Solving a Real-World Glass Cutting Problem

Jakob Puchinger, Günther R. Raidl, and Gabriele Koller

Institute of Computer Graphics and Algorithms,
Vienna University of Technology, Vienna, Austria
{puchinger|raidl|koller}@ads.tuwien.ac.at

Abstract. We consider the problem of finding two-dimensional cutting patterns for glass sheets in order to produce rectangular elements requested by customers. The number of needed sheets, and therefore the waste, is to be minimized. The cutting facility requires three-staged guillotineable cutting patterns, and in addition, a plan for loading produced elements on transportation wagons. The availability of only three loading docks for wagons imposes additional constraints on the feasibility of cutting patterns. We describe a greedy first fit heuristic, two branch-and-bound based heuristics, and different variants of an evolutionary algorithm for the problem, and compare them on a set of real-world instances. The evolutionary algorithm is based on an order representation, specific recombination and mutation operators, and a decoding heuristic. In one variant, branch-and-bound is occasionally applied to locally optimize parts of a solution during decoding.

1 Introduction

In this paper we present methods developed for solving a cutting problem that appears in real-world glass manufacturing. Customer specified rectangular elements need to be cut out from raw stock sheets. Due to the mechanics of the cutting machine, only three-staged guillotine cuts are possible, and additional constraints regarding the order in which elements are cut, must be obeyed. The problem falls into the general category of two-dimensional bin packing (2BP) problems; according to the notation of Lodi et al. [10] it can be classified as 2BP|R|G with additional constraints.

Our problem is enhanced by constraints resulting from the characteristics of the production process. A robot located at the end of the glass-cutting facility puts finished elements on wagons located at three different loading docks. The elements are partitioned in logical groups according to customer orders, and each wagon may carry elements from the same logical group only. Since the wagons cannot be exchanged arbitrarily, elements from a maximum of three logical groups can be produced in an intertwined way. A solution to our problem must also contain a schedule for loading finished elements onto wagons.

A detailed problem description is given in Sec. 2. Related methods from literature for solving the classical two-dimensional bin packing problem are summarized in Sec. 3. Sections 4 and 5 present a new greedy heuristic and two branch

This work is supported by the Austrian Science Fund (FWF) under grant P16263-N04.

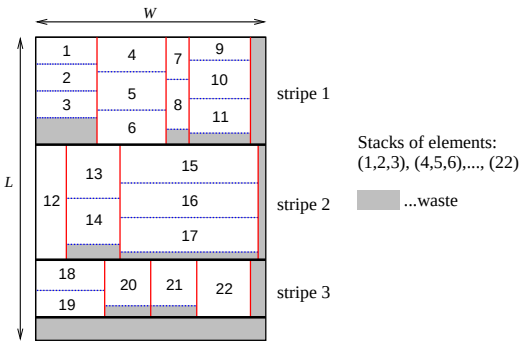

Fig. 1. An example for a three-stage cutting pattern in normal form.

and bound algorithms for our problem. Section 6 describes three variants of an evolutionary algorithm (EA). The different algorithms are experimentally evaluated and compared to each other and to a commercial solver in Sec. 7. In most cases, the EA yields the best results. More details on this work are given in [**?**].

2 Problem Description

In the *two-dimensional three-staged Bin Packing problem with Wagon Scheduling* (2BP-WS), we are given:

- Identical rectangular stock sheets of width $W > 0$, length $L > 0$, and thickness $T > 0$.
- A set of rectangular element types $\mathcal{E} = \{E_1, \ldots, E_m\}$ requested by customers; each element type E_i, $i = 1, \ldots, m$, is defined by the elements' width w_i ($0 < w_i \leq W$), length l_i ($0 < l_i \leq L$), a logical group identifier g_i indicating the customer to whom the elements are to be sent, and the number $n_i > 0$ of instances needed of this type. Let $n = \sum_{i=1}^{m} n_i$ be the total number of elements.
- A minimum fill level $B_{\min} > 0$ and a maximum fill level $B_{\max} \geq B_{\min}$ for the transportation wagons; the fill level of a wagon is given by the cumulated thickness of the elements loaded on it.

The goal is to find patterns for cutting out all instances of the given element types from the smallest possible number of stock sheets, thus, minimizing the waste, and a schedule for loading produced elements to wagons. Elements may be rotated by 90°. The production machine and loading robot impose the following restrictions on feasible cutting patterns.

Orthogonal guillotine cuts. Since glass sheets are cut by scratching and breaking them over a straight edge, only orthogonal guillotine cuts are possible, i.e. cuts parallel to a side of the piece being cut, going from one border straight to the opposite side.

Three stages. The production machine consists of three stages. The first stage only cuts a whole sheet horizontally into an arbitrary number of *stripes*. Stripes are further processed in the second stage where they are cut vertically into an arbitrary number of so-called *stacks*. The third stage performs, again, only horizontal cuts, producing the final elements (and waste) from the stacks. See Fig. 1 for an example of a three-stage cutting pattern.

Any feasible three-stage cutting pattern can always be reduced into its so-called *normal form* by moving each element to its uppermost and leftmost position, so that waste may only appear at the bottom of the stacks on the right of the rightmost stack in each stripe or at the bottom below the last stripe of each sheet. The pattern shown in Fig. 1 is in normal form. In the rest of the paper, we only consider cutting patterns in normal form, which reduces the general infinite search space of arbitrary three-stage cutting patterns to a finite number of possible solutions.

Wagon restrictions. For a given cutting pattern, elements are always produced by processing the stripes within each sheet from top to bottom, the stacks within each stripe from left to right, and the elements within a stack from top to bottom. The numbering of the elements in Fig. 1 denotes this order.

Finished elements are loaded on wagons in the same order as they are produced. Initially, three empty wagons are placed at three available loading docks. The following constraints apply to the loading of elements on wagons.

1. The first element put on a wagon initializes the wagon's logical group, i.e. it defines its destination address. Only elements of the same logical group may further be added.
2. A wagon is immediately closed and replaced by a new, empty one, if the fill level of a wagon reaches $B_{\max}$ or if all elements of its associated logical group have been produced and loaded.
3. A wagon may be closed and replaced earlier if its fill level is at least $B_{\min}$.

Only cutting patterns, for which a loading schedule fulfilling the above requirements exists, are considered feasible.

2.1 Objective Function and Continuous Lower Bound

The primary objective is to minimize the number $S(x)$ of sheets needed in a solution x. A refined objective function $F(x)$ considers the last sheet only partly:

$$F(x) = S(x) - \frac{L - c_l}{L}, \tag{1}$$

where c_l represents the position of the last stage-1 cut of the last sheet. This refined objective function allows for a better discrimination of the quality of solutions needing the same number of sheets: Cutting patterns leaving a larger unused stripe at the bottom of the last sheet are more promising. In practice, large unused final stripes may be useful later on and are therefore not necessarily considered as waste.

The continuous lower bound CLB for the objective function ($S(x)$ as well as $F(x)$) is given by the total area of all elements divided by a sheet's area:

$$CLB = \frac{\sum_{i=1}^{m} l_i \cdot w_i \cdot n_i}{L \cdot W} \tag{2}$$

3 Previous Approaches to Two-Dimensional Bin Packing

In this section we give a brief overview of known algorithms for solving the two-dimensional bin packing problem on which the new algorithms are based. For a more complete review we refer to the survey by Lodi et al. [9] and to the annotated bibliography by Dyckhoff et al. [5].

Most of the simple algorithms for 2BP are of greedy nature. The elements are placed one by one and never reconsidered again. There are one- and two-phase algorithms. In one-phase algorithms, the elements are directly placed in the bins. Two-phase algorithms first partition the elements into levels whose widths do not exceed W and whose total length is aimed to be minimized. In the second phase, levels are assigned to bins by heuristically solving a one-dimensional bin packing problem.

Berkey and Wang [1] described the classical Finite First Fit (FFF) heuristic, which is a greedy one-phase algorithm. The elements are sorted by decreasing lengths. The first element initializes the first level in the first bin and defines the level's length. Each following element is added to the first level to which it fits respecting the sheet's width W. If there is no such level, a new level is initialized in the first bin into which it fits. If there is no such bin, a new bin is initialized with the new level. Within a level, elements are never stacked.

For the general 2BP problem, Martello and Vigo [11] describe a branch-and-bound (B&B) algorithm which is based on a two-level branching-scheme: The elements are assigned to the bins by an outer decision tree. Possible packing patterns for the bins are heuristically generated. Only if the heuristic is not able to place all the elements of a bin, a pattern is tried to be found by an inner enumeration scheme. Morabito and Arenales [13] consider a staged and constrained guillotine cutting problem in which the utilization of only a single sheet is maximized. They describe a B&B algorithm whose decision tree corresponds to an AND/OR graph.

A broad overview of EAs for cutting and packing problems is given by Hopper [6]. Most of the EAs used to solve guillotineable 2BP, such as the genetic algorithm described by Kröger [8], are based on a slicing-tree representation and specific variation operators. Alternatively, an order-based encoding can be used indicating the order in which the elements are placed by some FFF-like decoding heuristic. Hwang et al. [7] describe such an approach and compare it to a slicing-tree representation. They conclude that the order-based method yields better results in several cases. Corno et al. [3] apply the order-based encoding successfully to an industrial cutting problem with specific real-time and pattern constraints. Another effective order-based approach is presented by Monaci [12] for two-dimensional strip packing.

Lodi et al. [10] describe a more general framework applicable to several variants of 2BP. Tabu search is used to assign the elements to the sheets, and cutting patterns for individual sheets are obtained by different inner heuristics.

Conclusions with respect to 2BP-WS. The algorithms presented so far cannot directly be applied to 2BP-WS because of the constraints imposed by the required wagon scheduling. Furthermore, several algorithms do not consider the restriction to three stages. The B&B algorithms, in particular, rely heavily on several symmetries of the general 2BP problem: Bins within a solution, levels within a bin, and elements within a level can be arbitrarily permuted. This is exploited by only considering levels and elements in specific orders. In case of 2BP-WS this simplification is not possible because the production order is a substantial part of the solution. Order-based EAs, however, do not principally rely on certain symmetries and can be adapted to our needs by modifying the decoding heuristic.

4 A Finite First Fit Heuristic for 2BP-WS (FFFWS)

The FFF heuristic from [1] has been modified and extended for 2BP-WS. In contrast to the classical FFF heuristic and due to the wagon scheduling requirements, FFFWS constructs a solution sheet by sheet and stripe by stripe in a greedy way; i.e. once a new sheet (stripe) is started, previous sheets (stripes) are never reconsidered.

Let $\Pi = (\Pi_1, \ldots, \Pi_m)$ be a list of references to the element types sorted according to decreasing lengths l_i, and let $|\Pi_1|, \ldots, |\Pi_m|$ be the corresponding numbers of not yet placed elements; initially $|\Pi_i| = n_i$.

The algorithm starts by placing a longest available element, i.e. an element of type Π_1, at the upper left position of the first sheet. This element is said to initialize the sheet and the first stripe, and it also defines this stripe's length $\hat{l}$. Whenever an element is placed, the corresponding counter $|\Pi_i|$ is decreased; if $|\Pi_i| = 0$, Π_i is removed from the list Π. The stripe is then iteratively filled from left to right with a stack of elements from the first type Π_i in list Π fulfilling the following conditions: (a) $l_{\Pi_i} \leq \hat{l}$; (b) w_i does not exceed the stripe's remaining width; (c) a permitted loading dock exists for elements of type Π_i with respect to wagon scheduling (see below). The number of stacked elements is $\min(|\Pi_i|, \lfloor \hat{l}/l_{\Pi_i} \rfloor)$. If no further element fits onto the current stripe, a new stripe is started with an element of the first type Π_i in list Π for which l_{Π_i} does not exceed the sheet's remaining length and a permitted loading dock is available. If no such element exists and Π is not empty, a stripe on a new sheet is started in the same way.

FFFWS tries to determine a permitted loading dock for an element by considering the element's logical group g_i and the current states of the wagons at the three loading docks. If g_i appears as logical group of one of the wagons at the loading docks, then this loading dock is a permitted target. Otherwise, we look for a loading dock whose wagon is either empty or has reached its minimum fill

level $B_{\min}$. In the latter case, the wagon is closed and replaced by a new one. If a permitted loading dock could be determined, its identifier is stored as loading target for the element as part of the solution. Wagon changes need not explicitly be stored in this kind of greedy solutions, since they can be implicitly derived.

The possible rotation of elements is considered by maintaining for each element type two references Π_i and Π'_i in Π with linked counters of not yet placed elements; hereby, Π'_i represents the rotated version of the element type.

In the worst-case, FFFWS has to process the complete list Π when looking for the next element to be placed. Therefore, the worst-case total time complexity of FFFWS is $O(n \cdot m)$.

5 Branch-and-Bound Algorithms for 2BP-WS

Two different B&B strategies were devised. Since symmetries cannot be exploited as in case of the classical 2BP, it seems to be impossible to solve 2BP-WS instances of practical size to provable optimality. Therefore, the following B&B strategies also contain heuristics and cannot guarantee optimal solutions.

BB-1. The first B&B variant follows the principle of the greedy two-phase algorithms for 2BP. In the first phase, BB-1 iteratively generates stripes of width W and length not exceeding L containing elements not yet placed on previous stripes. B&B is used to minimize the difference between the stripe's area and the total area of the elements on the stripe, i.e. the waste. Subproblems are created from partially filled stripes by considering any feasible extension by a single element or a stack of elements of the same type. A breadth-first enumeration strategy is applied. The waste of the currently best stripe is used as global upper bound. As local lower bound, the subproblem's waste reduced by the empty area to the right of the last stack is used. If during the construction of the stripe a wagon change involving also a change of the logical group occurs, we consider the stripe a *delimiter*.

In the second phase, stripes are arranged on sheets by an adapted first fit decreasing algorithm for one-dimensional bin packing [2]. Due to wagon scheduling, only the order of stripes in sections between delimiters may be permuted.

BB-2. In contrast to BB-1, BB-2 maximizes the utilized area of whole sheets instead of minimizing the waste of single stripes. The patterns for the sheets are generated one by one by B&B. New subproblems are created from partially filled sheets by considering any feasible extension by a single element or a stack of elements of the same type, if necessary on a new stripe. The FFFWS heuristic is applied to get an initial global lower bound for the utilized area. The local upper bound of a subproblem is calculated as the total area of already placed elements plus an upper bound for the area that might further be utilized. Additional elements may only be placed to the right of the last stack on the last stripe or below the last stripe. Again, a breadth-first enumeration strategy is applied.

6 Evolutionary Algorithms for 2BP-WS

The presented algorithms FFFWS, BB-1, and BB-2 yield respectable results in several cases. Nevertheless, independently optimized stripes or sheets, as in case of BB-1 and BB-2, do not guarantee an optimal overall solution. In order to approach the problem in a more global way, we apply evolutionary algorithms.

Representation and Decoding. Two variants of an order-based representation similar to those in [7,3] were chosen to encode candidate solutions.

Element type representation. Solutions are encoded as ordered vectors of references to element types $\Pi = (\Pi_1, \ldots, \Pi_m)$. The phenotypic solution is derived by applying a modified FFFWS heuristic as decoder. Instead of considering element types ordered by decreasing length, the order is given by the genotype. Furthermore, the decoding heuristic allows stacked elements to extend a stripe's length if the total waste of the stripe decreases.

Element representation. The element type representation has the disadvantage of placing elements of the same type always next to each other, if space permits. To allow more flexibility, we also consider an order-based representation in which a solution is encoded as permutation $\pi = (\pi_1, \ldots, \pi_n)$ of references to each individual element. This can significantly increase the size of the genotype and the solution space in general. The element type representation's decoding heuristic has been adapted accordingly. In particular, elements of the same type may only be stacked by the decoding heuristic if they appear directly next to each other in the encoded solution.

Rotation of elements is in both representations handled by maintaining linked second references as in FFFWS. For simplicity, we neglect here the resulting changes in the genotype's length.

Recombination and Mutation. In principle, any standard recombination and mutation operator designed for permutations is feasible. Regarding recombination, *order 3 crossover* (OX3) [4] turned out to be particularly well suited because it partially respects absolute positions and relative orders, which is a crucial condition for our problem.

Two mutation operators are used. *Reciprocal exchange* (RX) chooses two positions at random and swaps them. *Block exchange* (BX) swaps two randomly chosen non-overlapping blocks of length $\lceil 2^R \rceil$, with R being a random value in the interval $(0, \lfloor \mathrm{ld}\ \frac{m}{2} \rfloor]$ for the element type representation and $(0, \lfloor \mathrm{ld}\ \frac{n}{2} \rfloor]$ for the element representation. In this way, shorter blocks are chosen more likely, but large blocks up to half of the genotype's length are also possible.

Two additional operators were specifically designed for the element representation with the aim to encourage sequences of elements of the same type in order to exploit stacking more often.

Grouped order crossover (GOX) is an extension of the standard OX3 operator. As in OX3, two crossover points are randomly chosen. However, only positions inbetween elements of different types are allowed. The crossover region defined by the two crossover points is transmitted directly from the first parent

```
Algorithm Grouping Mutation (π);
j ← a random value ∈ {1, ..., n};   // starting position
p ← a random value ∈ [0, 1);   // element moving probability
left ← j − 1; right ← j + 1;
for i ← j − 1, ..., 1 do
  if type of element π_i = type of element π_j then
    with probability p do
      swap (π_i, π_left); left ← left − 1;
for i ← j + 1, ..., n do
  if type of element π_i = type of element π_j then
    with probability p do
      swap (π_i, π_right); right ← right + 1;
```

Fig. 2. Grouping mutation operator for the element representation.

to the offspring. All remaining positions are filled with the remaining elements of the first parent in the order given by the second parent.

Grouping mutation (GM) is used to form larger sequences of elements of the same type. Figure 2 shows the algorithm in detail. A position j of the permutation π is randomly chosen as starting position. π is scanned and each element of the same type as π_j is incorporated into the sequence around j with a probability chosen randomly from $[0, 1)$.

The computational effort of both operators, GOX and GM, is linear in the number of elements.

EA Variants The described operators were integrated in a standard steady-state EA. Initial solutions are created at random. In each iteration, a new candidate solution is created by always applying crossover and mutation. The new solution replaces the worst solution in the population if it is not identical to an already existing solution on the genotype level.

We consider the following three EA variants:

- *EAet* uses the element type representation.
- *EAe* uses the element representation.
- *EAebb* corresponds to EAe enhanced by phase 1 of BB-1: In the decoding, stripes are not just created by the greedy heuristic, but with a certain probability p_{bb} by applying B&B.

In case of EAe and EAebb the order of the elements in a decoded solution is reencoded into the genotype in a Lamarckian manner with probability p_{wb}.

7 Computational Results

31 real-world instances were provided by Soglatec GmbH and are available at `http://www.ads.tuwien.ac.at/pub/TestProblems/glasscut`. These instances have strongly varying characteristics: 1 to 31 logical groups, 1 to 78 element types, 4 to 651 elements in total, and continuous lower bounds *CLB* ranging from 0.34 to 23.90.

Table 1. Summarized results of selected algorithms; $\overline{CLB} = 6.010$; * denotes that the mutation operator was randomly chosen among RX, BX, and GM; the value printed after the mutation type represents p_{wb}; for EAebb the last value represents p_{bb}.

algorithm	$\overline{F}$	$\overline{\sigma_F}$	$\overline{F/CLB}$	$\bar{t}$[s]	best
XOPTS	7.195	n.a.	1.1967	n.a.	15
FFFWS	7.896	n.a.	1.3191	0.02	2
BB-1	8.280	n.a.	1.3213	0.07	6
BB-2	7.711	n.a.	1.2320	285.23	13
EAet OX3	7.132	0.032	1.1857	16.79	14
EAet OX3,RX	7.110	0.023	1.1833	13.88	18
EAe OX3,RX	7.122	0.040	1.1828	46.05	17
EAe OX3,RX,10%	7.089	0.032	1.1813	60.46	17
EAe GOX,RX	7.114	0.032	1.1821	47.54	17
EAe GOX,*	7.108	0.032	1.1815	46.57	17
EAe GOX,RX,10%	7.079	0.025	1.1803	47.62	18
EAe GOX,*,10%	7.080	0.026	1.1802	48.26	18
EAe GOX,*,100%	7.174	0.058	1.1868	37.45	15
EAebb GOX,*,10%,1%	7.079	0.027	1.1796	69.49	19
EAebb GOX,*,10%,10%	7.076	0.019	1.1791	267.18	21

We compare the greedy heuristic FFFWS, the B&B algorithms BB-1 and BB-2, the EA variants, and the commercial optimizer XOPTS of Albat & Wirsam GmbH. The new algorithms were implemented in C++ and tested on a Pentium 4 PC with 2.8 GHz. Results of XOPTS were provided by Soglatec GmbH and do not respect wagon scheduling. Thus, some of XOPTS' cutting patterns are actually infeasible for our specific 2BP-WS problem. Nevertheless, results from XOPTS provide an important basis for comparison, since 2BP-WS has not been addressed before.

In BB-2, due to memory and time restrictions, the enumeration for a sheet was aborted when the subproblem list exceeded 3 000 000 entries; the best cutting pattern so far was kept.

Strategy parameters of the EAs were determined by extensive preliminary experiments. A population size of 1 000 has been used, and each run was terminated after 50 000 iterations without improvement of the best solution. The number of mutations applied to each new candidate solution is chosen as a Poisson-distributed random variable with expected value 2.

Each EA variant has been tested with different combinations of the presented recombination and mutation operators. In some experiments more than one mutation operator was applied with equal probability; for each mutation the operator to be used was chosen at random. For each EA configuration, 30 independent runs per instance have been carried out.

Average results over all instances obtained by selected algorithm variants are listed in Table 1. Shown are average solution qualities ($\overline{F}$), corresponding average standard deviations ($\overline{\sigma}$) for the EAs, solution qualities relative to the continuous lower bounds $\overline{F/CLB}$, average total running times $\bar{t}$, and the number of instances for which the algorithm yielded the best solution on average.

It can be observed that average solution qualities do in general not differ very much. FFFWS is the fastest, but usually provides poor results. While BB-1 yields on average the worst results, the solutions of BB-2 are significantly better. However BB-2 also suffers from the greedy nature of optimizing each sheet independently; it is slow and highly memory consuming for larger instances; occasionally, the B&B optimization of some sheets had to be prematurely aborted. BB-2 yields sometimes better, but on average worse results than XOPTS.

In general, the EA variants obtained the best solutions and needed only moderate running times. As expected, EAet is significantly faster than EAe due to its smaller search space. EAebb is slower than EAe due to its B&B optimization of stripes. EAebb with GOX, the combination of all three mutation operators, Lamarckian write-back with probability $p_{\text{wb}} = 10\%$ and B&B optimization of stripes applied with a probability of $p_{\text{bb}} = 10\%$ performed best with respect to solution quality.

Table 2 provides more detailed results on each of the 31 instances for FFFWS, BB-1, BB-2 and a choice of EA configurations. For the EAs, columns F_{avg} list average solution values over 30 runs per instance, and columns F_{min} show solution values of the best runs.

It turns out that not a single EA configuration is consistently best for all tested instances. On average, however, the element representation yielded significantly better results than the element type representation on 9 instances, and in case of the element representation, GOX outperformed OX3. Lamarckian write-back with probability $p_{\text{wb}} = 10\%$ generally increases the quality of final solutions. Regarding mutation, the combined application of RX, BX, and GM, proved to be slightly more robust than only applying RX. The B&B optimization of stripes as done in EAebb increases the quality of solutions a bit further. A Wilcoxon rank test indicates a significant improvement between solutions from EAe and EAebb with $p_{\text{bb}} = 10\%$ (GOX, $p_{\text{wb}} = 10\%$, and combined mutation used in both cases) on an error level of less than 1% for 6 instances.

Two examples for cutting patterns obtained by FFFWS and EAebb for instance 12 are shown in Fig. 3.

8 Conclusion

We considered different heuristic algorithms for a real-world glass cutting problem, in which the loading of produced elements imposes difficult constraints on feasible cutting patterns. Therefore, approaches for 2BP such as XOPTS cannot be used directly. The proposed FFFWS heuristic is fast and well suited as decoder for an order-based EA. For the EA with element representation, GOX and GM were proposed to encourage sequences of elements of the same type in order to better exploit the possibility of stacking. Incorporating B&B in the decoding for occasionally locally optimizing stripes turned out to increase the solution quality in a few cases. Solutions obtained by the EA are highly satisfactory for practice.

Table 2. Detailed results of selected algorithms. Overall best (average) solution values are printed bold. For BB-2, a mark "a" in column F indicates that in one or more runs B&B had to be aborted for one or more sheets.

				CLB	XOPTS	FFFWS		BBHEU		BBALG			EAet OX3,RX				EAe GOX,*,10%				EAebb GOX,*,10%,10%			
inst.	groups	m	n		F	F	t	F	t	F		t	F_{avg}	σ	$F_{\min}$	t_{avg}	F_{avg}	σ	$F_{\min}$	t_{avg}	F_{avg}	σ	$F_{\min}$	t_{avg}
1	3	3	4	0.83	**0.98**	1.37	0.01	**0.98**	0.00	**0.98**	0.01		**0.98**	0.00	0.98	2.93	**0.98**	0.00	0.98	7.87	**0.98**	0.00	0.98	6.89
2	3	3	4	0.90	**0.95**	0.97	0.02	**0.95**	0.00	**0.95**	0.01		**0.95**	0.00	0.95	2.70	**0.95**	0.00	0.95	8.46	**0.95**	0.00	0.95	7.62
3	3	3	5	0.58	**0.74**	0.83	0.01	0.86	0.00	**0.74**	0.01		**0.74**	0.00	0.74	2.40	**0.74**	0.00	0.74	4.19	**0.74**	0.00	0.74	4.88
4	3	3	6	2.17	**2.88**	**2.88**	0.02	**2.88**	0.00	**2.88**	0.01		**2.88**	0.00	2.88	5.01	**2.88**	0.00	2.88	7.21	**2.88**	0.00	2.88	8.78
5	1	2	7	0.69	**0.80**	0.99	0.02	0.83	0.01	**0.80**	0.01		0.81	0.00	0.81	3.41	**0.80**	0.00	0.80	5.85	**0.80**	0.00	0.80	6.17
6	2	3	9	2.60	**2.87**	2.96	0.01	3.54	0.01	**2.87**	0.01		**2.87**	0.00	2.87	2.97	**2.87**	0.00	2.87	4.79	**2.87**	0.00	2.87	4.43
7	2	7	11	0.34	0.40	0.46	0.01	0.41	0.01	0.46	15.38	a	**0.38**	0.00	0.38	4.15	**0.38**	0.00	0.38	6.84	**0.38**	0.00	0.38	15.41
8	3	14	14	1.61	1.91	2.48	0.01	2.53	0.01	1.88	8.72		**1.85**	0.01	1.84	5.39	**1.85**	0.01	1.84	9.56	1.86	0.02	1.84	14.23
9	12	12	15	0.76	0.84	0.99	0.03	0.83	0.01	0.99	25.21	a	**0.83**	0.00	0.83	4.40	**0.83**	0.00	0.83	7.69	**0.83**	0.00	0.83	34.43
10	3	7	19	2.08	**2.44**	2.47	0.00	2.67	0.01	**2.44**	64.67	a	**2.44**	0.00	2.44	6.25	**2.44**	0.00	2.44	10.21	**2.44**	0.00	2.44	25.87
11	3	5	22	2.50	**5.56**	**5.56**	0.02	**5.56**	0.00	**5.56**	12.11	a	**5.56**	0.00	5.56	13.64	**5.56**	0.00	5.56	22.57	**5.56**	0.00	5.56	22.14
12	12	12	33	3.50	3.98	4.67	0.00	4.60	0.01	**3.94**	28.22		3.96	0.00	3.96	10.43	3.95	0.01	3.93	19.01	**3.94**	0.01	3.93	30.57
13	1	16	33	3.65	4.70	4.47	0.01	4.66	0.03	4.24	141.53	a	**4.17**	0.00	4.16	7.51	**4.17**	0.01	4.16	13.15	**4.17**	0.01	4.16	58.23
14	3	24	34	6.72	8.77	8.39	0.01	8.39	0.02	**7.66**	4.47		**7.66**	0.02	7.66	12.04	7.69	0.06	7.66	18.68	**7.66**	0.00	7.66	60.19
15	5	9	37	2.80	2.96	3.48	0.01	3.58	0.03	2.96	31.35		2.97	0.00	2.97	5.57	**2.95**	0.00	2.95	11.18	**2.95**	0.00	2.95	50.96
16	9	9	53	7.86	10.56	10.78	0.02	12.56	0.02	11.56	2.43		**10.24**	0.00	10.24	8.73	**10.24**	0.00	10.24	15.51	**10.24**	0.00	10.24	32.07
17	15	23	60	6.88	7.89	8.68	0.01	8.41	0.03	7.88	257.48		7.87	0.08	7.79	21.57	7.89	0.11	7.79	40.12	**7.80**	0.04	7.67	258.81
18	11	11	61	4.40	4.58	5.15	0.03	5.57	0.02	4.88	2689.97	a	4.64	0.01	4.57	8.80	4.60	0.02	4.56	23.92	**4.56**	0.01	4.55	70.26
19	7	9	69	7.63	**8.22**	8.94	0.01	9.54	0.02	8.54	16.66		8.39	0.01	8.39	9.69	8.25	0.03	8.22	28.83	8.27	0.01	8.22	51.44
20	1	1	70	5.56	6.97	8.68	0.02	**6.93**	0.00	**6.93**	0.03		**6.93**	0.00	6.93	8.24	**6.93**	0.00	6.93	26.48	**6.93**	0.00	6.93	43.35
21	29	31	72	5.28	**5.68**	5.96	0.02	6.66	0.14	5.89	209.92	a	**5.68**	0.03	5.63	29.18	5.75	0.05	5.67	45.60	5.73	0.05	5.64	830.37
22	31	31	72	5.28	**5.65**	5.96	0.00	6.66	0.15	5.89	215.79	a	5.67	0.02	5.64	27.41	5.75	0.05	5.67	43.25	5.75	0.04	5.68	1493.95
23	10	15	77	15.42	18.84	26.49	0.01	20.53	0.03	19.28	621.53	a	**18.53**	0.00	18.53	12.58	**18.53**	0.00	18.53	30.77	18.55	0.01	18.53	82.57
24	4	4	78	1.78	1.99	2.09	0.02	2.00	0.02	2.00	44.96	a	**1.98**	0.00	1.98	7.97	**1.98**	0.00	1.98	19.78	**1.98**	0.00	1.98	106.90
25	2	2	91	4.98	**5.40**	5.89	0.02	5.53	0.01	**5.40**	2.88		5.66	0.00	5.66	6.57	5.49	0.00	5.49	28.73	5.48	0.01	5.45	33.32
26	15	78	140	23.90	30.87	32.69	0.02	38.70	0.95	36.56	127.96		30.09	0.47	29.50	78.44	29.82	0.35	29.19	131.76	**29.80**	0.30	29.45	3484.31
27	3	4	147	8.00	**8.89**	9.16	0.01	9.60	0.03	**8.89**	382.09	a	9.15	0.00	9.15	12.81	8.92	0.04	8.89	69.13	**8.89**	0.00	8.89	78.42
28	11	45	149	22.85	**28.62**	32.62	0.02	37.62	0.38	36.62	979.34	a	28.63	0.02	28.62	29.53	28.63	0.02	28.62	66.87	28.63	0.02	28.62	315.66
29	6	13	151	3.53	3.78	4.00	0.02	4.19	0.05	3.75	398.56	a	**3.70**	0.01	3.68	17.03	3.71	0.01	3.68	63.18	**3.70**	0.01	3.66	178.21
30	8	15	151	9.68	**10.43**	10.93	0.03	12.59	0.03	11.34	387.25	a	10.62	0.02	10.58	23.00	10.49	0.04	10.44	103.95	10.49	0.02	10.47	354.93
31	7	7	651	21.54	23.88	23.79	0.05	26.32	0.14	24.28	2173.64	a	23.56	0.00	23.56	39.88	**23.44**	0.01	23.44	600.92	23.56	0.01	23.53	796.30

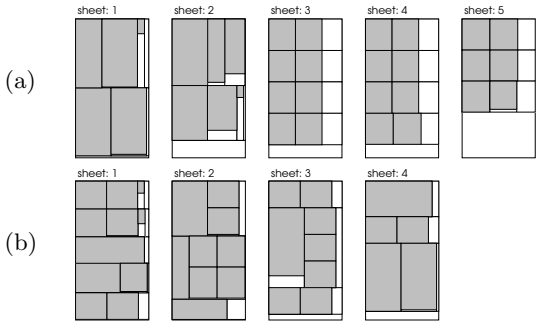

Fig. 3. Cutting patterns for instance 12 obtained by FFFWS (a) and EAebb with GOX, combined RX/BX/GM mutation, $p_{\text{bb}} = 10\%$, and $p_{\text{wb}} = 10\%$.

References

1. J. O. Berkey and P. Y. Wang. Two-dimensional finite bin packing algorithms. *Journal of the Operational Research Society*, 38:423–429, 1987.
2. E. Coffman, Jr., M. Garey, and D. Johnson. Approximation algorithms for bin packing: A survey. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*, pages 46–93. PWS Publishing, Boston, 1996.
3. F. Corno, P. Prinetto, M. Rebaudengo, M. S. Reorda, and S. Bisotto. Optimizing Area Loss in Flat Glass Cutting. In *IEEE Int. Conf. on Genetic ALgorithms in Engineering Systems: Innovations and Applications*, pages 450–455, 1997.
4. L. Davis, editor. *A Handbook Of Genetic Algorithms.* Int. Thomson Computer Press, 1991.
5. H. Dyckhoff, G. Scheithauer, and J. Terno. Cutting and packing: An annotated bibliography. In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 393–412. Wiley, 1997.
6. E. Hopper. *Two-Dimensional Packing Utilising Evolutionary Algorithms and Other Meta-Heuristic Methods.* PhD thesis, University of Wales, Cardiff, U.K., 2000.
7. S.-M. Hwang, C.-Y. Kao, and J.-T. Horng. On solving rectangle bin packing problems using GAs. In *Proceedings of the 1994 IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 1583–1590, 1997.
8. B. Kröger. Guillotineable bin packing: A genetic approach. *European Journal of Operational Research*, 84:545–661, 1995.
9. A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141:241–252, 2002.
10. A. Lodi, S. Martello, and D. Vigo. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11:345–357, 1999.
11. S. Martello and D. Vigo. Exact solutions of the two-dimensional finite bin packing problem. *Management Science*, 44:388–399, 1998.
12. M. Monaci. *Algorithms for Packing and Scheduling Problems.* PhD thesis, University of Bologna, Italy, 2002.
13. R. Morabito and M. N. Arenales. Staged and constrained two-dimensional guillotine cutting problems: An AND/OR-graph approach. *European Journal of Operational Research*, 94(3):548–560, 1996.