CEJOR
DOI 10.1007/s10100-013-0300-0

ORIGINAL PAPER

A hybrid evolutionary algorithm for the two-dimensional packing problem

Igor Kierkosz · Maciej Luczak

Abstract This paper presents a hybrid evolutionary algorithm for the two-dimensional non-guillotine packing problem. The problem consists of packing many rectangular pieces into a single rectangular sheet in order to maximize the total area of the pieces packed. Moreover, there is a constraint on the maximum number of times that a piece may be used in a packing pattern. The set of packing patterns is processed by an evolutionary algorithm. Three mutation operators and two types of quality functions are used in the algorithm. The best solution obtained by the evolutionary algorithm is used as the initial solution in a tree search improvement procedure. This approach is tested on a set of benchmark problems taken from the literature and compared with the results published by other authors.

Keywords Packing · Cutting · Evolutionary computations · Metaheuristics

1 Introduction

The problem discussed in this paper consists of packing rectangular pieces into a large rectangular sheet, or equivalently, cutting small rectangular pieces from a large rectangular plate. It is assumed that we are dealing with a non-guillotine cutting, i.e. one in which (unlike in a guillotine cutting) subsequent cuttings do not need to proceed across the entire length of the cut material. The maximum number of times that each type of piece can be cut from the plate is also fixed. The objective is to maximize the

I. Kierkosz (✉) · M. Luczak
Department of Civil and Environmental Engineering, Koszalin University of Technology, Sniadeckich 2, 75-453 Koszalin, Poland
e-mail: igor.kierkosz@tu.koszalin.pl

M. Luczak
e-mail: mluczak@wbiis.tu.koszalin.pl

Published online: 01 May 2013

total area of the rectangles cut, or equivalently, to minimize the unused area of the stock sheet (minimize trim loss).

Cutting and packing problems and similar problems occur in many practical applications, for example the cutting of steel or glass sheet into smaller elements, or cutting of rectangular plates in the process of furniture manufacturing. Cut materials may also include paper, fabric, foam polystyrene, etc. Equally, one can also consider a two-dimensional bin packing problem, VLSI systems design or the problem of arranging announcements and advertisements in newspapers. In various applications, depending on the specificity of the problem, one may be dealing with different constraints (for example Furian and Vössner 2011) and various criteria of solution assessment.

This diversity of cutting and packing (C&P) problems has led to attempts to classify them. The first such attempt was made by Dyckhoff (1990), who classified C&P problems by four criteria. In 2007, Wäscher et al. (2007) suggested a new typology. According to this typology, the problem under consideration in this paper may be classified as a two-dimensional single large object placement problem (2SLOPP) or two-dimensional single knapsack problem (2SKP). However, if the typology of Dyckhoff is adopted, we are dealing here with problems of 2/B/O/R or 2/B/O/M type (2—geometric dimension of the small items and large objects, B—all large objects are to be used and a selection of small items is to be assigned to large objects, O—one large object, R—many items of relatively few different figures, M—many items of many different figures).

Starting from the 1950s, various approaches to solving C&P problems have appeared in the literature. A review of the methods used can be found, for example, in Dyckhoff (1990), Hässler and Sweeney (1991), Sweeney and Paternoster (1992), Dowsland and Dowsland (1992), and Lodi et al. (2002).

It should be noted that the majority of works concern guillotine cutting. Regarding non-guillotine cutting, approximate methods clearly prevail. There are relatively few exact algorithms that solve this problem. The first person to propose such an algorithm was Beasley (1985). He applied the branch and bound method, where the upper bound was derived from the Lagrangean relaxation of a cutting problem formulated as a zero-one integer programming problem. Descriptions of other exact methods can be found in Scheithauer and Terno (1993), Hadjiconstantinou and Christofides (1995), Fekete and Schepers (1997), Boschetti et al. (2002), Caprara and Monaci (2004), and Clautiaux et al. (2007).

An interesting method is proposed in Arenales and Morabito (1995). In their approach, cutting patterns are presented in the form of an AND/OR graph which is searched by means of the branch and bound method. In order to reduce the size of the search tree, they employ some heuristic observations and confine themselves to non-guillotine cuttings created as a result of the combination of guillotine and simple non-guillotine cuts.

Many approximate methods developed for the non-guillotine cutting problem are based on the heuristic algorithm BL (bottom-left algorithm; see Baker et al. 1980) of placing rectangular elements on a rectangular plate, or on any of its modifications (for example Lai and Chan 1997a,b; Liu and Teng 1999; Leung et al. 2001). In this algorithm, starting from the upper right corner of the output plate, successive (randomly generated) elements are shifted down as far as possible, and then left. This "stepped"

movement for each element ends when a stable position has been achieved. An example modification of this procedure (BLF-algorithm; see Chazelle 1983) consists in the additional completion of the gaps created when elements are placed using the BL method. As shown in Liu and Teng (1999), not all cutting patterns can be achieved with the application of the BL procedure; however it does enable reduction of the number of generated cuttings.

Algorithms based on metaheuristics constitute a large group within the approximate methods. For instance the following may be mentioned: genetic and evolutionary algorithms (Jakobs 1996; Hadjiconstantinou and Iori 2007; Lai and Chan 1997a; Leung et al. 2001; Beasley 2004; Burke et al. 2004; Gonçalves 2007; Gonçalves and Resende 2011), a simulated annealing algorithm (Dowsland 1993; Lai and Chan 1997b; Leung et al. 2001, 2003, 2012; Burke et al. 2004), and a Tabu search algorithm (Alvarez-Valdes et al. 2007; Wei et al. 2011). A comparison of the efficiency of heuristic and metaheuristic algorithms can be found in Hopper and Turton (2001).

It should be emphasized that in practical applications one often faces a more general problem, which consists in minimizing the number of rectangular bins used to pack a set of smaller rectangles or in minimizing the number of plates which need to be cut to execute a particular production order (for example Mack and Bortfeldt 2012).

2 Problem description

Because cutting stock problems and packing problems are basically the same problems, in this paper the terms cutting and packing will be used interchangeably.

The two-dimensional non-guillotine packing problem addressed in this paper is the problem of cutting from a single rectangular stock sheet (plate) $A = (W, H)$, of width W and height H, smaller rectangular pieces p_i with dimensions (w_i, h_i), $i = 1, \ldots, m$. We call each possible way of cutting a plate a *cutting pattern.*

Let A be located in the Cartesian coordinate system Oxy, with its bottom left corner placed in position $O(0, 0)$ and with its edges parallel to the x-axis and y-axis.

The following assumptions are made:

- the cuts are made with their edges parallel to the edges of the stock plate (orthogonal cuts);
- each piece p_i has an associated value v_i equal to its area ($v_i = w_i h_i$, for each $i = 1, \ldots, m$);
- in a cutting pattern there may be used no more than b_i replicates of each piece p_i for all $i = 1, 2, \ldots, m$ (constrained problem)—the total number of items available for cutting is denoted by $M (M = \sum_{i=1}^{m} b_i)$;
- the pieces have fixed orientation, i.e. a piece of width w and height h is not the same as a piece of width h and height w;
- the cuts are infinitely thin;
- all input parameters are integers.

In cutting and packing problems the objective is usually to maximize the total value of the pieces cut:

$$\max \rightarrow V = \sum_{i=1}^{m} a_i v_i$$

where a_i is the number of times the rectangle p_i appears in a given cutting pattern ($a_i \leq b_i$ for each $i = 1, 2, \ldots, m$). Because we are assuming that each rectangle p_i has an associated value equal to its area, the objective is to maximize the total area of the rectangles cut (called here the *filling rate*):

$$\max \rightarrow g = \sum_{i=1}^{m} a_i w_i h_i$$

which expresses the filling degree of the stock sheet. This formulation is equivalent to minimizing the unused area of the stock rectangle (trim loss).

3 Method description

This paper presents a new algorithm for solving the cutting problem described above. The main part of our approach is the evolutionary algorithm. Its most characteristic features include natural coding of individuals in the population, and application of a heuristic algorithm to generate the initial population and in mutations. The algorithm uses two types of quality functions and a division into parts of the larger cutting problem. The division into parts occurs depending on the problem's degree of complexity, which is estimated using a special procedure. A relevant orientation change (rotation) of the parts obtained as a result of the plate division in some cutting problems is also a characteristic feature. In the evolutionary algorithm three mutation operators were used, although no crossover operators are defined. Cuttings obtained through the evolutionary algorithm are subject to post-optimization with the application of the aforementioned heuristic algorithm.

The scheme of the main algorithm is as follows:

```
01  procedure Main algorithm
02      Estimation of the size of the search tree;
03      Division into parts (for large plates only);
04      Rotation of the plate or the parts (if necessary);
05      Evolutionary algorithm;
06      Improvement of the final solution;
07  end.
```

3.1 Placement algorithm (PA)

The algorithm described below will be frequently used in this paper. It is a tree search improvement procedure where elements are placed on a plate similarly to the placement in the BL (bottom-left) strategy. It makes it possible to cut off branches of the search tree where the trim loss (the sum of inserted complements), treated as the cost of a particular partial solution, is not less than the trim loss of the currently best solution found.

Global variables:

`Elements`—list of elements (rectangles) for cutting;
`MinTrimLoss`—initial minimal trim loss;
`BestCuttingPattern`—solution variable.

```
01  procedure PA(cuttingPattern, trimLoss)
02     if trimLoss >= MinTrimLoss then Exit;
03     if cuttingPattern fills the entire plate then
04        BestCuttingPattern := cuttingPattern;
05        MinTrimLoss := trimLoss;
06        Exit;
07     end;
08     placingPoint := the current placing point of the cutting pattern;
09     anyPlaced := false;
10     foreach element in Elements do
11        if element can be placed at placingPoint then
12           anyPlaced := true;
13           newCuttingPattern := cuttingPattern plus element;
14           newTrimLoss := trimLoss;
15           PA(newCuttingPattern, newTrimLoss);
16        end;
17     end;
18     if not anyPlaced then
19        complement := the empty rectangle completing the cutting at the current placing point;
20        newCuttingPattern := cuttingPattern plus complement;
21        newTrimLoss := trimLoss + area of complement;
22        PA(newCuttingPattern, newTrimLoss);
23     end;
24  end.
```

Comments on the pseudocode:

`(02)` Branches of the search tree where the trim loss is not smaller than for the current best solution are cut off.
`(08)` The current placing point is found by minimization of the coordinate y and then (with the already determined value of the coordinate y) the coordinate x for the unused area of the plate.
`(11)` An element (rectangle) can be placed at the current placing point if its intersection with elements of the current cutting (excluding the edges of the rectangles) is empty and the constraint on the number of elements of the given type is not exceeded.
`(19)` The rectangle completing the cutting at the current placing point is a rectangle with maximum width which can be fitted to the minimum height of the neighboring ones (the shadow rectangle in Fig. 1). A complement is placed only if no element can be placed at the current placing point.

Procedure initiation:

`Elements` := *current list of elements for cutting*;
`MinTrimLoss` := *area of the entire plate or estimated minimal trim loss*;
`initialCuttingPattern` := *the empty cutting pattern without any elements or complements*;
`PA(initialCuttingPattern, 0);`

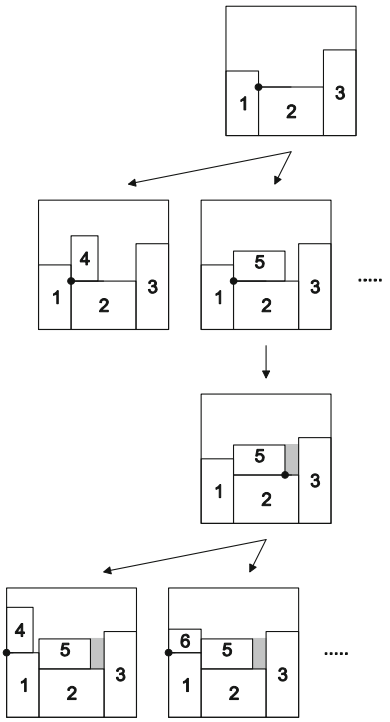

Fig. 1 Illustration of the PA algorithm

As a result of procedure PA, the variable `BestCuttingPattern` includes the cutting pattern with minimal trim loss found smaller than the initial value of the variable `MinTrimLoss`.

The described cutting algorithm is not an exact algorithm. There are cuttings which cannot be obtained by the application of this method. This results from the fact that from the set of all possible placing points which would need to be taken into account when placing a particular element on the plate, only that one is always selected (while rejecting other possibilities) which progressed most to the left among the lowest located points of the plate area where no elements have yet been placed. During experiments with the described tree search improvement procedure it was found that sometimes, before the initiation of the PA algorithm, it is sufficient to make a rotation by 90° of both the plate and all elements to achieve better solutions in the case of certain testing problems than those achieved in the standard manner. Such rotation is in reality equivalent to changing the order of minimization of the coordinates of the placing point for the version of the algorithm without a rotation: first the coordinate x is minimized, and then (with the determined value of the coordinate x) the coordinate y. This operation does not in any way change the adopted assumptions—after obtaining a solution it is enough to rotate it to return to the assumed sizes of the plate and elements. Therefore it was decided that in the case of problems of small or moderate size our evolutionary algorithm would be initiated twice: once in the version without

the rotation, and once with the rotation of elements and the plate. As the results of the computational experiments demonstrate, such an approach is justified.

This algorithm is used to improve the filling rate of the cutting after the evolution process and as a local optimization procedure in one of the mutation operators. It will be elaborated on later in the paper.

If in the above algorithm, instead of proceeding across all possible elements in a given placing point, one of the elements is selected at random (or an empty rectangular element if none of the elements fits), a random version of this method will be applied. In comparison with a simple BL algorithm, in the generated cutting patterns no gaps are formed where any more elements could fit. This random version of the algorithm is applied to generate the initial population of the evolutionary algorithm and in the mutation operators of this algorithm. It will be elaborated on later in the paper.

3.2 Estimation of the size of test problems and division into parts

3.2.1 Estimation of the size of the search tree

The previously described PA algorithm can be also employed for estimation of the number of cuttings that can be obtained for a specific problem. In the estimation procedure for each placing point a set of all elements which can be positioned on the plate and placed at a given point is determined. The decision as to which element of this set will be used in the cutting is made randomly, and the probability of selecting each element is the same.

Let us assume that at a given ith placing point, k_i elements (where $1 \leq k_i \leq M$) can be placed. Then the size of the tree search (more precisely, the number of tree nodes other than the root) can be evaluated by means of the following calculation:

$$k_1 + k_1k_2 + k_1k_2k_3 + \cdots + k_1k_2 \ldots k_i + \cdots$$

Since the achieved value depends on the results of the drawing of elements which will be located at subsequent placing points, the estimation procedure is generated many times (in our case 10,000 times) and then an average of the obtained estimates is determined. This average is adopted as the final estimation of the number of all solutions of a given problem possible to be achieved with the use of the PA algorithm.

3.2.2 Division into parts of large cutting problems

Plates in large cutting problems are divided into parts and the evolutionary algorithm is applied to each of them. The elements which were not yet used in cuttings of previous parts are applied for cutting each subsequent part. The setting of parameters of the evolutionary algorithm (in particular the stop condition) differs slightly depending whether it is the last part of the cutting or one of the previous parts. Exact settings of parameters are described in Sect. 4.1. The division into parts is performed in such a manner as to be most natural. If the largest element (in terms of width or height) determines a natural division, i.e. this size is more or less one half (in the case of

division into two parts) or one third (in the case of division into three parts) of the plate size, this is how the plate is divided. In other cases the plate is divided into two or three equal parts. The obtained pieces of the plate are set vertically, that is, first a section of the plate is rotated along with all elements for cutting, if necessary, and then the algorithm for seeking the best cutting is performed. The final cutting is again compiled with cuttings of relevant parts (with relevant rotations if necessary).

3.3 Evolutionary algorithm

3.3.1 Main loop

The scheme of the evolutionary algorithm is as follows:

```
01  procedure Evolutionary algorithm
02      Initial population;
03      repeat
04          Selection (fitness function);
05          Reproduction (mutations);
06          Best in population (valuation function);
07      until Stop condition
08  end.
```

3.3.2 Coding of individuals

No special coding of cutting patterns is used in the algorithm. Each individual (cutting) has a natural geometric interpretation as a sequence of rectangles (both elements and empty pieces) placed within the rectangle of the plate subject to cutting.

3.3.3 Selection of individuals

In the evolutionary algorithm two quality functions of individuals are applied. The *fitness function* is used for the selection of individuals for the process of reproduction (mutation). In the selection process, the roulette method with linear scaling is applied (Michalewicz 1992). The *valuation function* is used for selection of the best individual in a given population, and it is this individual that is remembered as the best found in a particular generation. This allows the selection process of individuals for reproduction to be separated from the process of selecting the best individual according to the criteria of our choice. Individuals most suitable for reproduction, most promising in terms of improvement (fitness function), do not need to be the individuals with the best assessment (valuation function). However, the ultimate selection of the best individual in a generation is performed by means of the valuation function which reflects our criteria for the best individual.

Valuation function The valuation function of an individual is calculated by

$$\texttt{valuation} = \texttt{fillingRate} - \texttt{parE} * \texttt{numOfElements},$$

where:

`fillingRate` is the filling rate of the individual;
`numOfElements` is the number of elements in the individual;
`parE` is a parameter determining how much the number of elements of the individual influences the value of the valuation function.

The main component of the valuation function is the filling rate of a particular individual, i.e. the filling rate of the cut plate. However, attention is also paid to the number of elements of which such individual is composed. Out of two cuttings with the same filling rate, the one which is composed of a smaller number of elements is better. With a small value of `parE` (as used in the present work), `numOfElements` distinguishes only individuals with the same filling rate.

This is of particular importance in those problems where the plate is divided into parts. If the cutting of the first part of the plate contains fewer elements, it is more likely that the remaining larger number of elements will be able to fit better (one has more choice) in the next part of the cutting. In one-part cuttings, or in the last parts of multipart cuttings, it is of little relevance—the result will simply be an individual with the smallest number of elements among the cuttings with the largest filling rate.

Fitness function The fitness function value of an individual is calculated by

$$\texttt{fitness} = \texttt{fillingRate} + \texttt{parD} * \texttt{W} * \texttt{emptySpaceVertPos} - \texttt{parE} * \texttt{numOfElements};$$

where:

`emptySpaceVertPos` is the minimal coordinate y of placing points of empty pieces or the height of the entire plate if the cutting pattern of the individual includes no empty rectangles;
`parD` is a parameter determining how much the structure of empty pieces influences the value of the fitness function.

The fitness function, apart from the filling rate and the number of elements of which the cutting consists, also takes into account the structure of the unfilled places of the cutting, the so-called empty pieces. The higher the empty pieces are located in the cutting, the larger is the value of the function. Two cuttings with the same value of the valuation function may differ in terms of the fitness function (Fig. 2). It may happen that one individual is better than the other in terms of the fitness function although it is worse in terms of the valuation function. Application of such a fitness function results in the movement of empty pieces in the evolution process up the cutting until (if possible) a cutting with full filling rate is achieved. The coefficient `parD` serves to indicate what percentage of the surface of the entire plate may be added to the value of the fitness function if there are no empty pieces in the cutting.

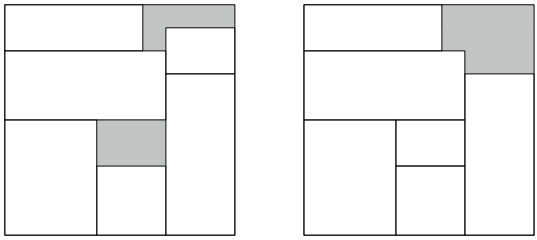

Fig. 2 Two cutting patterns with the same value of the valuation function (the same filling rate and number of elements) but different values of the fitness function (different structure of empty pieces)

3.3.4 Mutations

In each generation of the evolutionary algorithm, after performing the selection, the reproduction process takes place. In this process three mutations are used. Each of them occurs with a fixed, predetermined probability.

M1 mutation This mutation consists in the random trimming of an individual (removal of some of the recent cutting elements) and then random completion. The maximum number of cutting elements which may be removed depends on the current number of the generation of the evolutionary algorithm. This value will be marked as `maxNumOfElements` and it will be calculated by:

$$\texttt{maxNumOfElements} = \left\lceil \frac{\texttt{numOfGenerations} - \texttt{generationNum}}{\texttt{numOfGenerations}} * \texttt{numOfElements} \right\rceil,$$

where:

`numOfGenerations` is the total number of generations (iterations) of the evolutionary algorithm;
`generationNum` is the number of the current generation; $\lceil \cdot \rceil$ denotes rounding up to the next integer.

Then an integer from the range from 0 to `maxNumOfElements` is randomized, and such number of recent cutting elements is removed. Now the completion process takes place. In the empty places of the cutting, random elements (or empty rectangles) are inserted, in accordance with single run of the random version of the PA algorithm. Thus the mutation cuts off and completes the cutting at random, and the further in the evolutionary process one is (the higher the number of the generation), the smaller number of final cutting elements stand a chance of being trimmed.

M2 mutation This mutation acts similarly to the M1 mutation; however between the cutting-off phase and the completion phase there is an additional operation, the so-called local optimization. In this phase the PA algorithm is produced, but only for a certain established (`M2NumOfSteps`) number of steps forward in the search tree. As a result, at this point in the cutting, an optimal (in terms of the PA algorithm) placement of cutting elements occurs.

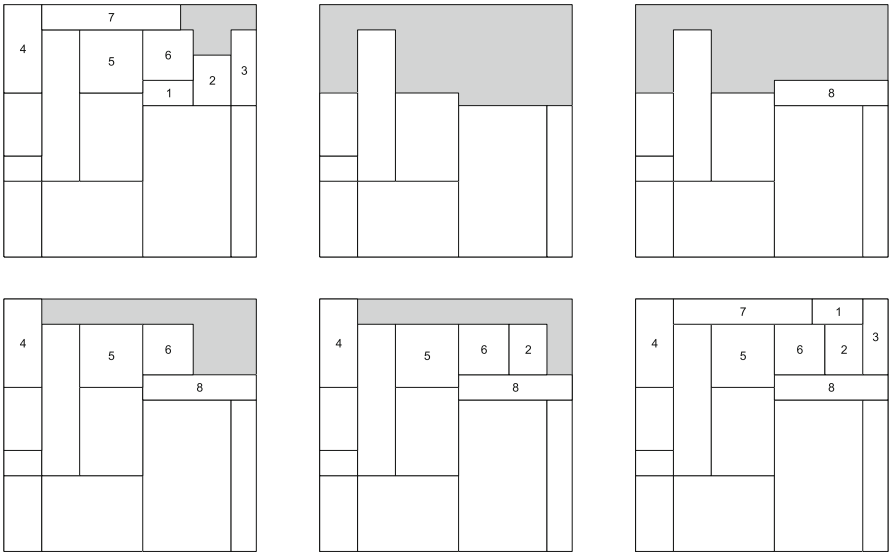

Fig. 3 M3 mutation mechanism (test problem HT1)

M3 mutation This mutation also attempts to optimize locally a selected cutting. The procedure below is repeated until a better individual (in terms of the fitness function) is achieved than the one subject to the mutation, or until all possibilities are exhausted:

1. Trim an individual at random (however, unlike in the previous mutations, always `maxNumOfElements = numOfElements);`
2. Remember the elements (and their sequence) which were removed;
3. At the current placing point find all the elements which can be fitted to the width or height of the neighboring ones in the trimmed cutting;
4. If such elements exist, choose one of them at random and place at the current placing point;
5. From the remembered cut-out elements, remove all those which have a common part (excluding the edges) with the inserted element;
6. Attempt to add the remaining remembered elements, and if one of them does not fit, insert a random fitting one (or an empty piece if none fits);
7. If there is still an empty place, complete the rest of the elements randomly, as in the previous mutations.

The diagram below (Fig. 3) shows an example trial of the operation of M3 mutation. From an output cutting the elements from 1 to 7 are removed (and remembered). Then, at the current placing point, a new element 8 is added, which fits to the width of the remaining cutting elements. From the sequence of the removed and remembered elements, those that have a common part with the added element 8 are removed, these being the elements 1, 2, 3. The remaining elements are attempted to be added without changing their order. In this case the following elements are added: 4, 5, 6. The subsequent element 7 cannot be added, thus a random element 2 is added and then the remaining empty places continue to be completed (at random). In the example this

led to the total completion of the cutting. As an individual better than the initial one, it is selected as a result of the M3 mutation.

3.3.5 Stop condition

The evolutionary algorithm is terminated after one of three conditions:

1. The last generation is exceeded, i.e. the algorithm performed the maximal number of iterations `numOfGenerations`;
2. The number of different individuals in the population is smaller than the value of the parameter `stopNumOfUnequalIndividuals`. Equal individuals are those with geometrically identical cutting patterns. If the diversification of populations decreases and there are many of the same individuals, the algorithm is terminated;
3. The filling rate of the best individual is greater than (or equal to) the value of the parameter `stopFillingRate`, and in a number of iterations no better individual is found:

$$\texttt{generationNum} >= \texttt{parP} * \texttt{bestGenerationNum};$$

$\texttt{bestGenerationNum}$ = *the generation number with the best current solution*.

3.4 Improvement of the final solution

As all genetic operators applied in the evolutionary algorithm are compliant with the heuristic algorithm (PA), i.e. the arrangement of elements of each cutting could be achieved through the algorithm (PA), one can use it to improve the final solution. Having obtained the best cutting in the evolution process, this cutting is used as a starting configuration of the algorithm (PA). It is run for a determined amount of time which (sometimes) makes it possible to increase the filling rate of the cutting. In the case of multi-part cuttings, this procedure is used only for the last part. Its application for the earlier parts could disturb the proper structure of the cutting achieved due to the operation of the fitness function. It could happen that a minimal increase of the filling rate occurs with a simultaneous large increase in the number of elements used in the cutting, which would be disadvantageous for the global solution.

4 Computational experiments

4.1 Algorithm setup

- Maximal number of generations (iterations): `numOfGenerations` = 1,000.
- Number of individuals in the population: `numOfIndividuals` = 1,000.
- Stop conditions: `stopNumOfUnequalIndividuals` = 500. Other parameters depend on the part of the cutting (if it is divided): for the first part: `stopFillingRate` = *entire filling rate*, `parP` = 1.5; for the last part: `stopFillingRate` = *entire filling rate*, `parP` = 0 (the algorithm is terminated immediately when the entire filling rate is reached).

- The scaling factor of the roulette method in the selection process is equal to 1.5.
- Probabilities of mutations are the same and equal to 1/3.
- The number of steps in mutation M2 (`M2NumOfSteps`) is equal to 2 or 3. For problems without division into parts: `M2NumOfSteps` = 3; for problems with division: `M2NumOfSteps` = 2 or 3.
- In valuation and fitness functions: `parD` = 0.1; `parE` = 0.001.
- The time of the post-optimization is predetermined as 60 s (provided the optimal solution was not found by the evolutionary algorithm).

4.2 Experimental conditions

For each test problem, 28 runs of the algorithm for various settings of the random generator were performed. The results always specify the best and the average value. The times of individual runs were also registered. The results specify the average time from all 28 runs. To the times of the runs of the evolutionary algorithm the time (60 s) of the post-optimization is added (provided it was performed).

A set of programs in the C# .net 3.0 language of the server/computational clients was created which enables distributed computing. This was installed on a set of 28 computers with 2.0 GHz processors connected with the local network. On each computer, one computational client was activated, which made it possible to perform parallel computations of the aforementioned 28 algorithm runs.

4.3 Description of test problems

The algorithm described in the paper was tested on two sets of test problems. The first set was composed of the following problems:

- 21 problems from Hopper and Turton (2001): HT1–HT21;
- 3 problems from Lai and Chan (1997a): LC1–LC3;
- 5 problems from Jakobs (1996): J1–J5;
- 2 problems from Leung et al. (2003): LYT1, LYT2.

In these problems the value of each element equals its area and, additionally, they have been designed in such a manner as to ensure that the optimal solution is a cutting with a zero trim loss. Thus for each of these problems the optimal value of the objective function adopted by us equals the area of the plate. It should be explained here that in the case of two problems in this set (LC2 and LYT2) from Leung et al. (2003) the dimensions of some elements were specified wrongly. In the problem LC2, the element 10 should have the size of 100×70, and in the problem LYT2 the correct dimensions of element 12 and 20 are 16×14 and 32×8, respectively. Since in the literature both variants of these two problems can be found, calculations were performed both for the correct version and for the version with erroneous data.

The problems in the first set can be considered to be problems of moderate and large (computational) complexity. The complexity of these problems was evaluated using the procedure described in Sect. 3.2.1. Table 1 specifies the estimated dimensions of the search tree (the number of tree nodes) and basic information about the test problems.

Table 1 Description of the first set of problems (corrected version in brackets)

Test problem	Container (L, W)	m	M	Estimated test problem size
HT1 (1-1)	(20, 20)	16	16	5.61×10^{9}
HT2 (1-2)	(20, 20)	17	17	6.22×10^{10}
HT3 (1-3)	(20, 20)	16	16	4.37×10^{9}
HT4 (2-1)	(40, 15)	25	25	1.89×10^{19}
HT5 (2-2)	(40, 15)	25	25	1.98×10^{18}
HT6 (2-3)	(40, 15)	25	25	2.78×10^{19}
HT7 (3-1)	(60, 30)	28	28	1.67×10^{19}
HT8 (3-2)	(60, 30)	20	20	1.23×10^{20}
HT9 (3-3)	(60, 30)	28	28	1.93×10^{17}
HT10 (4-1)	(60, 60)	49	49	3.02×10^{41}
HT11 (4-2)	(60, 60)	49	49	1.43×10^{41}
HT12 (4-3)	(60, 60)	49	49	3.46×10^{43}
HT13 (5-1)	(60, 90)	73	73	3.01×10^{71}
HT14 (5-2)	(60, 90)	73	73	2.28×10^{64}
HT15 (5-3)	(60, 90)	73	73	3.29×10^{74}
HT16 (6-1)	(80,120)	97	97	4.67×10^{103}
HT17 (6-2)	(80,120)	97	97	2.12×10^{93}
HT18 (6-3)	(80,120)	97	97	3.03×10^{99}
HT19 (7-1)	(160, 240)	196	196	1.08×10^{238}
HT20 (7-2)	(160, 240)	196	196	2.70×10^{222}
HT21 (7-3)	(160, 240)	196	196	8.92×10^{229}
LC1	(400, 200)	9	10	324,513
LC2	(400, 200)	7	15	2.60×10^{6} (1.59×10^{6})
LC3	(400, 400)	5	20	1.32×10^{8}
J1	(70, 80)	14	20	5.78×10^{11}
J2	(70, 80)	16	25	3.81×10^{14}
J3	(120,45)	22	25	4.57×10^{17}
J4	(90, 45)	16	30	1.36×10^{20}
J5	(65, 45)	18	30	2.66×10^{20}
LYT1	(150, 110)	40	40	1.63×10^{32}
LYT2	(160, 120)	50	50	8.16×10^{44} (9.50×10^{43})

It should be recalled here that the evaluation of the problem size is performed based on the procedure of placing elements on the plate (PA algorithm) as applied by us, which does not guarantee the achievement of all possible cuttings related to a given problem. Therefore the actual dimensions of the solution space are larger than those given in Table 1. However, based on this data, the complexity degree of individual problems may be evaluated and compared. Based on the estimates obtained, a decision is made

Table 2 Description of the second set of problems

Test problem	Container (L, W)	m	M	Estimated test problem size
ngcut1	(10, 10)	5	10	170
ngcut2	(10, 10)	7	17	427
ngcut3	(10, 10)	10	21	398,970
ngcut4	(15, 10)	5	7	580
ngcut5	(15, 10)	7	14	1,505
ngcut6	(15, 10)	10	15	44,530
ngcut7	(20, 20)	5	8	985
ngcut8	(20, 20)	7	13	1,246
ngcut9	(20, 20)	10	18	57,830
ngcut10	(30, 30)	5	13	5,645
ngcut11	(30, 30)	7	15	1,309
ngcut12	(30, 30)	10	22	33,220
hccut03	(30, 30)	7	7	5,817
hccut08	(30, 30)	15	15	7.46×10^6
wang20	(70, 40)	19	42	5,947
cgcut03	(40, 70)	20	62	1.20×10^6
okp1	(100, 100)	15	50	1.05×10^{12}
okp2	(100, 100)	30	30	3.94×10^8
okp3	(100, 100)	30	30	2.06×10^9
okp4	(100, 100)	33	61	1.15×10^{12}
okp5	(100, 100)	29	97	5.61×10^{10}

as to whether in a given problem a division of the plate into smaller parts (Sect. 3.2.2) will be performed.

The second set of problems was composed of the following problems:

- 12 problems from Beasley (1985): ngcut1–ngcut12;
- 2 problems from Hadjiconstantinou and Christofides (1995): hccut03, hccut08;
- 1 problem from Wang 1983: wang 20;
- 1 problem from Christofides and Whitlock (1977): cgcut03;
- 5 problems from Fekete and Schepers (1997): okp1–okp5.

These problems are taken from works in which the objective is to maximize the total value of the pieces cut, and the value of the element does not need to correspond to its area. With such an assumption for these problems, the values of optimal solutions are known. It is assumed in our calculations that the value of a given element equals its area. In such case, optimal solutions are not known for all problems (so far the best results are found in Gonçalves 2007).

The problems in the second set may be considered to be problems of small and moderate size. Table 2 provides information regarding the estimated number of nodes of the search tree for the problems in this set.

4.4 Computational results

For each of the described sets of test problems a series of preliminary optimizations was performed with the purpose of selecting proper parameters of the algorithm. Based on these experiments, the values given in Sect. 4.1 were assumed. It should be added that the determined values are a kind of compromise between the universality and efficiency of the algorithm. The point was to adopt such parameter values which would make the algorithm generate satisfactory solutions for a wide range of problems—however, this does not mean that these are optimal parameters of the algorithm operation for each problem. We realize that for some problems the results presented below could be still improved by adjusting parameter values to the specific problem rather than to the entire set of problems.

Despite the fact that as a result of the initial experiments the majority of algorithm parameters was assigned values which were not subject to further changes, in the case of some mechanisms it was still decided to make the manner of their operation conditional on the nature of the problem. Justification for this approach can be found in Table 3, where the calculation results for the first set of test problems are presented.

One of the mechanisms whose application is dependent on the complexity of the problem is the division of the cut plate into smaller parts followed by the initiation of the evolutionary algorithm for each part separately. It turned out that in the case of problems of high complexity the use of such a mechanism not only contributes to improving the results, but also results in a shortening of the calculation time. However, specification of the conditions which need to be met to perform the division may present a difficulty. Based on the computational experiments and determined estimates of the complexity degree of problems a decision was made to perform the division of the plate into two parts in the final version of the algorithm if the estimated size of the search tree will be between 10^{30} and 10^{130} nodes. In the case of problems of larger size the plate will be divided into three parts, while in cases of small size (below 10^{30}) there will be no division. It should be added, however, that in order to establish the principles of the division of the plate into parts in a more precise manner, experiments for a larger number of problems would need to be performed. Already based on analysis of the data in Tables 3 and 1 it can be stated that in the case of problem HT11 (estimated number of nodes in the search tree 1.43×10^{41}) slightly better results were obtained in the version without the division, whereas for problems LYT1 and LYT2 (with sizes of 1.67×10^{32}, 8.19×10^{44}), the plate division contributed to a significant improvement of the results. This problem might be solved to some degree by determining such ranges of the problem complexity degree for which both versions of the algorithm would be initiated, with and without a division.

Moreover, for problems of small and moderate size (i.e. problems for which the division of the plate into smaller parts was not performed), it was decided that in the final version the algorithm will be initiated twice: once for the standard form of problems and once in the variant with rotated elements and plate. By analysis of the results in Tables 2 and 5, it can be stated that in some problems such an approach can bring an improvement in the obtained results. However, in such a case, the obtained computation times should be added. Additionally, the performed computational experiments proved that the application of the plate and element rotation procedure is particularly

Table 3 Computational results for the first set of problems (corrected version in brackets)

Test problem	Optimum	Maximum				Average				Time (s)			
		M2NumOfSteps = 3		Divided into parts		M2NumOfSteps = 3		Divided into parts		M2NumOfSteps = 3		Divided into parts	
		Standard	Rotated	M2NumOf Steps = 3	M2NumOf Steps = 2	Standard	Rotated	M2NumOf Steps = 3	M2NumOf Steps = 2	Standard	Rotated	M2NumOf Steps = 3	M2NumOf Steps = 2
HT1 (1-1)	400	**400**	**400**			**400.00**	**400.00**			1	1		
HT2 (1-2)	400	**400**	**400**			**399.21**	392.00			34	65		
HT3 (1-3)	400	**400**	**400**			**400.00**	**400.00**			1	1		
HT4 (2-1)	600	**600**	**600**			**600.00**	**600.00**			15	20		
HT5 (2-2)	600	**600**	**600**			**600.00**	**600.00**			28	52		
HT6 (2-3)	600	**600**	**600**			**600.00**	**600.00**			11	12		
HT7 (3-1)	1,800	**1,800**	**1,800**			**1,797.75**	1,797.68			94	110		
HT8 (3-2)	1,800	1,792	**1,800**			1,776.00	**1,780.64**			237	249		
HT9 (3-3)	1,800	**1,800**	**1,800**			**1,799.64**	1,798.29			47	52		
HT10 (4-1)	3,600	3,580	**3,592**	**3,592**	**3,592**	3,561.14	3,572.18	**3,574.71**	3,568.96	885	1,050	531	451
HT11 (4-2)	3,600	**3,600**	3,594	3,592	3,596	**3,584.93**	3,581.64	3,553.04	3,557.64	1,028	1,127	492	482
HT12 (4-3)	3,600	**3,600**	3,594	**3,600**	**3,600**	**3,586.29**	3,581.29	3,584.57	3,586.11	1,007	1,001	512	522
HT13 (5-1)	5,400	5,388	5,388	**5,400**	**5,400**	5,380.46	5,360.46	**5,388.50**	5,383.07	2,302	2,905	1,935	1,150
HT14 (5-2)	5,400	5,396	5,358	5,392	**5,397**	**5,372.14**	5,322.71	5,355.68	5,361.04	2,824	2,669	1,587	1,156
HT15 (5-3)	5,400	5,392	5,392	**5,400**	**5,400**	5,371.64	5,371.71	5,382.71	**5,385.75**	2,664	2,663	1,688	1,325
HT16 (6-1)	9,600			9,590	**9,592**			**9,575.39**	9,573.54			4,321	2,538
HT17 (6-2)	9,600			9,594	**9,595**			9,553.61	**9,563.39**			4,488	2,372
HT18 (6-3)	9,600			9,586	**9,594**			**9,566.61**	9,563.36			3,722	2,683
HT19 (7-1)	38,400				**38,271**				**38,157.96**				12,422
HT20 (7-2)	38,400				**38,299**				**38,170.39**				11,391
HT21 (7-3)	38,400				**38,301**				**38,186.71**				11,288

Table 3 continued

Test problem	Optimum	Maximum				Average				Time (s)			
		M2NumOfSteps = 3		Divided into parts		M2NumOfSteps = 3		Divided into parts		M2NumOfSteps = 3		Divided into parts	
		Standard	Rotated	M2NumOf Steps = 3	M2NumOf Steps = 2	Standard	Rotated	M2NumOf Steps = 3	M2NumOf Steps = 2	Standard	Rotated	M2NumOf Steps = 3	M2NumOf Steps = 2
LC1	80,000	**80,000**	**80,000**			**80,000.00**	**80,000.00**			0	0		
LC2	79,000	**79,000**	**79,000**			**79,000.00**	**79,000.00**			1	1		
	(80,000)	**(80,000)**	**(80,000)**			**(80,000.00)**	**(80,000.00)**			(1)	(1)		
LC3	160,000	**160,000**	**160,000**			**160,000.00**	**160,000.00**			3	6		
J1	5,600	**5,600**	**5,600**			**5,600.00**	**5,600.00**			5	8		
J2	5,600	**5,600**	5,540			**5,524.54**	5,440.50			205	157		
J3	5,400	**5,400**	**5,400**			**5,400.00**	**5,400.00**			25	8		
J4	4,050	**4,050**	**4,050**			**4,050.00**	**4,050.00**			33	6		
J5	2,925	**2,925**	**2,925**			2,919.86	**2,925.00**			79	16		
LYT1	16,500	16,332	16,340	**16,444**	**16,444**	16,201.00	16,237.57	16,324.86	**16,345.86**	580	510	335	314
LYT2	?19,200	19,076	19,116	19,124	**19,164**	18,957.14	19,058.00	19,086.71	**19,101.29**	881	830	599	531
	(19,200)	(18,992)	(19,020)	(19,080)	**(19,088)**	(18,927.60)	(18,984.00)	(19,033.20)	**(19,054.00)**	(841)	(936)	(456)	(515)

advisable in the case of problems where the width of the plate is greater than its height. This is related to the fitness function applied by us, which prefers cuttings with empty pieces located as high as possible, which contributes to a better diversification of cuttings with the same filling rate. Therefore, in the case of problems of larger size, where the plate is divided into smaller parts, pieces obtained as a result of such division are always arranged in a vertical orientation.

The value of the `M2NumOfSteps` parameter, which specified the number of steps (search levels) performed by the PA algorithm used by the M2 mutation, also depends on the problem's degree of complexity. As was found from the results of the preliminary experiments, in problems where the plate division into parts was not made, better results were achieved applying the M2 mutation with the `M2NumOfSteps` parameter value set to 3, whereas in the problems of large size it was usually more advantageous to set `M2NumOfSteps` = 2. Increasing the value of this parameter caused a significant increase in the computation time.

Table 4 presents a comparison of the results obtained for set I using our algorithm (HEA) with the results provided by the following authors:

- Binkley and Hagiwara (2007)—they present results of calculations for two algorithms which they developed based on the four corners packing heuristic in combination with either self-adapting parallel recombinative simulated annealing (marked as BH07a in Table 4) or the self-adapting genetic algorithm (BH07b);
- Gonçalves (2007)—he designed a hybrid genetic algorithm based on the original procedure of placing elements on the plate. Table 4 presents results which he obtained (best and average) for 20 (G07a) and 10 (G07b) algorithm runs;
- Alvarez-Valdes et al. (2007)—they developed a new heuristic algorithm based on tabu search techniques (APT07);
- Bortfeldt and Winter (2009)—in their method (marked as BW09 in Table 4) complete cuttings are assembled from a certain number of rectangular layers of various lengths, while their width equals the width of the cut plate. The genetic algorithm is responsible for solution processing (shifting the layer between cuttings);
- Gonçalves and Resende (2011)—they proposed a hybrid genetic algorithm based on random keys with a novel placement procedure (GR11).

In Table 4, the best solutions found for individual problems are distinguished with a bold font. Based on analysis of the results, it can be stated that the algorithm developed by us found solutions of a higher (for 15 problems) or equal value of the objective function for almost every problem, compared with the algorithms mentioned above. Moreover, for the four problems HT12, HT13, HT15, J2 (and, considering additionally different versions of the algorithm, also for problem HT11), our algorithm was the only one to generate optimal solutions. Figure 4 presents the cuttings obtained for two of those four problems.

In the case of problem HT12, the optimal solution was found both by the standard algorithm version (without plate division) and the variant of the algorithm with division of the plate into two parts. The cuttings obtained are shown in Fig. 5.

The averages calculated in our experiments for each problem based on the results from 28 algorithm runs are, for many test problems, higher than the averages presented by Binkley and Hagiwara (2007) and Gonçalves (2007).

Table 4 Comparison with other algorithms (corrected version in brackets)

Test problem	Optimum	Maximum								Average				
		BH07a	BH07b	G07a	G07b	APT07	BW09	GR11	HEA	BH07a	BH07b	G07a	G07b	HEA
HT1 (1-1)	400	**400**	**400**		**400**	**400**	**400**	**400**	**400**	**400.00**	**400.00**		**400.00**	**400.00**
HT2 (1-2)	400	386	396		**400**	**400**	396	**400**	**400**	385.70	386.80		**395.60**	392.00
HT3 (1-3)	400	**400**	**400**		**400**	**400**	**400**	**400**	**400**	**400.00**	**400.00**		**400.00**	**400.00**
HT4 (2-1)	600	**600**	**600**		**600**	**600**	**600**	**600**	**600**	**600.00**	596.20		599.00	**600.00**
HT5 (2-2)	600	**600**	**600**		**600**	**600**	597	**600**	**600**	599.40	598.20		599.40	**600.00**
HT6 (2-3)	600	**600**	**600**		**600**	**600**	**600**	**600**	**600**	**600.00**	599.60		**600.00**	**600.00**
HT7 (3-1)	1,800	**1,800**	**1,800**		**1,800**	**1,800**	**1,800**	**1,800**	**1,800**	**1,800.00**	1,789.70		1, 791.30	1,797.68
HT8 (3-2)	1,800	1,792	1,792		1,786	**1,800**	1,781	1,796	**1,800**	1,780.20	1,769.71		1, 779.10	**1,780.64**
HT9 (3-3)	1,800	**1,800**	**1,800**		**1,800**	**1,800**	**1,800**	**1,800**	**1,800**	1,792.91	1,788.70		1, 785.50	**1,798.29**
HT10 (4-1)	3,600	3,573	3,565		3,580	3,580	3,576	3,591	**3,592**	3,551.69	3,543.41		3, 565.70	**3,568.96**
HT11 (4-2)	3,600	3,577	3,570		3,570	3,564	3,582	3,588	**3,596**	3,561.80	3,546.50		**3, 563.00**	3,557.64
HT12 (4-3)	3,600	3,588	3,584		3,588	3,580	3,586	3,594	**3,600**	3,581.32	3,566.02		3, 582.20	**3,586.11**
HT13 (5-1)	5,400	5,388	5,370	5,388	5,388	5,342	5,388	5,396	**5,400**	5,376.29	5,340.22	5,370.65	5, 372.30	**5,383.07**
HT14 (5-2)	5,400	5,388	5,349	5,388	5,388	5,361	5,386	**5,400**	5,397	**5,377.48**	5,319.49	5,371.50	5, 373.20	5,361.04
HT15 (5-3)	5,400	5,386	5,377	5,392	5,392	5,375	5,382	5,392	**5,400**	5,376.62	5,349.19	5,375.70	5, 378.40	**5,385.75**
HT16 (6-1)	9,600	9,564	9,533	9,556	9,556	9,548	9,555	9,582	**9,592**	9,537.98	9,508.90	9,537.30	9, 540.30	**9,573.54**
HT17 (6-2)	9,600	9,584	9,551	9,590	9,588	9,448	9,575	**9,595**	**9,595**	**9,573.98**	9,492.38	9,570.35	9, 568.50	9,563.39
HT18 (6-3)	9,600	9,568	9,555	9,564	9,564	9,565	9,568	9,582	**9,594**	9,554.78	9,514.27	9,543.55	9, 544.00	**9,563.36**
HT19 (7-1)	38,400	37,941	38,090	38,066	38,040	38,026	38,005	38,146	**38,271**	37,871.62	38,013.31	37,984.21	37, 987.61	**38,157.96**
HT20 (7-2)	38,400	**38,299**	38,199	38,298	38,298	38,145	38,157	**38,374**	38,299	38,202.24	38,144.26	**38,257.45**	38, 254.10	38,170.39

Table 4 continued

Test problem	Optimum	Maximum								Average				
		BH07a	BH07b	G07a	G07b	APT07	BW09	GR11	HEA	BH07a	BH07b	G07a	G07b	HEA
HT21 (7-3)	38,400	38,143	38,171	38,203	38,197	37,867	38,127	38,254	**38,301**	38,054.02	38,099.71	38,120.10	38,102.50	**38,186.71**
LC1	80,000	**80,000**	**80,000**	**80,000**		**80,000**	**80,000**	**80,000**	**80,000**	**80,000.00**	**80,000.00**	**80,000.00**		**80,000.00**
LC2	79,000					**79,000**	**79,000**	**79,000**	**79,000**					**79,000.00**
	(80,000)	**(80,000)**	**(80,000)**	**(80,000)**					**(80,000)**	**(80,000.00)**	**(80,000.00)**	**(80,000.00)**		**(80,000.00)**
LC3	160,000	**160,000**	**160,000**	**160,000**		**160,000**	157,300	**16,000**	**160,000**	**160,000.00**	**160,000.00**	159,865.04		**160,000.00**
J1	5,600	**5,600**	**5,600**	**5,600**		**5,600**	**5,600**	**5,600**	**5,600**	5,577.49	5,505.70	**5,600.00**		**5,600.00**
J2	5,600	5,540	5,540	5,540		5,540	5,512	5,540	**5,600**	5,500.32	5,465.10	5,457.56		**5,524.54**
J3	5,400	**5,400**	**5,400**	**5,400**		**5,400**	**5,400**	**5,400**	**5,400**	**5,400.00**	5,374.78	5,392.80		**5,400.00**
J4	4,050	**4,050**	**4,050**	**4,050**		**4,050**	**4,050**	**4,050**	**4,050**	4,035.58	4,014.00	4,035.60		**4,050.00**
J5	2,925	**2,925**	**2,925**	**2,925**		**2,925**	**2,925**	**2,925**	**2,925**	2,915.99	2,898.00	2,900.71		**2,919.86**
LYT1	16,500	16,160	16,212	16,248		16,280	16,196	16,340	**16,444**	16,100.37	16,064.40	16,120.61		**16,345.86**
LYT2	?192,00					19,044	18,908	19,116	**19,164**					**19,101.29**
	(19,200)	(18,936)	(18,976)	(18,960)					**(19,088)**	(18,772.80)	(18,714.43)	(18,806.81)		**(19,054.00)**

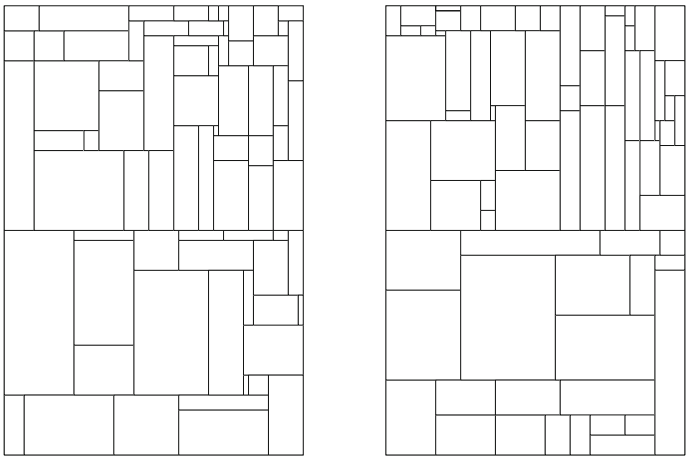

Fig. 4 Cutting patterns with complete filling rates (100 %) for problems HT13 (*left*) and HT15 (*right*)

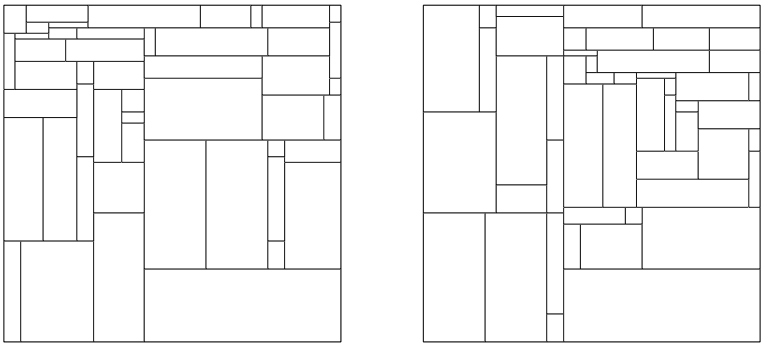

Fig. 5 Cutting patterns of the problem HT12 without division into parts (left) and with division into two parts (*right*)

Equally good results are obtained for the problems from set II, i.e. problems of small and moderate complexity (Table 5).

For the problems in this set the algorithm developed by us found either the optimal solution (where it is known) or a solution with a value equal to the highest value found so far. For the first 16 problems, optimal solutions were found in each algorithm run, regardless of whether the elements were rotated or not. In case of problem okp1 the optimal solution was found only after rotating all elements. It can be assumed that without the rotation of elements, the applied procedure of placing elements on the plate could not generate this solution. Additionally, in the case of the last five problems rotation of elements brought an improvement in the average of the obtained values of the objective function.

5 Conclusions

In this paper a hybrid evolutionary algorithm for the non-guillotine cutting problem has been presented. Many, often specialized mechanisms were employed with the

Table 5 Computational results for the second set of problems

Test problem	Optimum	Maximum			Average			Time (s)	
		Standard	Rotated	G07	Standard	Rotated	G07	Standard	Rotated
ngcut1	95	**95**	**95**	**95**	**95.00**	**95.00**	**95.00**	0.31	0.31
ngcut2	97	**97**	**97**	**97**	**97.00**	**97.00**	**97.00**	0.37	0.40
ngcut3	100	**100**	**100**	**100**	**100.00**	**100.00**	**100.00**	0.44	0.51
ngcut4	138	**138**	**138**	**138**	**138.00**	**138.00**	**138.00**	0.34	0.35
ngcut5	140	**140**	**140**	**140**	**140.00**	**140.00**	**140.00**	0.35	0.40
ngcut6	150	**150**	**150**	**150**	**150.00**	**150.00**	**150.00**	0.45	0.44
ngcut7	175	**175**	**175**	**175**	**175.00**	**175.00**	**175.00**	0.40	0.40
ngcut8	380	**380**	**380**	**380**	**380.00**	**380.00**	**380.00**	0.40	0.43
ngcut9	390	**390**	**390**	**390**	**390.00**	**390.00**	**390.00**	0.66	0.48
ngcut10	879	**879**	**879**	**879**	**879.00**	**879.00**	**879.00**	0.35	0.35
ngcut11	842	**842**	**842**	**842**	**842.00**	**842.00**	**842.00**	0.41	0.54
ngcut12	898	**898**	**898**	**898**	**898.00**	**898.00**	**898.00**	0.60	1.34
hccut03	761	**761**	**761**	**761**	**761.00**	**761.00**	**761.00**	0.38	0.37
hccut08	?807	**807**	**807**	**807**	**807.00**	**807.00**	**807.00**	1.55	2.90
wang20	2,726	**2,726**	**2,726**	**2,726**	**2,726.00**	**2,726.00**	**2,726.00**	1.30	0.77
cgcut03	2,726	**2,726**	**2,726**	**2,726**	**2,726.00**	**2,726.00**	**2,726.00**	0.58	1.33
okp1	9,974	9,938	**9,974**	**9,974**	9,938.00	**9,974.00**	**9,974.00**	69.60	9.21
okp2	?9,876	**9,876**	9,847	**9,876**	9,800.54	**9,847.00**	9,801.20	69.60	67.60
okp3	9,877	**9,877**	**9,877**	**9,877**	9,812.86	**9,877.00**	9,861.44	58.71	11.17
okp4	?9,976	**9,976**	**9,976**	**9,976**	9,965.50	**9,976.00**	**9,976.00**	69.11	68.87
okp5	?9,982	**9,982**	**9,982**	**9,982**	9,948.18	**9,977.79**	9,974.60	74.36	75.50

aim of improving the efficiency of our approach. Various mutation operators were developed and tested, and finally it was decided to use three of them. In addition, a procedure of post-optimization was applied to improve the solution generated by the evolutionary algorithm. In the case of test problems of large size a decision was made to perform a division of the cut plate into smaller parts. The developed algorithm was tested on a number of test problems derived from the literature, and the results obtained were compared with those presented by other authors, including those applying metaheuristics. The computational experiments performed demonstrate the high efficiency of our method. For almost all of the test problems our algorithm found a better or equally good solution as the solutions obtained by other authors, and in the case of some problems it was the only one that found an optimal solution.

References

Alvarez-Valdes R, Parreño F, Tamarit JM (2007) A Tabu Search algorithm for two-dimensional non-guillotine cutting problems. Eur J Oper Res 183:1167–1182

Arenales M, Morabito R (1995) An AND/OR-graph approach to the solution of two-dimensional non-guillotine cutting problems. Eur J Oper Res 84:599–617

Baker BS, Coffman EG, Rivest RL (1980) Orthogonal packing in two dimensions. SIAM J Comput 9:846–855

Beasley JE (1985) An exact two-dimensional non-guillotine cutting tree search procedure. Oper Res 33:49–64

Beasley JE (2004) A population heuristic for constrained two-dimensional non-guillotine cutting. Eur J Oper Res 156:601–627

Binkley KJ, Hagiwara M (2007) Applying self-adaptive evolutionary algorithms to two-dimensional packing problems using a four corners' heuristic. Eur J Oper Res 183:1230–1248

Bortfeldt A, Winter T (2009) A genetic algorithm for the two-dimensional knapsack problem with rectangular pieces. Int Trans Oper Res 16:685–713

Boschetti MA, Hadjiconstantinou E, Mingozzi A (2002) New upper bounds for the two-dimensional orthogonal non guillotine cutting stock problem. IMA J Manag Math 13:95–119

Burke EK, Kendall G, Whitwell G (2004) A new placement heuristic for the orthogonal stock-cutting problem. Oper Res 52:655–671

Caprara A, Monaci M (2004) On the two-dimensional knapsack problem. Oper Res Lett 32:5–14

Chazelle B (1983) The bottom-left bin packing heuristic: an efficient implementation. IEEE Trans Comput 32:697–707

Christofides N, Whitlock C (1977) An algorithm for two dimensional cutting problems. Oper Res 25:30–44

Clautiaux F, Carlier J, Moukrim A (2007) A new exact method for the two-dimensional orthogonal packing problem. Eur J Oper Res 183:1196–1211

Dowsland KA (1993) Some experiments with simulated annealing techniques for packing problems. Eur J Oper Res 68:389–399

Dowsland KA, Dowsland WB (1992) Packing problems. Eur J Oper Res 56:2–14

Dyckhoff H (1990) A typology of cutting and packing problems. Eur J Oper Res 44:145–159

Fekete SP, Schepers J (1997) A new exact algorithm for general orthogonal d-dimensional knapsack problems. Springer Lect Notes Comput Sci 1284:144–156

Furian N, Vössner S (2011) Constrained order packing: comparison of heuristic approaches for a new bin packing problem. Cent Eur J Oper Res. doi:10.1007/s10100-011-0228-1

Gonçalves JF (2007) A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. Eur J Oper Res 183:1212–1229

Gonçalves JF, Resende MGC (2011) A parallel multi-population genetic algorithm for a constrained two-dimensional orthogonal packing problem. J Comb Opt 22:180–201

Hadjiconstantinou E, Christofides N (1995) An exact algorithm for general, orthogonal, two-dimensional knapsack problems. Eur J Oper Res 83:39–56

Hadjiconstantinou E, Iori M (2007) A hybrid genetic algorithm for the two-dimensional single large object placement problem. Eur J Oper Res 183:1150–1166

Hässler RW, Sweeney PE (1991) Cutting stock problems and solution procedures. Eur J Oper Res 54:141–150

Hopper E, Turton BCH (2001) An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. Eur J Oper Res 128:34–57

Jakobs S (1996) On genetic algorithms for the packing of polygons. Eur J Oper Res 88:165–181

Lai KK, Chan JWM (1997a) An evolutionary algorithm for the rectangular cutting stock problem. Int J Ind Eng 4:130–139

Lai KK, Chan JWM (1997b) Developing a simulated annealing algorithm for the cutting stock problem. Comput Ind Eng 32:115–127

Leung TW, Chan CK, Troutt MD (2001) Applications of genetic search and simulated annealing to the twodimensional non-guillotine cutting stock problem. Comput Ind Eng 40:201–214

Leung TW, Yung CH, Troutt MD (2003) Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem. Eur J Oper Res 145:530–542

Leung SCH, Zhang D, Zhou C, Wu T (2012) A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem. Comput Oper Res 39:64–73

Liu D, Teng H (1999) An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. Eur J Oper Res 112:413–420

Lodi A, Martello S, Monaci M (2002) Two-dimensional packing problems: a survey. Eur J Oper Res 141:241–252

Mack D, Bortfeldt A (2012) A heuristic for solving large bin packing problems in two and three dimensions. Cent Eur J Oper Res 20:337–354

Michalewicz Z (1992) Genetic algorithms + data structure = evolution programs. Springer, New York

Scheithauer G, Terno J (1993) Modeling of packing problems. Optimization 28:63–84

Sweeney PE, Paternoster ER (1992) Cutting and packing problems: a categorized, application-orientated research bibliography. J Oper Res Soc 43:691–706

Wang PY (1983) Two algorithms for constrained two-dimensional cutting stock problems. Oper Res 31:573–586

Wäscher G, Haussner H, Schumann H (2007) An improved typology of cutting and packing problems. Eur J Oper Res 183:1109–1130

Wei L, Oon WC, Zhu W, Lim LCA (2011) A skyline heuristic for the 2D rectangular packing and strip packing problems. Eur J Oper Res 215:337–346