. 3. Graph of occupation rate improvement over generations for instance X2 per different Local Search rates

The proposed Replacement Strategy was implemented based on that proposed by Bennell *et al.*[4]. At every fifty generations, any duplicated solution is overwritten by a new solution that is a copy of the best solution with some genes mutated. In this way it is possible to avoid premature convergence and also create diversity on the population without loss of specificity.

Chromosome Representation Initially a solution was represented by a chromosome was of N genes, where each gene was composed by the three parameters.

However, this was found to be ineffective as the sequencing constraint resulted in swapping of information between genes. In this way the children were losing a lot of genetic information from their parents. This problem was particularly prominent in instances where there were many items per stack. This motivated the representation with $3N$ genes given in the previous section.

Mutation Operator In the initial implementation, the mutation operator was divided into two parts. The first one dealt with the Boolean genes, where each gene swapped its value based on the mutation rate probability. The second part involved a permutation mutation operator. However, when analysed empirically it was observed that the operator behaviour was extremely sensitive to the size of the instance. In other words, the same mutation rate could generate a high number of mutations in a big instance, which made the search highly stochastic, while in a small instance it rarely generated any mutation. This was the motivation and inspiration behind the mutation operator described in the previous section.

5.3 Evaluation

Figure 4 shows the results obtained with the GALS-2BP algorithm compared to the MBA* solution in the set of instances with size ranging from 5 to 656 items. As depicted in the graph, the GALS-2BP achieves results near to the state-of-the-art in instances with up to 300 items. However, the gap tends to increase in larger instances. Nevertheless, the GALS-2BP still found solutions with around 90% of occupation rate, even in instances greater than 400 items.

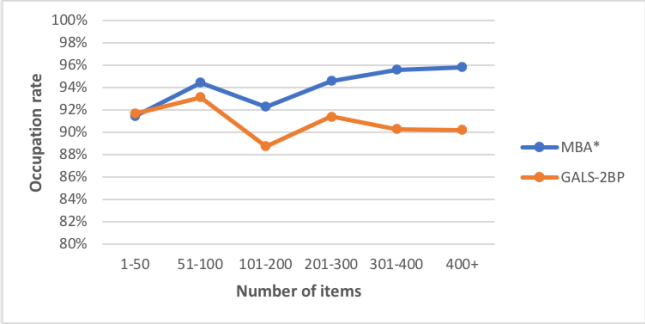

Fig. 4. Graphic shows the comparison of the GALS-2BP Algorithm and the MBA*, in different sizes of instances.

Figure 5 depicts the benchmark between the GALS-2BP algorithm and the MBA* algorithm per different average of stack size. The GALS-2BP managed to outperform the benchmark algorithm in instances with less than 5 items per stack, and also managed to catch results very near to the MBA* in instances with up to 15 items per stack on average. Unfortunately, when the number of items per stack starts to increase more than this, the difference tends to increase as well.

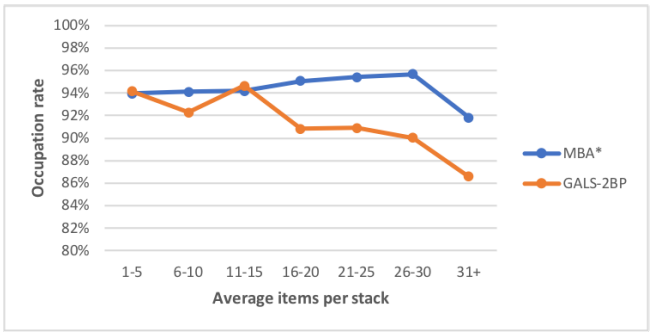

Fig. 5. Graphic shows the comparison of the GALS-2BP Algorithm and the MBA* per average Stack size.

6 Discussion

Even though the GALS-2BP did not outperform the overall results of the MBA* algorithm [12], it still found very good solutions, particularly for instances with fewer sequencing constraints. It is also important to highlight the consistency of the proposed hybrid algorithm, where in the total of fifty different executions for each instance the best result found throughout these executions always remained at most 2% above the average best solution for each instance.

The reason for the better performance on instances with fewer sequencing constraints is that the GA has more freedom to combine solutions. The population evolves with less noise in the crossover and mutation operator. The local search also has a larger neighbourhood to improve the solution when fewer sequencing constraints and therefore more possibilities for improvement.

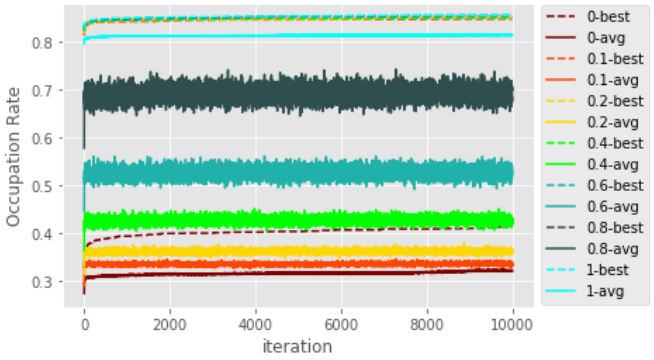

Fig. 6. Graph of the average occupation rate improvements along 10 executions over generations for instance X6 for best solution and average population at different Local Search rates

With regard to the crossover operators it was found that the PMX crossover performed better. This is due to the problem's high sensitivity to changes, since

the PMX crossover is has a higher capacity to preserve the parents characteristics.

It should also be noted that, since the Local Search is executed before the calculation of the fitness function, the goal of the GA is effectively to generate the best inputs for the Local Search step instead of looking for the best final solution. In order to demonstrate the impact of this, the Local Search was run for different proportions of the population (from 0% to 100% in steps of 10%).

The results of this are shown in Figure 6. One can clearly see, in the *average* occupation rate for the population along iterations, the unstable behaviour of the algorithm when the Local Search is used in only a part of the population. This instability happens because part of the population is trying to evolve to the best final solution while the other part is trying to evolve to the best input for the Local Search process. Thus, a worse solution from the GA can in some cases be a better input to the Local Search than a good solution would be. For example a worse solution can result in avoiding a local minimum in the Local Search.

7 Conclusion

In this paper we introduced a novel hybrid algorithm, combining Genetic Algorithms and Local Search, to solve the Two-Dimensional Guillotine Bin-Packing problem. Empirical analysis demonstrated the worth of the method, in particular finding an average occupation rate of 91% across a range of challenging instances. In future work, one are for improvement would be the development of placement strategies to help the process of decoding the phenotype chromosome into a better final solution. This would be extremely beneficial, particularly for instances with many sequencing constraints.

References

1. Abdulal, W., Ramachandram, S.: Reliability-Aware Genetic Scheduling Algorithm in Grid Environment. International Conference on Communication Systems and Network Technologies pp. 673–677 (2011)
2. Baker, B., Coffman, E., Rivest, R.: Orthogonal Packings in Two Dimensions. SIAM J. Comput. **9**, 846–855 (1980). https://doi.org/10.1137/0209064
3. Baldi, M.M., Crainic, T.G., Perboli, G., Tadei, R.: Branch-and-price and beam search algorithms for the Variable Cost and Size Bin Packing Problem with optional items. Annals of Operations Research **222**, 125–141 (2014)
4. Bennell, J.A., Soon Lee, L., Potts, C.N.: A genetic algorithm for two-dimensional bin packing with due dates. International Journal of Production Economics **145**, 547–560 (2013)
5. Fernández, A., Gil, C., Márquez, A.L., Baños, R., Montoya, M.G., Parra, M.: A memetic algorithm for two-dimensional multi-objective bin-packing with constraints. Proceedings of the 13th annual conference companion on Genetic and evolutionary computation - GECCO p. 341 (2011)

6. Gilmore, P.C., Gomory, R.E.: Multistage cutting stock problems of two and more dimensions. Operations research **13**, 94–120 (1965)
7. Gonçalves, J.F., Resende, M.G.: A biased random key genetic algorithm for 2D and 3D bin packing problems. International Journal of Production Economics **145**, 500–510 (2013)
8. Huang, W., Ye, T., Chen, D.: Bottom-Left Placement Theorem for Rectangle Packing. arXiv:1107.4463 [cs] (2011)
9. Hwang, H., Choi, B., Lee, G.: A genetic algorithm approach to an integrated problem of shelf space design and item allocation. Computers & Industrial Engineering **56**, 809–820 (2009)
10. Jakobs, S.: On genetic algorithms for the packing of polygons. European Journal of Operational Research **88**, 165–181 (1996)
11. Kierkosz, I., Luczak, M.: A hybrid evolutionary algorithm for the two-dimensional packing problem. Central European Journal of Operations Research **22**, 729–753 (2014)
12. Libralesso, L., Fontan, F.: An anytime tree search algorithm for the 2018 ROADEF/EURO challenge glass cutting problem. arXiv:2004.00963 [cs] (2020)
13. Liu, D., Teng, H.: An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. European Journal of Operational Research pp. 413–420 (1999)
14. Lodi, A., Monaci, M., Pietrobuoni, E.: Partial enumeration algorithms for Two-Dimensional Bin Packing Problem with guillotine constraints. Discrete Applied Mathematics pp. 40–47 (2017)
15. Merz, P., Freisleben, B.: Memetic Algorithms for the Traveling Salesman Problem. Complex Systems p. 50 (2001)
16. Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts - Towards Memetic Algorithms. Caltech Concurrent Computation Program (1989)
17. Moscato, P., Cotta, C.: A Gentle Introduction to Memetic Algorithms. Operations Research & Management Science **57**, 105–144 (2003)
18. Pisinger, D., Sigurd, M.: Using Decomposition Techniques and Constraint Programming for Solving the Two-Dimensional Bin-Packing Problem. INFORMS Journal on Computing **19**, 36–51 (2007)
19. Puchinger, J., Raidl, G.R.: Models and algorithms for three-stage two-dimensional bin packing. European Journal of Operational Research **183**, 1304–1327 (2007)
20. Ronald, S.: Duplicate genotypes in a genetic algorithm. IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360) pp. 793–798 (1998)