ELSEVIER

European Journal of Operational Research 88 (1996) 165–181

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

# On genetic algorithms for the packing of polygons

Stefan Jakobs

*RWTH Aachen, Lehrstuhl C für Mathematik, Templergraben 55, D-52062 Aachen, Germany*

Received June 1993

**Abstract**

A genetic algorithm for placing polygons on a rectangular board is proposed. The algorithm is improved by combination with deterministic methods.

*Keywords:* Optimization; Genetic algorithms; Mathematical programming; Adaptive processes; Packing problems

## 1. Introduction and motivation

In the steel industry problems frequently occur when the need to stamp polygonal figures from a rectangular board arises. The aim is to maximize the use of the contiguous remainder of the board. Similar problems exist in the textile industry, when clothes are cut out of a rectangular piece of material.

In order to solve these problems let us consider the following simpler approach. Given a finite number of rectangles $r_i$, $i = 1, \ldots, n$, and a rectangular board, an *orthogonal packing pattern* requires by definition a disjunctive placement of the rectangles on the board in such a way that the edges of $r_i$ are parallel to the $x$- and $y$-axes, respectively. The computation of the orthogonal packing pattern with minimal height is called *orthogonal packing problem* (OPP).

Baker, Coffman and Rivest propose an heuristic for the orthogonal packing problem; in addition they present an upper bound for the height of the packing pattern [2]. A recent survey on packing problems and their respective heuristics is given in [16]. The extension from rectangles to polygons can be realized in several ways. The first method places the polygons directly on the board and then the algorithm optimizes locally by means of shifts and rotations [23]. A second approach places two or three polygons in a cluster. The clusters are then placed on the board [1].

In this article we use another approach, namely an evolutionary algorithm. There are three main classes in this approach, each of which is independently developed. The first class is called *evolutionary programming* (EP). L.J. Fogel, Owens, and Walsh were the first to develop the EP-algorithms [5]. D.B. Fogel has recently improved this approach [6]. The second class was developed by Rechenberg and Schwefel. They called their approach *evolutionary strategies* (ES) [17–20]. Finally, Holland developed the so called *genetic algorithm* (GA) [12]. The genetic algorithm has been perfected by De Jong [13] and Goldberg [9].

The paper is organized as follows. It begins by explaining the problem and its complexity. In the next section the data structure and its transformation into a packing pattern are described. Sec-


*SSDI* 0377-2217(94)00166-A

tion 4 provides the genetic algorithm in combination with a deterministic algorithm, and numerical examples are presented. In Section 5 two approaches for the extension to polygons are proposed. The straightforward extension applies the genetic algorithm directly to the polygons. This method results, however, in a rather long computing time. An alternative to this method is the application of the genetic algorithm to rectangles in which the polygons are embedded; subsequently, the use of a deterministic shrinking step moves the polygons closer to each other.

## 2. The problem

The size of the search space of the orthogonal packing problem is infinite, because every movement of a rectangle into a packing pattern in a feasible direction creates a new packing pattern. In order to effectively reduce the number of possible orthogonal packing patterns the so called *bottom-left-condition* (BL-condition) is introduced. The orthogonal packing pattern fulfills the BL-condition if no rectangle can be shifted further to the bottom or to the left.

In addition, the complexity of the problem must be considered. The OPP is a natural generalization of the one-dimensional bin-packing problem. Indeed, if all rectangles are required to have the same height, then the two problems coincide. On the other hand, the case in which all rectangles have the same width corresponds to the well-known makespan minimization problem of combinatorial scheduling theory. Both these restricted problems are known to be NP-complete [8].

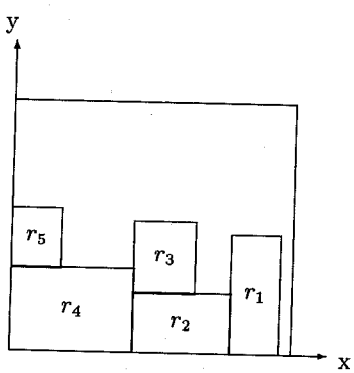
Other authors as Sleator, who did not use the BL-condition, mentioned that the packing problem can be reduced to the partition problem [22,7].

## 3. Data structure

The data structure is important for the genetic algorithm. The first genetic algorithms (shortly GAs) worked with bit-strings. Over the last few years, GAs have been developed which are based on other data structures. In this way the difference between GAs and the evolutionary strategies has diminished [15]. The theory about genetic algorithms calls the data structure a *genotype* and its decoding (here: packing pattern) is called *phenotype*. These technical terms are based on biological terminology [9].

The natural representation of a packing pattern is based on the placement-coordinates of each rectangle on the board. If the left lower and the right upper corner of all rectangles are known, then the packing pattern can be reconstructed easily. For example see Fig. 1.

The advantage of the natural representation lies in its easy reconstruction. But if small changes in the coordinates are made it is probable that a packing pattern with overlaps will be created.




| rectangle | $(x_0, y_0)$ | $(x_1, y_1)$ |
|---|---|---|
| $r_1$ | (18,0) | (22,10) |
| $r_2$ | (10,0) | (18,5) |
| $r_3$ | (10,5) | (15,11) |
| $r_4$ | (0,0) | (10,7) |
| $r_5$ | (0,7) | (4,12) |

Fig. 1. Natural representation of a packing pattern.

This property of natural representation is, however, not suited for GA. Consequently, a more variable data structure is needed.

Alternatively, a packing pattern can be represented by a permutation $\pi$.

$i_j$ – Index of the rectangle ($r_{i_j}$).
$\pi = (i_1, \ldots, i_n)$ – Permutation.

The permutation represents the sequence in which the rectangles are packed. The advantage of this data structure is the facile creation of new permutations by changing the sequence. A consequence of the variable data structure is the fact that every permutation has to be assigned to a unique packing pattern. This decoding of the genotype needs more effort than the conversion of the natural representation into the packing pattern. Hence, the aim is to create a fast decoding algorithm.

### 3.1. BL-algorithm

*Step 1.* Place $r_{\pi(1)}$ into the left lower corner of the board.

*Step i.* Shift $r_{\pi(i)}$ alternately, beginning from the upper right corner of the board, as far as possible to the bottom and then as far as possible to the left.

Fig. 2 illustrates the packing process of the BL-algorithm.

It is easy to show, that the packing pattern, which is created by the BL-algorithm, fulfills the BL-condition, because otherwise at least one of the rectangles could be shifted further to the bottom or to the left. This is in contradiction to Step i of the BL-algorithm.

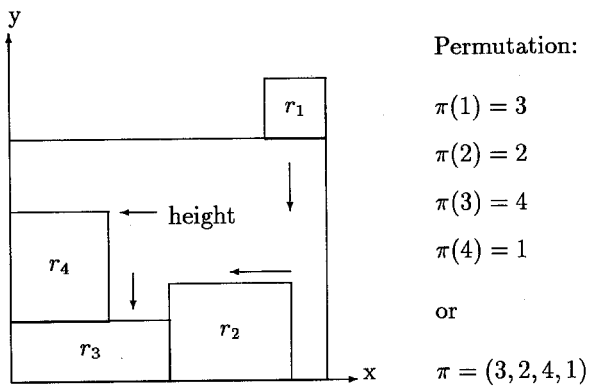



Fig. 2. Illustration of the BL-algorithm.

Some properties of the BL-algorithm are presented below. The first one is an upper bound to the possible packing patterns. Given $n$ rectangles, the number $2^n \cdot n!$ is an upper bound to the packing patterns which can be calculated by the BL-algorithm. This is a consequence of the fact that the orthogonal packing problem is a permutation problem. So there are $n!$ sequences of rectangles. Furthermore each rectangle can be placed in two ways such that the edges are parallel to the $x$- and $y$-axes. In practice, less packing patterns than $2^n \cdot n!$ can be created by the BL-algorithm. For example in Fig. 3 two permutations have the same packing pattern.

The magnitude of the search space is larger than the search space in the travelling salesman

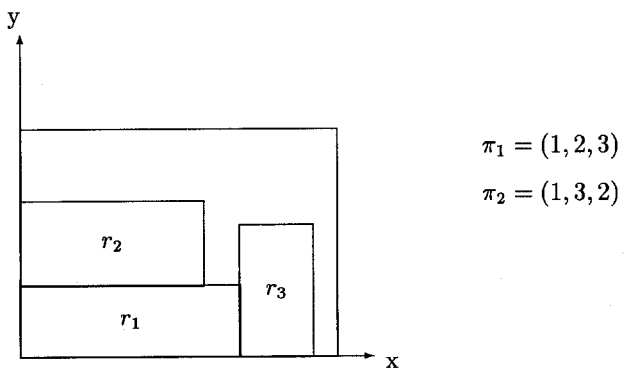



Fig. 3. Two permutations with the same packing pattern.

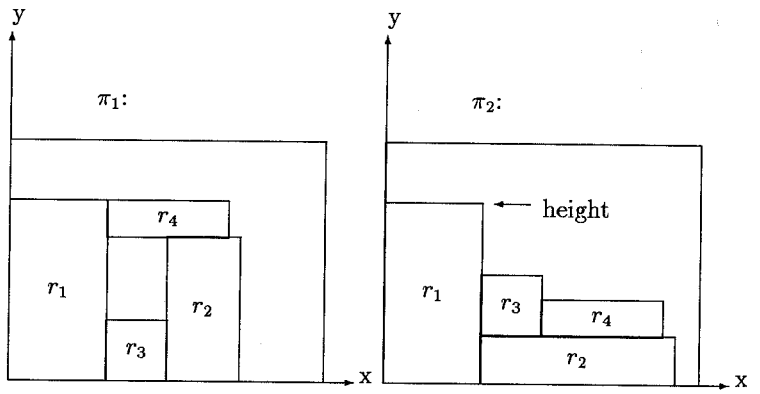



Fig. 4. Fitness-function based on the height of the packing pattern.

problem. If for example $n = 25$ rectangles are given, then

$$2^{25} \cdot 25! \geq 10^{7.5} \cdot (25/e)^{25} \geq 10^{7.5} \cdot 10^{24} \geq 10^{31}$$

orthogonal packing patterns exist.

An additional property of the BL-algorithm from Baker, Coffman and Rivest is based on the following special case. Let the axes-parallel rectangles be sorted according to the width, which corresponds to the $x$-coordinates of the rectangles given by a database, i.e.

$$width(r_{\pi(i)}) \geq width(r_{\pi(j)}) \quad \text{for } i < j.$$

Then the estimation

$$h_{BL} \leq 3 \cdot h_{OPT}$$

holds, where $h_{BL}$ and $h_{OPT}$ denote the BL-algorithm height and the optimal height of the packing pattern, respectively [2].

In addition, the cost of the BL-algorithm is $\mathcal{O}(n^2)$. This based on the fact, that each rectangle $r_i$ can be shifted a maximum of $i$ times, because each shift is limited by one of the $i-1$ placed rectangles or by the corners of the board. Hence, the cost of placing rectangle $r_i$ is $\mathcal{O}(i)$ and the whole cost amounts to $\mathcal{O}(n^2)$.

## 4. Genetic algorithm

For the GA an evaluation of the packing pattern is necessary. This is represented by an appropriate fitness-function

$$f: \pi \to \mathbb{R}_+$$

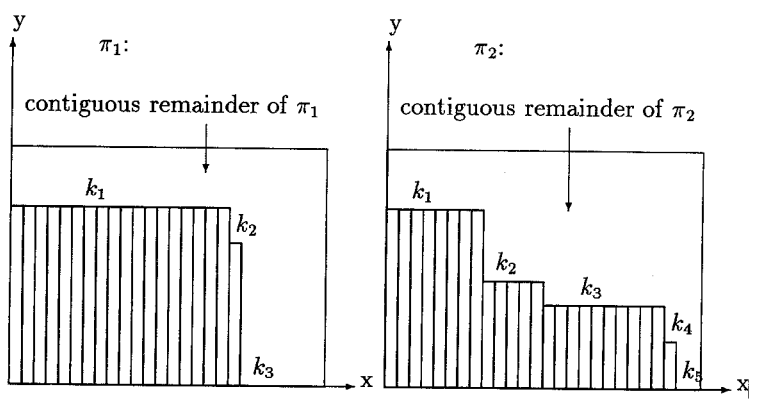



Fig. 5. Contiguous remainder of the packing patterns.

with the property

$$f(\pi_i) > f(\pi_j)$$

if $\pi_i$ is a 'better' packing pattern than $\pi_j$. The computation of the natural approach of the fitness-function is inversely proportional to the height of the packing pattern.

$$f(\pi) = 1/h_{\mathrm{BL}}(\pi).$$

If two packing patterns have the same height, their fitness-values are equal, although one of the packings is 'better' (see Fig. 4).

For this reason a differentiated approach is necessary. In order to find a differentiated fitness-function the biggest resulting contiguous remainder among the packing patterns on the given board must be considered. Fig. 5 shows the contiguous remainder of the packing patterns from Fig. 4. It is evident that the contiguous remainder of $\pi_2$ is greater than of $\pi_1$. The comparison suggests the following fitness-function;

$$f(\pi) = Area(ContiguousRemainder(\pi)) = \sum_{k_i} |x_1^{(i)} - x_2^{(i)}| \cdot (height(board) - y^{(i)})$$

with

$$k_i = \{(x_1^{(i)}, y^{(i)}), (x_2^{(i)}, y^{(i)})\}.$$

### 4.1. Initialization

Now we come to the GA and its operators. Each evolutionary algorithm, in particular genetic algorithms, work with $m$ objects in our case packing patterns:

$$\pi_1, \ldots, \pi_m.$$

Each packing pattern is assigned its fitness value:

$$f_i = f(\pi_i), \quad i = 1 \ldots m.$$

Each individual $A_i$ is defined by the permutation $\pi_i$ and its fitness $f_i$:

$$A_i = (\pi_i, f_i).$$

All individuals together represent the population, which is initialized as follows. The width-sorted sequence of all rectangles forms the first permutation $\pi_1$. In contrast to this operation, $\pi_2, \ldots, \pi_m$ represent random permutations. Then it is guarantee that the height of the best individual of the initial population ($h_{\mathrm{best}}$) fulfills the following inequality:

$$h_{\mathrm{best}} \leq 3 \cdot h_{\mathrm{OPT}}$$

After the initialization the BL-algorithm computes the fitness of $\pi_1, \ldots, \pi_m$.

### 4.2. Proportional selection

The first genetic operator selects two individuals with the probability of

$$p_i = f_i / \left( \sum_{j=1}^{m} f_j \right) > 0.$$

In practice the interval $I = [0, 1)$ is divided into $m$ sub-intervals, such that each individual is assigned a sub-interval.

$$A_1 \leftrightarrow I_1 = [0, p_1),$$
$$A_2 \leftrightarrow I_2 = [p_1, p_1 + p_2),$$
$$\vdots$$
$$A_m \leftrightarrow I_m = [1 - p_m, 1).$$

Then two random numbers $p_i \in [0, 1)$, $i = 1, 2$, are generated and the corresponding sub-intervals determine the individuals.

### 4.3. Crossover

In contrast to the classical Crossover [9,15] the Crossover-Operator presented here produces a new permutation from the two selected individuals. The example of two selected permutations below explains the operator:

$\pi_i = (1, 2, 3, 4, 5, 6)$ and
$\pi_j = (6, 4, 2, 5, 3, 1)$.

At the random position $p$ the crossover copies $q$ elements out of $\pi_i$ to the beginning of the new permutation $\pi_{\mathrm{new}}$ with $1 \leq p,\ q \leq n$. For $p = 2$, $q = 3$ we get

$\pi_{new}(1) = \pi_i(p) = \pi_i(2) = 2.$
$\pi_{new}(2) = \pi_i(p+1) = 3.$
$\pi_{new}(3) = \pi_i(p+2) = 4.$

Finally, $\pi_{new}$ is filled up by the other elements of $\pi_j$ in the same order:

$\pi_{new}(4) = \pi_j(1) = 6.$
$\pi_{new}(5) = \pi_j(4) = 5.$
$\pi_{new}(6) = \pi_j(6) = 1.$

So it follows

$\pi_{new} = (2, 3, 4, 6, 5, 1).$

### 4.4. Mutation

Subject to a small mutation rate $p_m$ the operator mutates the offspring $\pi_{new}$. There are a lot of approaches for the implementation of the mutation-operator [15]. For example the sequence of a random block is inverted or some elements of the permutation are exchanged. From now on for the rest of this paper this kind of mutation will be called *NormalMutation*. A further approach compensating for the disadvantage of deterministic packing algorithms is described as follows.

The mutation operator rotates rectangles at random by 90° with a probability of $p_m$.

```
FOR i := 1 TO n DO
BEGIN
  p = Random(0, 1)
  IF p < p_m THEN
     Rotate(π_new(i))
END
```

After completion of the three genetic operations the offspring is converted into the phenotype (packing pattern). In practice the BL-algorithm is used. Then the the fitness is computed and subsequently, the worst individual characterized by the least fitness in the current population will be replaced by the offspring. All these procedures are repeated until a given upper bound to the loops is reached or further improvement is not noticed over a given period of time. In Table 1, a presentation of the whole GA with the cost of the procedures is given.

We emphasize that the cost of the GA is determined by the cost of the BL-algorithm. That is the reason why our GA requires a method of operation different to that of the classical GA. In fact, the classical GA computes m offsprings before sorting out the bad individuals by selection. In contrast to the classical approach one of the worst individuals is sorted out after an offspring has been created by the BL-algorithm. So a good offspring could influence the next population. The genetic operators presented here are only a small extract of all possibilities. Otherwise we would blow up the size of this paper. Most important however is the main idea of the combination of deterministic procedures with genetic algorithms for considerable improvement of deterministic computated results.

Table 1

| Procedures | Cost |
|---|---|
| $\pi_1$ = SortRectangles() | $\mathcal{O}(n \cdot \log n)$ |
| BL-algorithm($\pi_1$) | $\mathcal{O}(n^2)$ |
| $f_1 = f(\pi_1)$, $A_1 = (\pi_1, f_1)$ | $\mathcal{O}(n)$ |
| FOR $i := 2$ TO $m$ DO | |
| BEGIN | |
| $\pi_j$ = RandomPermutation() | |
| BL-algorithm($\pi_j$) | $\mathcal{O}(n^2)$ |
| $f_j = f(\pi_j)$, $A_j = (\pi_j, f_j)$ | $\mathcal{O}(n)$ |
| END | $\mathcal{O}(m \cdot n^2)$ |
| $t = 1$ | |
| WHILE $t <$ MAX_LOOPS DO | |
| BEGIN | |
| $i$ = SelectIndividual() | $\mathcal{O}(m)$ |
| $j$ = SelectIndividual() | $\mathcal{O}(m)$ |
| $\pi_{new}$ = Crossover($\pi_i$, $\pi_j$) | $\mathcal{O}(n)$ |
| MutationNormal($\pi_{new}$) | $\mathcal{O}(n)$ |
| Mutation($\pi_{new}$) | $\mathcal{O}(n)$ |
| BL-algorithm($\pi_{new}$), $f_{new} = f(\pi_{new})$ | $\mathcal{O}(n^2)$ |
| ReplaceWorstIndividual($\pi_{new}$) | $\mathcal{O}(m)$ |
| $t = t + 1$ | |
| END | |
| | $\mathcal{O}(t \cdot m \cdot n^2)$ |

### 4.5. Numerical examples

In the first example 25 rectangles with integer dimensions are to be packed. For testing packing

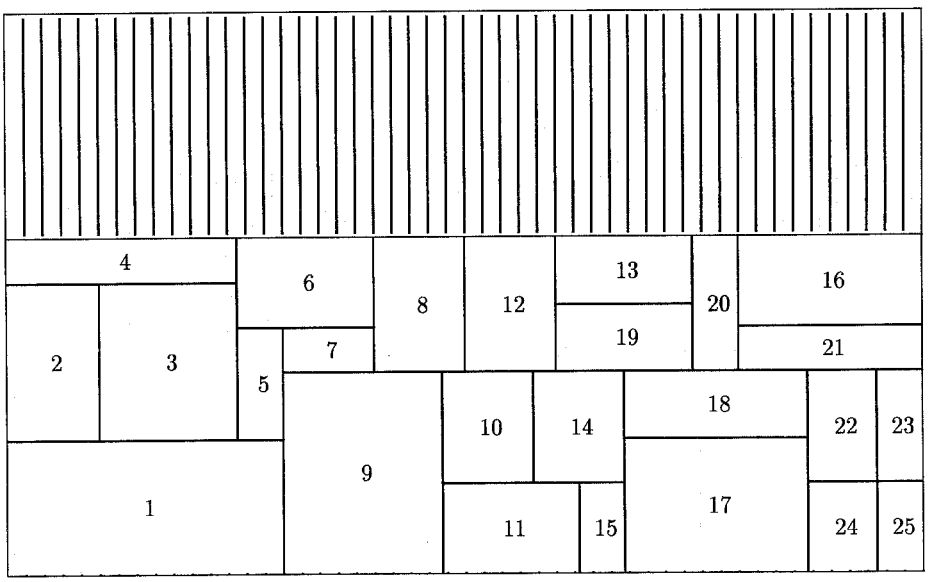



Fig. 6. Optimal packing pattern; height = 15.

algorithms we start with some optimal packing pattern. To do so, a big rectangle (here: $40 \times 15$ units) is divided randomly into 25 rectangles (see Fig. 6).

If these rectangles are packed on a board with the dimensions $40 \times \textit{height}$ ($\textit{height} > 15$), the optimal height is 15.

If these rectangles are sorted according to width, the BL-algorithm generates the following packing pattern (see Fig. 7). For improving the

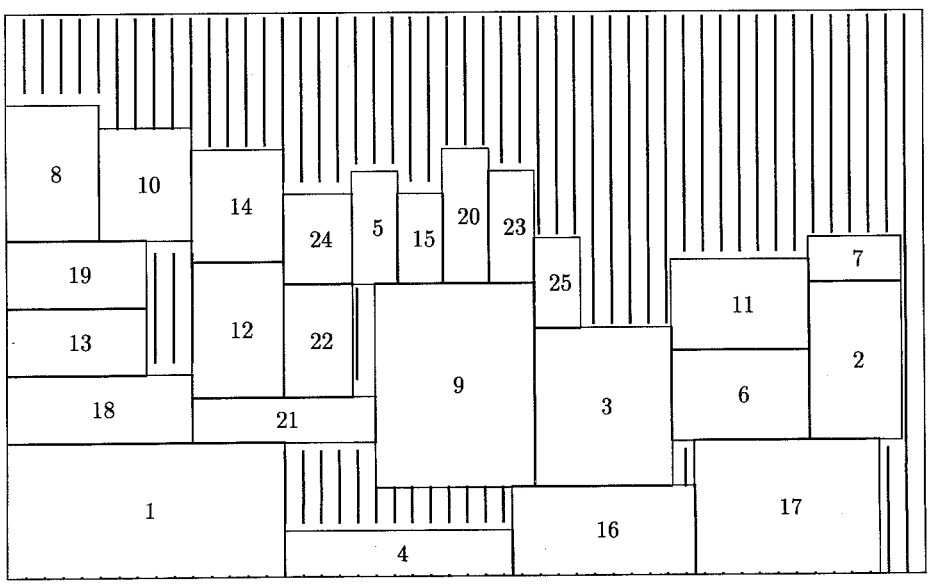



Fig. 7. Packing pattern generated by the BL-algorithm with rectangles sorted by width; height = 21.

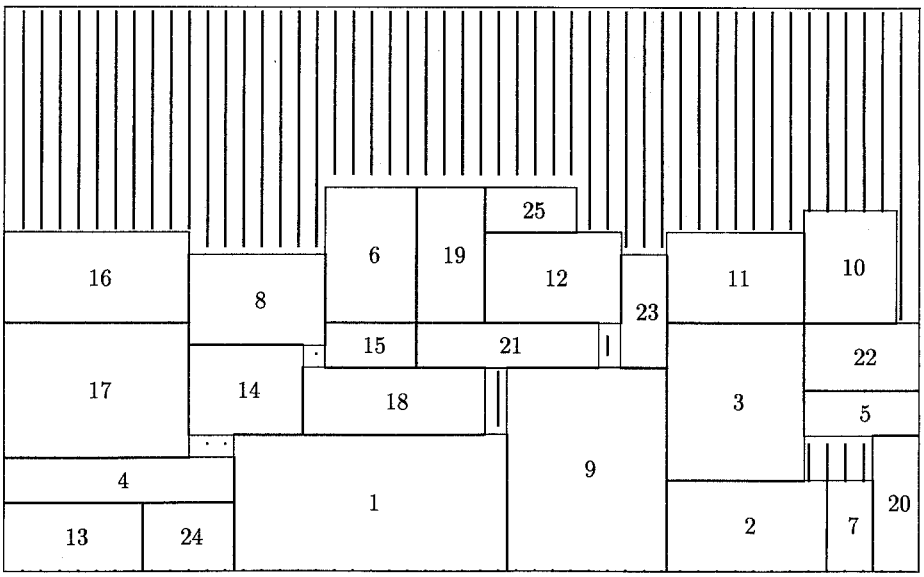



Fig. 8. Packing pattern generated by GA (2000 steps); height = 17.

packing pattern the GA is executed in 2000 steps with the following parameters:

$n = 25$.
Width(board) = 40.
Height(board) = 25.
$m = 20$.
MAX_LOOPS = 2000.
$p_m = 0.4$.

After 2000 steps the best individual is shown in Fig. 8. The improvement of the height amounts to 19% and only a few small gaps exist. The progression of the minimum and the average height through all steps for this special run is shown in Fig. 9.

The results of the computed heights only indicate facts about this corresponding run. The problem is, how do we find the result of the next run? The solution is given by the *Law of Large*

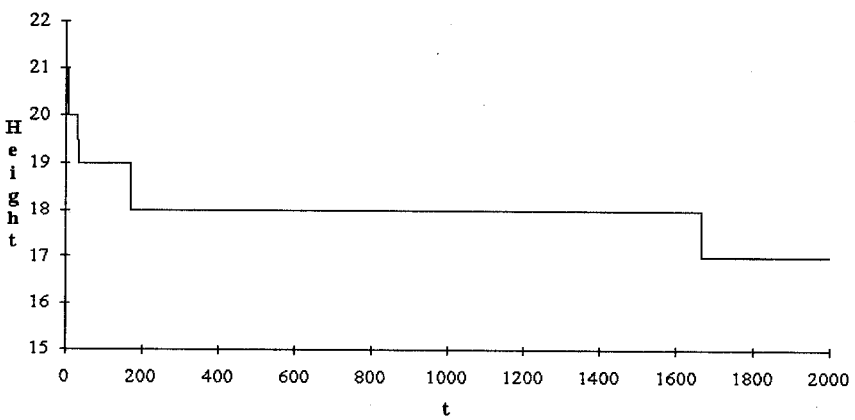



Fig. 9. Heights of the best individuals.

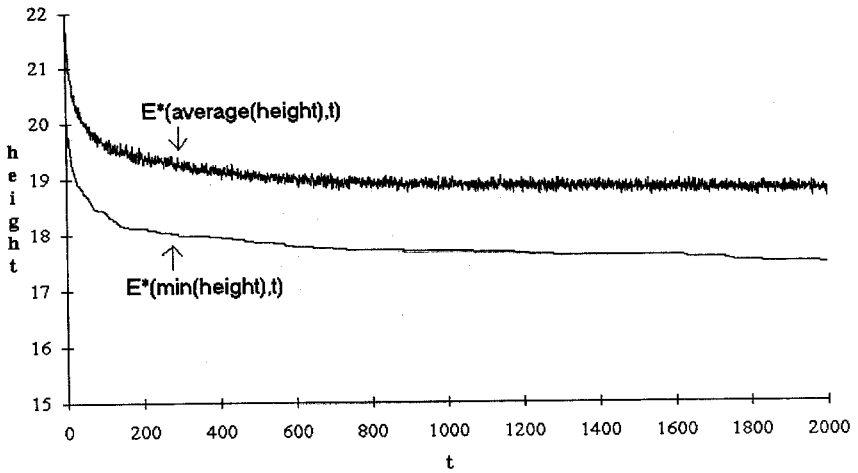



Fig. 10. Approximated estimations of the minimal and average heights.

*Numbers*: The arithmetic mean of the result of the stochastic process repeated $n$ converges to the expectation $E$. To compute approximately the $E(height, t)$, the GA run is repeated 100 times.

$$E(height, t) \approx E^*(height, t)$$

$$:= \frac{1}{100} \sum_{i=1}^{100} height(i, t),$$

$$t = 1 \ldots 2000,$$

with $i$ representing the $i$-th run. In Fig. 10 the approximated expectations of the minimal heights

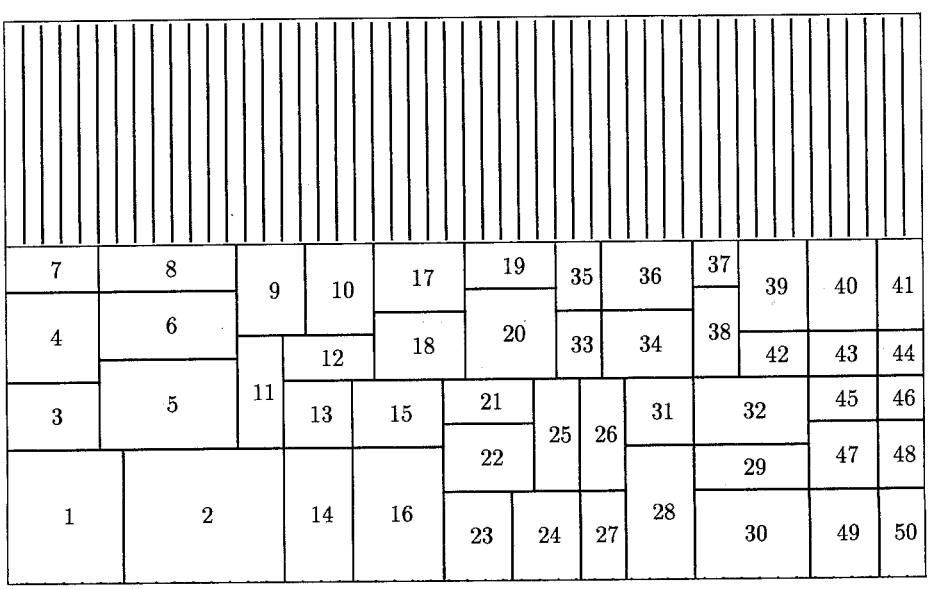



Fig. 11. Optimal packing pattern; height = 15.

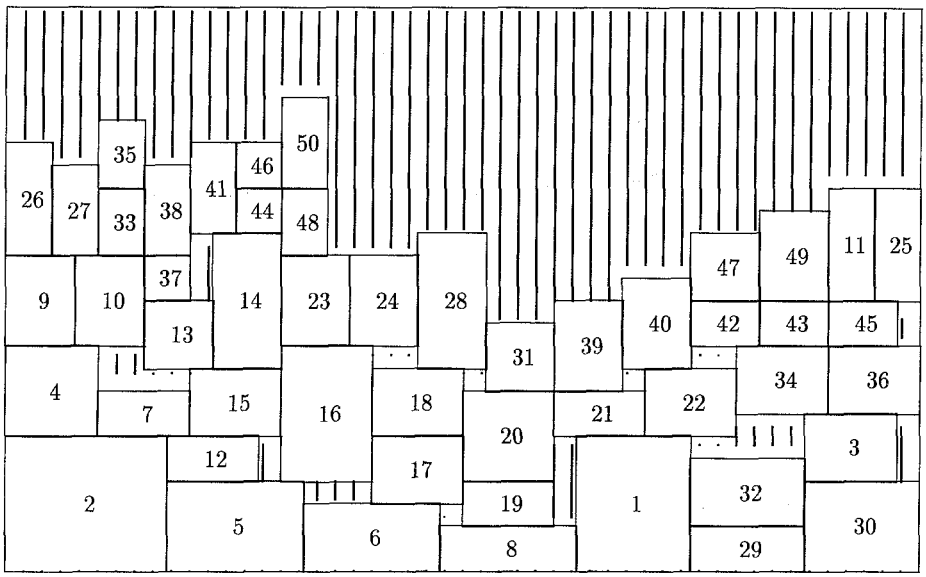



Fig. 12. Packing pattern generated by the BL-algorithm with rectangles sorted by width; height = 21.

and the approximated expectations of the average heights are shown. Both curves fall exponentially and

$E^*(\min(height), 2000) = 17.48.$

The major issue is: With a probability of over 50% a packing pattern with the height 17 is computed after 2000 steps.

The following example corresponds to the afore-mentioned case. The same rectangle used earlier ($40 \times 15$) is randomly divided into 50 rectangles instead of 25. The optimum packing pattern is shown in Fig. 11. Fig. 12 shows the packing pattern created by the BL-algorithm based on the rectangles sorted by width. The development of the approximated expectations are presented

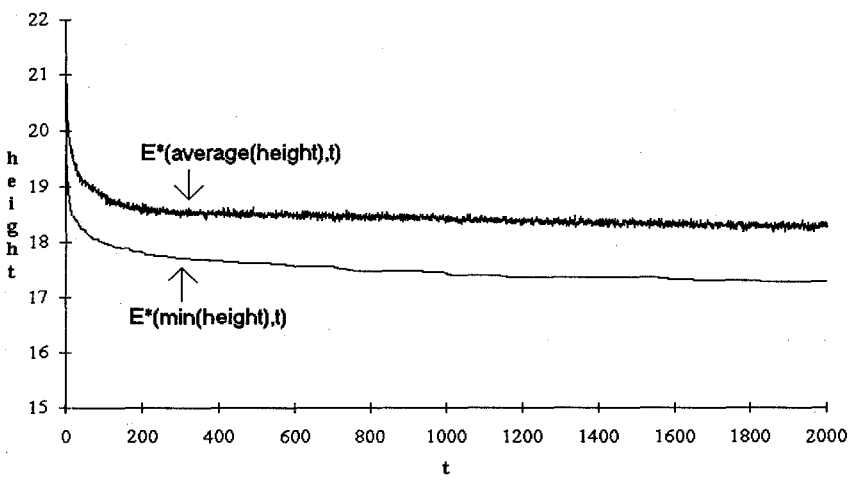



Fig. 13. Approximated estimations of the minimal and average heights.

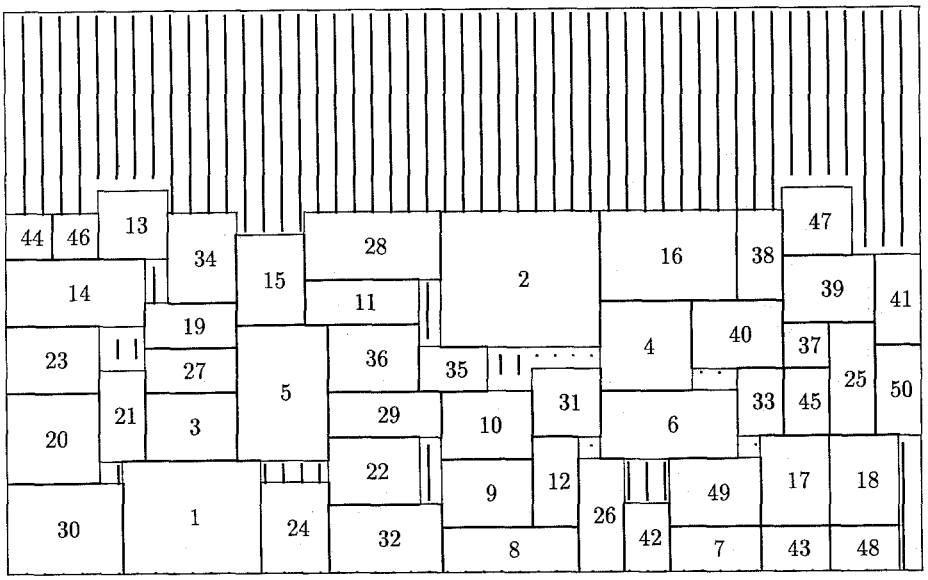



Fig. 14. Packing pattern computed after 2000 steps; height = 17.

in Fig. 13. The expected minimum height is greater than in the example described above.

$E^*(\min(\textit{height}), 2000) = 17.28.$

This result is based on the fact that smaller rectangles can be packed closer together. The packing pattern computed after 2000 steps of the 50 rectangles is shown in Fig. 14.

## 5. Extension to polygons

One approach for the extension to polygons is based on the use of a deterministic algorithm to convert the permutation of polygons into a packing pattern. The cost of existing algorithms [23,16] is greater than $\mathcal{O}(n^2)$. In the GA for each step one permutation has to be converted. For this reason it is not advisable to use this approach.

Our *Embedding-Shrinking Algorithm* offers a faster alternative. It consists of three steps:

*Step 1.* Embed the polygons into rectangles.

*Step 2.* Apply the GA to the embedded rectangles.

*Step 3. Shrinking-Step*: Shift the polygons closer to each other.

The first step is to determine the embedding rectangles with minimum area for all polygons. To achieve this aim we use the following heuristic. A polygon is rotated once around the centre of gravity of all its corner points in a fixed number of equal angular increments. At each increment the embedding rectangle parallel to the $x$- and $y$-axes with minimum area is computed. Finally, a minimisation of all increments is performed (see Fig. 15). Alternatively, more than

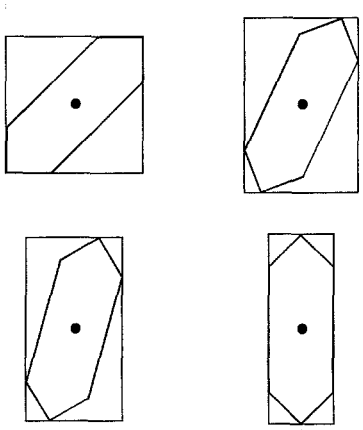


Fig. 15. Determining the embedding rectangle with minimum area.

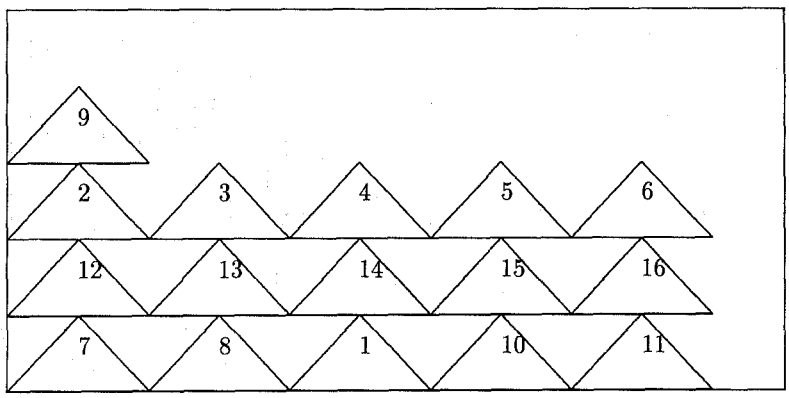



Fig. 16. Pattern which cannot be improved by shrinking; height = 16.

one polygon is put into a rectangle called *cluster* [23,16].

In Step 2 the genetic algorithm starts packing with the embedding rectangles. If the fitness of the best individual no longer improves, it is necessary to move the polygons closer together, be-

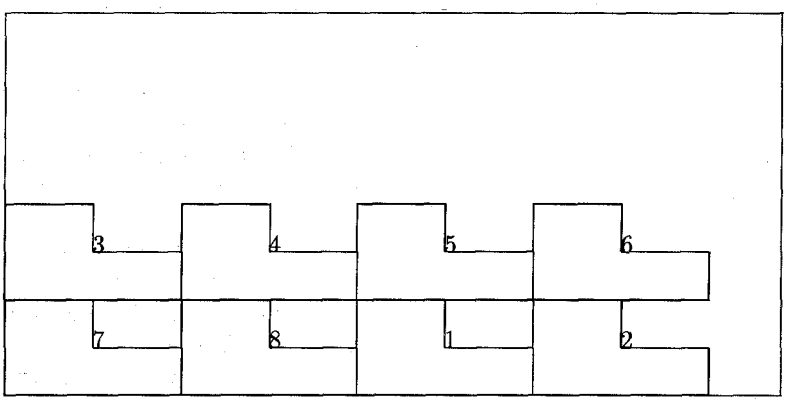



Fig. 17. Pattern which cannot be improved by shrinking; height = 10.

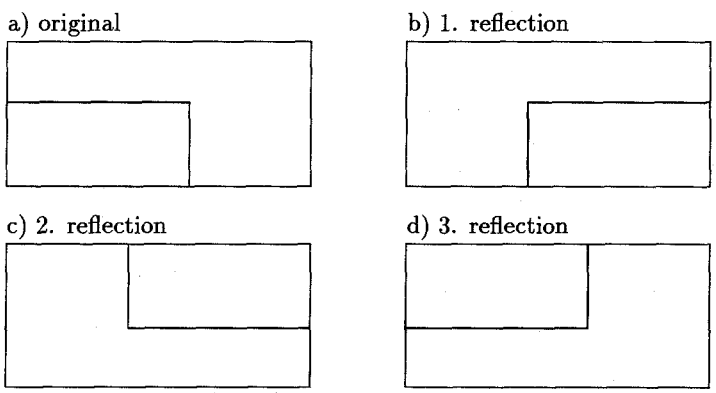



Fig. 18. Reflections of a polygon.

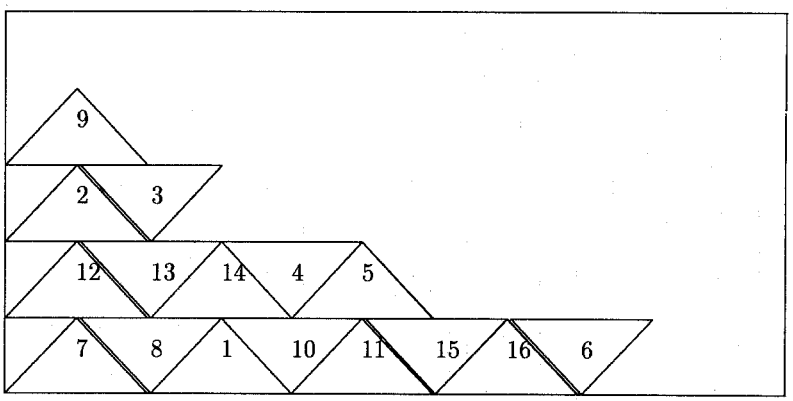



Fig. 19. Improved pattern of Fig. 16; height = 16.

cause there might be big gaps between the polygons.

The shrinking algorithm moves the polygons to one another as close as possible using the idea of the BL-heuristic. So the polygons are shifted alternately as far as possible to the bottom and to the left. More than two shifting directions are possible only with restriction of packing performance. The order in which the polygons are shifted is given by the permutation.

The following two examples cannot be improved by the shrinking step (see Figs. 16 and 17).

In order to shift the polygons closer in this case, three reflections of the polygons within the embedding rectangle are performed and shifted, too (see Fig. 18). The polygon of the four possibilities, which can be shifted over the greatest distance, replace the original. So the improved examples are shown in Fig. 19 and Fig. 20.

In the implementation, the shifting of the polygons is done with some fixed increment. After each increment it has to be tested whether overlaps are generated. This is done by the *Polygon-Cut-Algorithm* (PCA).

Given two polygons $p_i$, $p_j$, the PCA compares each edge of $p_i$ with each edge of $p_j$ for a point of intersection. The latter can be done by solving a linear system of two equations (parallel edges have to be treated as a special case).

In [14] and [21] other methods of calculating

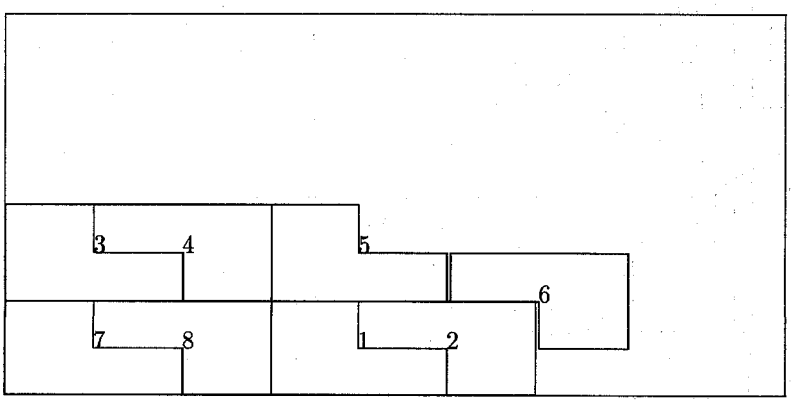



Fig. 20. Improved pattern of Fig. 17; height = 10.

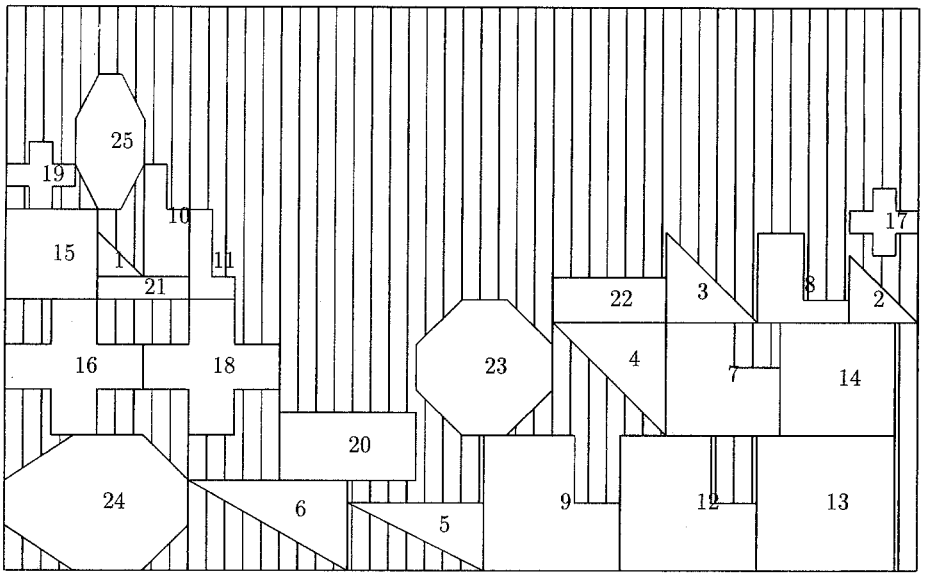



Fig. 21. Packing pattern generated by the BL-algorithm; height = 22.

points of intersection of two polygons are given. These are effective under additional geometric restrictions (e.g. convexity).

### 5.1. Numerical examples

The first example is based on 25 polygons randomly created. After the embedding the BL-algorithm creates the packing pattern shown in Fig. 21. The reason that big gaps exist in Fig. 21 comes from the fact that the embedding rectangles are left out.

The GA working on the embedding rectangles has the same parameters as in Section 4 about rectangles. Representing a packing pattern created after 1000 steps, we obtain Fig. 22.

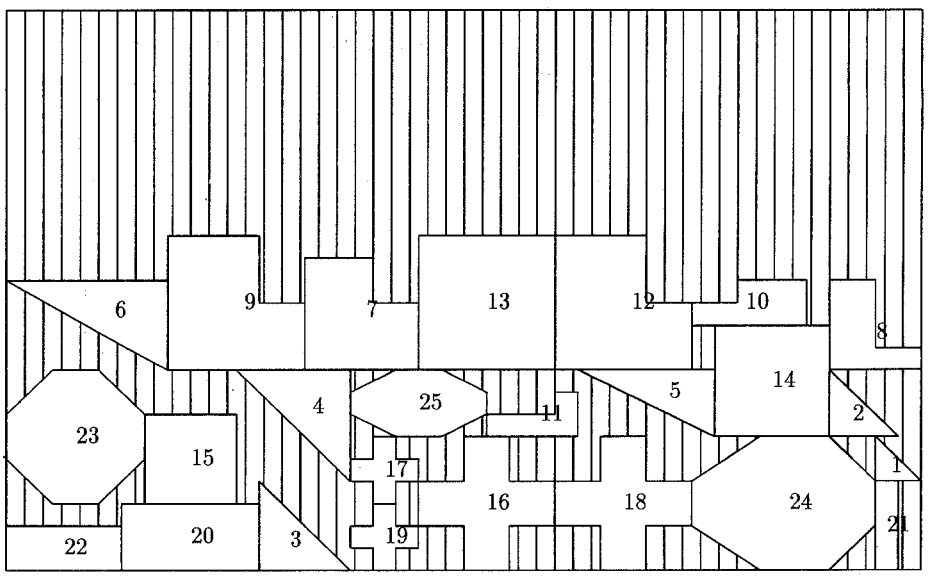



Fig. 22. Packing pattern generated by GA (1000 steps); height = 15.

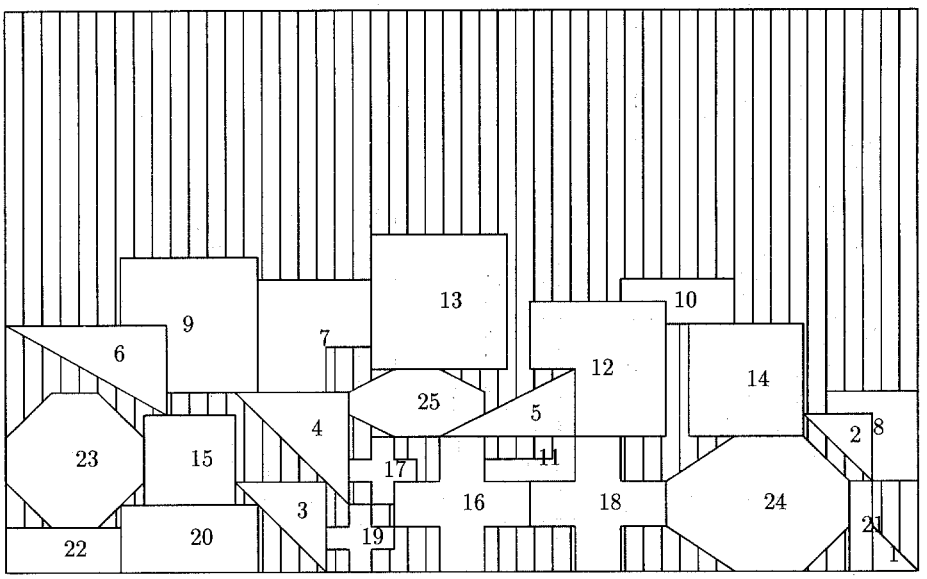



Fig. 23. Packing pattern generated by the shrinking algorithm; height = 15.

Although the GA decreases the height and improves the fitness a lot of big gaps exist after deleting the embedding rectangles.

Now a shrinking step is needed. The result of the shrinking algorithm is shown in Fig. 23. Obviously the fitness of the packing pattern is improved and the gaps get smaller. Rectangle 14 in Fig. 23 could be shifted further to the left. The gap between rectangle 14 and polygon 12 is based on the fact that rectangle 14 is shifted earlier. A further shrinking step is necessary to close the gap. A further example with 25 polygons is pre-

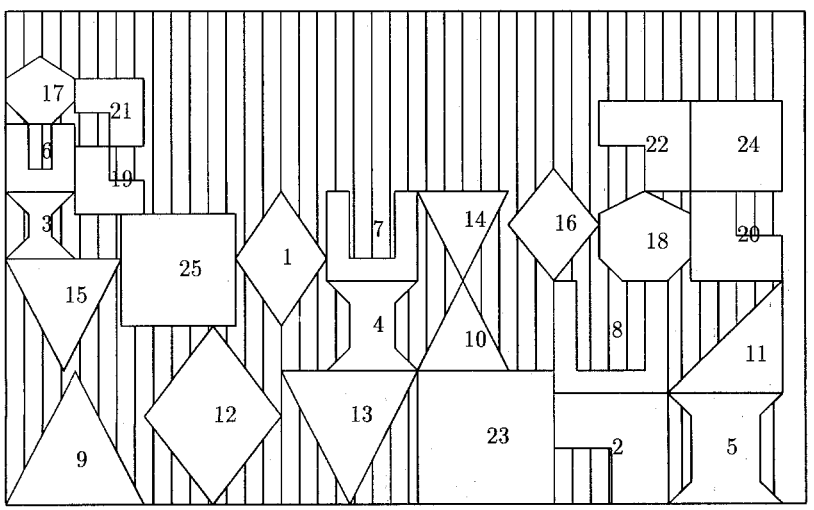



Fig. 24. Packing pattern generated by the BL-algorithm; height = 20.

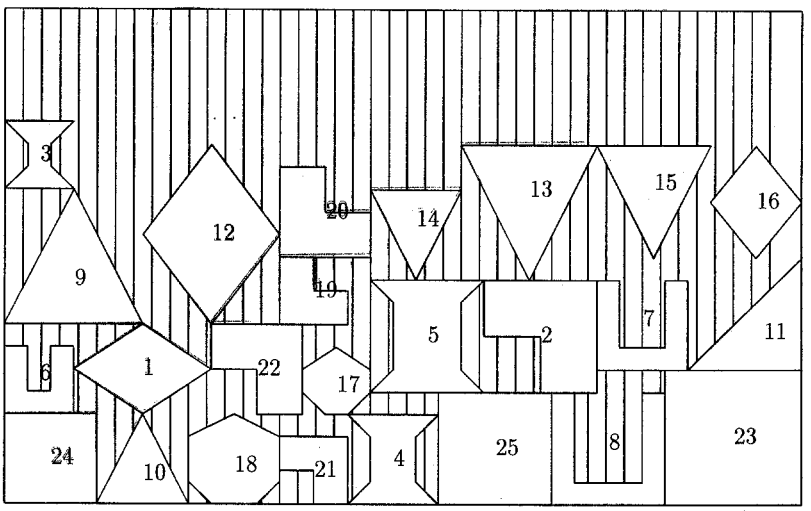



Fig. 25. Packing pattern generated by GA (1000 steps); height = 17.

sented below. The results of the steps according to the former ones are shown in Figs. 24–26.

## 6. Improvements and conclusions

We addressed the problem of improving deterministic packing algorithms. In practice, the combination of deterministic and genetic algorithms provides a possible escape out of local minima. A further advantage is the easy implementation of the combination.

Any deterministic packing algorithm based on permutation could be improved by the genetic algorithm presented here. The improvement of the BL-algorithm is the first step in this direction.

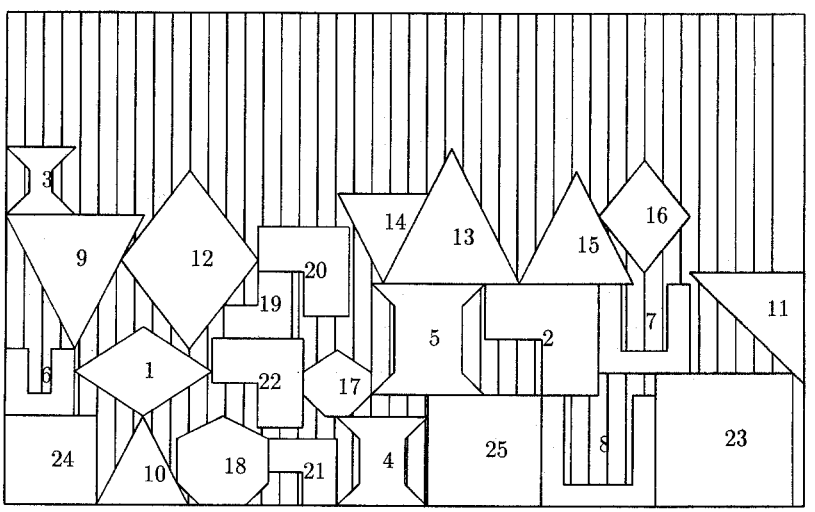



Fig. 26. Packing pattern generated by the shrinking algorithm; height = 16.

More improvements described below are possible.

Instead of embedding one polygon into one rectangle, the embedding of several polygons into one rectangle could be done. The computation of optimal clusters of polygons for embedding is a problem of optimization which is too extensive to be presented here. For more details see [10] and [11].

The disadvantage of the BL-algorithm is that groups of rectangles exist for which the BL-algorithm cannot generate the optimal packing pattern [3], thus making it necessary to use a greater and more expensive deterministic algorithm to transform a permutation into a packing pattern.

Finally, the development of a mature mathematical foundation in genetic algorithms would be a very interesting and exciting direction for future research.

## Acknowledgements

I would like to thank H.Th. Jongen and I. Schiermeyer for their helpful suggestions. The author would also like to thank the anonymous referees for their constructive comments.

## References

[1] Adamowicz, M., and Albano, A., "Nesting two-dimensional shapes in rectangular modules", *Computer Aided Design* 8, (1976) 27–33.

[2] Baker, B.S., Coffman, E.G., and Rivest, R.L., "Orthogonal packings in two dimensions", *SIAM Journal on Computing* 9/4 (1980) 846–855.

[3] Brown, D.J., "An improved BL lower bound", *Information Processing Letters* 11/1 (1980) 37–39.

[4] Coffman, Jr., E.G., Garey, M.R., and Johnson, D.S., "Approximation algorithms for bin-packing – An updated survey", *Approximation Algorithms for Computer System Design*, 1984, 49–106.

[5] Fogel, L.J., Owens, A.J., and Walsh, M.J., *Artificial Intelligence through Simulated Evolution*, Wiley, New York, 1966.

[6] Fogel, D.B., *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling*, Ginn, Needham Heights, MA, 1991.

[7] Fowler, R.J., Paterson, M.S., and Tanimoto, St. L., "Optimal packing and covering in the plane are NP-complete", *Information Processing Letters* 12/3 (1981) 133–137.

[8] Garey, M.R., and Johnson, D.S., *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.

[9] Goldberg, D.E., *Genetic Algorithms*, Addison-Wesley, Reading, MA, 1989.

[10] Haims, M., "On the optimum two-dimensional allocation problem", Ph.D. Dissertation, Dep. of Elec. Engrg., New York University, 1966.

[11] Haims, M., and Freeman, H., "A multistage solution of the template layout problem", *IEEE Transactions on Systems Science and Cybernetics* 6 (1970).

[12] Holland, J., *Adaptation in Natural and Artificial Systems*, Michigan Press, 1975.

[13] De Jong, K., "An analysis of the behavior of a class of genetic adaptive systems", Doctoral Dissertation, University of Michigan, 1975.

[14] Mehlhorn, K., *Data Structures and Algorithms 3*, EATCS Monographs on Theoretical Computer Science, Springer-Verlag, Berlin, 1986, 88.

[15] Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin, 1992.

[16] Nelißen, J., "Die Optimierung zweidimensionaler Zuschnittprobleme", Schriften zur Informatik und Angewandten Mathematik Nr. 150, RWTH Aachen, 1991.

[17] Rechenberg, I., "Cybernetic solution path of an experimental problem", Roy. Aircr. Establ., Libr. Transl. 1122, Hants, Farnborough, 1965.

[18] Rechenberg, I., *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog, Stuttgart, 1973.

[19] Schwefel, H.-P., *Numerische Optimierung von Computer-Modellen mittels der Evolutions-strategie*, Volume 26 of Interdisciplinary Systems Research, Birkhäuser, Basel, 1977.

[20] Schwefel, H.-P., *Numerical Optimization of Computer Models*, Wiley, Chichester, 1981.

[21] Shamos, M.I., and Hoey, D., "Geometric intersection problems", in: *Proc. 17th IEEE Annual Symposium Foundation of Computer Science*, 1976, 208–215.

[22] Sleator, D.D.K.D.B., "Times optimal algorithm for packing in two dimensions", *Information Processing Letters* 10/1 (1980) 37–40.

[23] Terno, J., Lindemann, R., and Scheithauer, G., *Zuschnittprobleme und ihre Praktische Lösung*, Harri Deutsch-Verlag, 1987.