European Journal of Operational Research 128/1, 34-57, 2000.

An Empirical Investigation of Meta-heuristic and Heuristic Algorithms for a 2D Packing Problem

E. Hopper and B. C. H. Turton

School of Engineering, Cardiff University, The Parade, PO Box 689, Cardiff CF2 3TF, UK
HopperE@gmx.net, Turton@cf.ac.uk

Abstract

In this paper we consider the two-dimensional rectangular packing problem, where a fixed set of items have to be allocated on a single object. Two heuristics, which belong to the class of packing procedures that preserve bottom-left stability, are hybridised with three meta-heuristic algorithms (genetic algorithms, simulated annealing, naïve evolution) and local search heuristic (hill-climbing). This study compares the hybrid algorithms in terms of solution quality and computation time on a number of packing problems of different size. In order to show the effectiveness of the design of the different algorithms, their performance is compared to random search and heuristic packing routines.

Keywords: cutting; packing; genetic algorithms; simulated annealing; optimisation;

1 Introduction

Cutting and packing problems are encountered in many industries, with different industries incorporating different constraints and objectives. The wood-, glass- and paper industry are mainly concerned with the cutting of regular figures, whereas in the ship building, textile and leather industry irregular, arbitrary shaped items are to be packed.

Packing problems are optimisation problems that are concerned with finding a *good* arrangement of multiple *items* in larger containing regions (*objects*). The usual objective of the allocation process is to maximise the material utilisation and hence to minimise the "wasted" area. This is of particular interest to industries involved with mass-production as small improvements in the layout can result in savings of material and a considerable reduction in production costs.

Our work is concerned with a two-dimensional rectangular packing problem. The problem consists of packing a collection of items onto a rectangular object while minimising the used object space. The packing process has to ensure that there is no overlap between the items. The

specific problem that is addressed in this paper has the following characteristics:

- a set of items, which may contain identical items
- one single object of fixed width and infinite height
- all pieces are of rectangular shape
- items can be rotated by 90°
- non-guillotineable (Dyckhoff, 1990)

The two dimensional stock cutting problem occurs as an industrial problem where of a number of rectangular items need to be cut from a roll of material. Since the order (set of items) is small compared to the stock material, the height of the object can be regarded as infinite. Processor allocation can also be treated as a two-dimensional packing problem (Hwang, 1997).

2 A Review of Meta-heuristic Packing Algorithms

Many heuristic packing algorithms have been suggested in the literature. Surveys on solution methodologies for various types of the two-dimensional rectangle packing problem can be found in Hinxman (1980), Sarin (1983) and Hässler and Sweeney (1991). In comparison to the great quantity of literature on heuristic algorithms to the packing problem, only a few researchers have experimented with meta-heuristic algorithms.

2.1 Packing and Genetic Algorithms

Genetic algorithms for packing problems mainly concentrate on guillotineable layouts as found in the wood, glass and paper industry (Kröger, 1995; András, 1996) and one-dimensional bin-packing (Falkenauer and Delachambre, 1992). With respect to the specific packing problem described in section 1 three types of solution approaches involving genetic algorithms can be distinguished.

The majority of literature concentrates on hybrid algorithms, where a genetic algorithm is combined with a heuristic placement routine. In this two-stage approach a genetic algorithm is used to determine the sequence, in which the items are to be packed. A second algorithm is then needed, which describes how this sequence is allocated onto the object. One of the first researchers who implemented genetic algorithms in the domain of packing is Smith (1985). He experimented with two heuristic packing routines, one of which implements backtracking and produces denser layouts, however, is computationally more expensive. Comparisons between the two hybrid approaches show that the combination with the more sophisticated heuristic generates better packing patterns. The packing problem Smith studied is special in that the orientation of the rectangles is fixed.

In the hybrid approach by Hwang et al. (1994) a genetic algorithm is combined with a well-

known heuristic from bin packing, the so-called First-Fit-Decreasing-Height algorithm (FFDH). Although this technique produces guillotineable layouts, this is not treated as an additional constraint in this work and suggested for the general case of the rectangle packing problem. Comparisons with another GA technique, which will be described below, and the FFDH heuristic itself show that this hybrid technique performs best.

Jakobs (1996) uses a heuristic, which belongs to the class of bottom-left packing heuristics to hybridise an order-based genetic algorithm. In order to reduce computational complexity the heuristic does not necessarily place an item at the lowest available bottom-left position, however, preserves bottom-left stability in the layout (see section 3.1). The method allows high quality layouts to be generated.

Lai and Chan (1997) use an evolutionary algorithm, which is combined with a heuristic routine. This routine is similar to the BL-heuristic and places items in the position that is closest to the lower-left corner of the object. Comparisons with a mathematical programming algorithm show that the evolutionary approach is computationally more efficient, however, it generates patterns with slightly higher trim loss.

Dagli and Poshyanonda (1997) also used the genetic algorithm to generate an input sequence for the placement algorithm, which is based on a sliding method combined with an artificial neural network. Every incoming item is placed next to the partial layout and all scrap areas generated are recorded. If there is a match between an incoming item and one of the scrap areas, the neural network selects the best match.

A second category of solution approaches with genetic algorithms aims at incorporating some of the layout information into the data structure of the genetic algorithm. However, some additional rules are still needed to fix the position in the layout.

The genetic algorithm by Kröger et al. (1991) is based on a directed binary tree to encode the problem. This representation fixes one dimension of the position of an item in the layout. The second dimension is determined by the bottom-left condition. Since its performance is compared to well-known packing heuristics, a relative comparison with our work is possible.

Hwang et al. (1994) also use a directed binary tree, which combines two rectangles to a larger rectangle by either placing them horizontally or vertically next to each other. The position within the containing larger rectangle is left justified. As mentioned before, comparisons with a hybrid GA technique show that this method is less efficient in terms of packing height.

The third group of GA solution approaches attempts to solve the problem in 2D space. Herbert and Dowsland (1996) developed a two-dimensional coding technique for a pallet-loading problem of identical rectangles. The layout is represented by 2D matrix indicating available positions for vertical and horizontal placement, whereby the horizontal one has

priority. This technique works well for small problems. In order to improve the outcome for medium sized problems additional repair and enhance operators have been introduced.

The method developed by Ratanapan and Dagli (1997) is different from the other approaches described so far, since it does not make use of a data structure to represent the problem. The items are represented as two-dimensional pieces with their true geometric dimensions. After the initialisation process, which places all items into non-overlapping positions on the object, a series of genetic operators is applied, which consist of moving, relocation and recombination operations.

2.2 Packing and Simulated Annealing

Only few researchers have applied simulated annealing to 2D rectangular packing problems. Kämpke (1988) applies simulated annealing to one-dimensional bin packing. Dowsland (1993) has experimented with simulated annealing on pallet loading problems involving identical as well as non-identical boxes. In the identical case the number of feasible positions for the placement of one item is reduced to the co-ordinates, which are a multiple of its length and width away from the container edge. The neighbourhood has been defined as the set of solutions, which is obtained, when each item is moved to any other position with some restrictions. Since these movements lead to overlapping patterns, this constraint has been dealt with in the objective function. Extending this method to non-identical pieces, the condition for the feasible positions is that it has to be at a valid combination of lengths and widths of the other item types from the container edge.

2.3 Comparison of Meta-Heuristic Methods

As can be seen from the literature overview given above genetic algorithms and simulated annealing have been successfully applied to the two-dimensional rectangle packing problem. However, none of the researchers has compared the performance of these meta-heuristic algorithms using the same packing problems. Burke and Kendall (1998) have carried out the only research in this area on the clustering of rectangles. Their findings indicate that tabu search and simulated annealing outperform genetic algorithms for this specific problem.

In this paper our main objective is to compare the performance of genetic algorithms, naïve evolution and simulated annealing with each other for small to large packing problems. For all methods a two-stage approach has been chosen, where the meta-heuristic algorithm is combined with a heuristic packing policy. As far as we are aware simulated annealing has not been implemented in a two-stage approach for packing problems such as genetic algorithms (see section 2.1).

Two different heuristic routines are used to hybridise the meta-heuristic algorithms. The

first technique is the BL heuristic, which has been used by Jakobs (1996) in a hybrid genetic algorithm. The second heuristic algorithm also creates bottom-left justified patterns, however, is more sophisticated and hence computationally more expensive (Chazelle, 1983). The outcome of meta-heuristic algorithms is compared with the two packing heuristics as well as with other search techniques such as hill-climbing and random search. In order to show the effectiveness of the design of the genetic algorithms their performance is compared with a naïve evolution algorithm. Results indicate that the performance of the hybrid algorithms is strongly dependent on the nature of the placement routine and the problem size. In particular in industrial applications not only the solution quality has to be considered, but also the computational cost of the various packing methods.

This paper is organised as follows: section 3 gives a brief introduction into the heuristic and meta-heuristic algorithms used in our research. In section 4 the experiments and the test problems are outlined. Section 5 and 6 contain an overview and a discussion of the results and section 7 summarises the findings of this research.

3 Heuristic Placement Algorithms

In this paper we turn our attention to the class of bottom-left heuristics (Baker et al., 1980). These packing procedures preserve bottom-left stability in the layout. An item is allocated in a bottom-left stable position if it cannot be moved any further to the left or downwards. Two implementations of a bottom-left heuristic are combined in our study with meta-heuristic algorithms.

3.1 Bottom Left Algorithm (BL)

The BL algorithm described below has been used by Jakobs (1996) in a hybrid genetic algorithm. Starting from the top-right corner each item is slid as far as possible to the bottom and then as far as possible to the left of the object. These successive vertical and horizontal movement operations are repeated until the item locks in a stable position. A valid position is found when the rectangle collides with the partial layout at its lower and left sides. Figure 1 shows the placement of a sequence of rectangles, which is described by the permutation (2, 6, 4, 3, 0, 1, 5).

Figure 1: BL routine (Jakobs, 1996)

The major disadvantage of this routine consists of the creation of empty areas in the layout, when larger items block the movement of successive ones. On the other hand its time complexity is only $O(N^2)$, when N is the number of items to be packed. Due its low complexity this heuristic is favourable in a hybrid combination with a meta-heuristic, since the decoding

routine has to be executed every time the quality of a solution is evaluated and hence contributes to a great extent to the run time of the hybrid algorithm.

3.2 BLF-Algorithm

Since the BL-routine described above tends to generate layouts with relatively large empty areas, a second more sophisticated bottom-left heuristic has been considered for hybridisation with meta-heuristics. The strategy here consists of placing a rectangle into the lowest available position of the object and left-justifying it. Figure 2 demonstrates the placement policy using the same permutation example as in Figure 1.

Figure 2: Bottom-Left-Fill heuristic

Since the generation of the layout is based on the allocation of the lowest sufficiently large region in the partial layout rather than on a series of bottom-left moves, it is capable of filling existing gaps in packing pattern. In order to distinguish it from the BL-algorithm described in section 3.1 it is referred to as the Bottom-Left-Fill (BLF) heuristic. Compared to the BL-routine this method results in denser packing patterns. The major disadvantage, however, lies in its time complexity, which is $O(N^3)$ (Chazelle, 1983).

4 Heuristic Search Techniques

The quality of the layout which is constructed using the above placement algorithms depends on the sequence in which the rectangles are presented to the routine. Since the number of combinations is too large to be explored exhaustively in a reasonable amount of time, meta-heuristic algorithms are used as a more efficient search strategy. In the following hybrid approaches the task of the meta-heuristic is to search for a good ordering of the items. A placement routine is then needed to interpret the permutation and evaluate its quality. Heuristic search techniques will result in a good, however, not necessarily optimal solution within reasonable computing time.

4.1 Hill-Climbing

Hill-climbing is a local search technique, which moves from one solution to another one in the neighbourhood. If the quality of the new solution is better than the previous one, this move is accepted and the search continues from here. If the neighbouring state does not result in an improvement, the move is rejected and the search continues from the current state. The main disadvantage of this method is that the search process might get trapped in a local minimum, which is not equal to the global one. A useful variation on simple hill-climbing considers a series of moves from the current state and selects the best one as the next state. This method is known as gradient search or steepest-ascent hill-climbing.

4.2 Meta-Heuristic Algorithms

In order to overcome the main disadvantage of local search algorithms such as hill-climbing, whose weakness lies in the inability to escape from local minima, more sophisticated heuristic search strategies are designed to avoid such a situation. This implies the temporary acceptance of a state of lower quality. Hence meta-heuristic algorithms can be considered to some extent as local search strategies, however, they include a means to escape from local minima.

Genetic Algorithms

Genetic Algorithms are search and optimisation procedures that operate in a similar way to the evolutionary processes observed in nature. The search is guided towards improvement using the 'survival of the fittest principle'. This is achieved by extracting the most desirable features from a generation of solutions and combining them to form the next generation. The quality of each solution is evaluated and the 'fitter' individuals are selected for the reproduction process. Continuation of this process through a number of generations will result in optimal or near-optimal solutions. The main difference between genetic algorithms and other meta-heuristic approaches such as simulated annealing and tabu search is that they deal with populations of solutions rather than a single solution and therefore explore the neighbourhood of the whole population. Operators such as selection, crossover and mutation are used to explore the neighbourhood and generate a new generation. Further theoretical and practical details can be found in (Davies, 1991; Goldberg, 1989).

Naïve Evolution Algorithm

The basic idea behind naïve evolution is the same as for the genetic algorithm. However, no crossover operator is applied to manipulate the search space. Only the mutation operator is used for the generation of the next population. A naïve evolution algorithm can be used to test the efficiency of the crossover operator in a genetic algorithm. Falkenauer (1998) applied this technique in experiments on one-dimensional bin-packing problems.

Simulated Annealing

Eglese (1990) has investigated the application of simulated annealing as a tool for Operations Research. Simulated annealing was introduced as an optimisation tool in the 1980's when the concept of physical annealing first was applied in combinatorial optimisation. Transferring this model to combinatorial problems the energy states in a system correspond to the various feasible solutions for a problem and the energy of the system to the cost function to be minimised.

Simulated annealing can be seen as a variant of the hill-climbing method, however, it attempts to avoid getting trapped in a local minimum. Instead of only accepting neighbouring

solutions that result in an improvement, also solutions, which are worse, may be accepted randomly with a certain probability. This probability depends on the increase in cost and a control parameter, i.e. temperature in physical annealing. The smaller the increase in the cost and the higher the temperature the more likely uphill moves will be accepted. During the annealing process the temperature is gradually lowered according to the cooling schedule. This means the algorithm becomes more and more selective in accepting new solutions. At the end of the process only moves, which result in an improvement are accepted in practice. The search process terminates when it arrives at a lower bound for temperature or cost.

5 Search Techniques and Test Data

5.1 Implementation of the Search Algorithms

A number of problem-specific and generic decisions have to be made for the implementation of the meta-heuristic search algorithms. The problem-specific choices concern the objective function, initial solution, representation scheme as well as the operators applied to manipulate the search space. Generic decisions include the probabilities at which the search space manipulators such as cross-over and mutation are applied, the cooling schedule in the case of SA and the population and generation sizes for the GA as well as stopping criteria for the search algorithms.

In our work the packing problem is tackled with a two-stage approach, where the meta-heuristic search methods (GA and SA) and the local search algorithm search the solution space for good permutations. The permutation represents the order, in which the finite set of items is packed.

The heuristic placement routine is then used to decode and evaluate the quality of the permutation according to the fitness function. The quality of a packing pattern is first of all determined by its height, since the unused rectangular area can be re-used. However, this variable is not sufficient to express how tightly the items are packed. For the fitness function a weighted sum has been used so the packing height is weighted at 70% and the packing density at 30%.

The search space of this problem is extended by the orientation of the items, which can rotate by 90°. In order to allow the meta-heuristic and local search algorithms to explore the orientation of the items, an operator is used which flips the orientation of each rectangle in the sequence with a certain probability.

Genetic Algorithm

Since an order-based encoding is used for this problem, care has to be taken that valid

chromosomes are generated during the crossover and mutation operations. Partially matched crossover (PMX) (Goldberg, 1989) and order-based mutation (Syswerda, 1991) are suitable for this type of encoding and have been used in this case. Proportional selection and generational replacement have been applied. The orientation of the rectangles is considered in the genetic algorithm in the form of mutation. In the case of orthogonal packing only two orientations for an item are possible. The rotation operator is applied to every item in the chromosome and changes the orientation with a certain probability.

Further techniques that have been implemented include elitism and seeding (Goldberg, 1989). Since heuristic placements with pre-ordered input sequences can yield packing patterns, whose quality can lie above average (Coffman et al., 1984), the initial population has been seeded with the permutation, which describes the rectangles sorted according to decreasing height. For the seeding the best out of 50 evaluations of the placement heuristic using randomly generated but height-sorted input sequences has been taken. The initial population has been generated randomly and contains the seeded individual. The genetic algorithms we implemented use a population size of 50 and a generation size of 1000. The probability for crossover is 60% and the one for the two types of mutation is 3%.

Naïve Evolution Algorithm

The problem-specific and generic decisions for the naïve evolution algorithm are the same as for the genetic algorithm with the difference that no crossover operator is used to manipulate the solution space.

Simulated Annealing Algorithm

The packing problem is represented by a permutation that is interpreted as the order in which the rectangles are packed. The neighbourhood structure of the current solution is defined by the set of solutions that can be reached applying the following two manipulation operations. The first one is analogous to the order-based mutation operator used in the genetic algorithm and swaps two randomly selected items in the permutation. The second operator considers only the orientation and flips the rotation variable of one randomly selected item. In the translation to the next solution only one of the operators is applied with a 50% chance. The initial solution is randomly generated.

The generic choices for the implementation of a simulated annealing algorithm are summarised in the annealing or cooling schedule. The schedule presented in Press et al. (1995) is used in this study with a few modifications. The temperature function is geometric and decreased by 10%. The initial value for the temperature has been determined as 19.23 applying the method described by in Press et al. (1995). The number of iterations at each temperature has

been modified in order to reduce the total simulation time. The temperature is held constant for 50N total moves or 5N successful moves at each step with N representing the number of items.

Hill-Climbing

The neighbourhood structure as well as the manipulators for the search space used in the local search procedure is the same as in simulated annealing. The initial solution is generated randomly. The search process is stopped after N unsuccessful moves in the search space.

Random Search

In order to demonstrate the effectiveness of the design the different meta-heuristic search techniques are compared to random search. In random search permutations are generated randomly. The search process is run over the same number of function evaluations as the genetic algorithm.

5.2 Test Problems

Performance of the meta-heuristic and heuristic algorithms has been tested with seven different sized packing tasks ranging from 17 to 197 items. Three instances have been generated for each problem category. The dimensions of the rectangles are produced randomly with a maximum aspect ratio of 7. The problems have been constructed such that the optimal solution is known (see Table 1). The ratio of the two dimensions of the object varies between 1 and 3. Three instances of each problem have been simulated. Detailed information about the various item sets is given in the Appendix.

Table 1: Test problems

problem category	number of items	optimal height	object dimensions
C1	16 or 17	20	20x20
C2	25	15	40x15
C3	28 or 29	30	60x30
C4	49	60	60x60
C5	72 or 73	90	60x90
C6	97	120	80x120
C7	196 or 197	240	160x240

5.3 Simulation

The genetic algorithm and the naïve evolution algorithm have both been simulated over 1000 generations using a population size of 50. The stopping criterion for the simulated annealing and the hill-climbing algorithm is based on the number of unsuccessful moves. Hence

both search processes have been run until termination by this criterion. In order to establish the efficiency of the optimisation processes random search has been applied over the same amount of iterations as the GA (i.e. 50000). The results presented below are the average of 10 simulations. The outcome of the meta-heuristic methods is compared on basis of the number of iterations. The heuristic methods have been run 50 times using random input sequences as well as sequences which have been sorted by decreasing height (DH) or width (DW). The simulations have been run on a PC with a Pentium Pro 200 MHz processor and 65MB of RAM under Windows NT4.0. The algorithms have been implemented in C++ using LEDA (Library of Efficient Data types and Algorithms), version 3.7.1 (Mehlhorn et al., 1998), to generate the graphic output.

6 Results

6.1 Comparison between the heuristic algorithms BL and BLF

The comparison of the two heuristic packing algorithms shows that the more sophisticated placement routine (BLF) achieves better layouts. Table 2 summarises the relative distances between the lowest packing height found and the height of the optimal solution. Using random input sequences the layouts generated by the BLF algorithm are between 10 and 30% better than the ones obtained with the BL rule.

Table 2: Relative distance of best solution to optimum height [%] for heuristic methods with and without pre-ordered input sequences

	C1	C2	C3	C4	C5	C6	C7
BL	25	39	33	33	31	34	41
BL-DH	17	68	27	21	18	19	31
BL-DW	18	31	24	18	22	21	29
BLF	14	20	17	15	11	12	10
BLF-DH	11	42	12	6	5	5	4
BLF-DW	11	12	12	5	5	5	5

Pre-ordering the input sequences according to decreasing width (DW) or height (DH) of the items improves the outcome of both packing heuristics by 5 to 10% compared to the performance on random input sequences. Comparing the best solutions achieved with both methods for each problem shows that the BLF heuristic outperforms the BL routine by up to 25% with the performance gain being higher for the larger problems.

Figure 3: Average elapsed time per 1000 iterations for heuristics

Table 3: Average elapsed time per placement of one item for heuristic methods [μs]

problem	C1	C2	C3	C4	C5	C6	C7
size	17	25	29	49	73	97	197
BL	46	61	56	95	119	158	324
BLF	72	87	114	252	470	794	3234

As already mentioned in section 3.2 the computational complexity of the BLF algorithm is higher than the one of the BL routine. The average run time increases exponentially with the problem size. This difference is particularly noticeable for larger packing problems (Figure 3). The average time needed to place one item is higher for the BLF algorithm (Table 3). The number of available bottom-left stable positions, which are tested on average, before a suitable position is found, is higher than the number of movements carried out using the sliding method of the BL algorithm. The number of free BL-positions in the layout increases exponentially with the problem size.

6.2 Comparison between the heuristic algorithms BL and BLF

In the following the performance of the meta-heuristic algorithms is investigated. First of all, the meta-heuristics methods, which use the BL decoder, are compared. Then the same comparison is made for the meta-heuristic algorithms, which are hybridised with the more sophisticated BLF routine. Finally, the performance of the two types of hybrid combinations is analysed. In order to study the efficiency of the meta-heuristic methods over the simple heuristics their outcomes have been compared to random search, which evaluates the packing routines over the same number of iterations as the GA (i.e. 50000) using random input sequences.

The meta-heuristic search methods using the BL decoder achieve layouts of higher quality than the simple packing heuristic (BL). The packing heights achieved by the hybrid are up to 24% better than the ones by the BL (Table 4). The performance of the random search (RS) lies between the one of the meta-heuristics and the packing heuristic. Hence some of the performance gain achieved by the meta-heuristics is due to the higher number of iterations. The outcomes of the two evolutionary methods (GA and NE) are very similar with the NE algorithm performing slightly better for some problems (up to 2%). Hill-climbing performs better than random search for most problems or at least equally well and outperforms the simple BL heuristic by up to 5%. For all techniques, heuristics as well as meta-heuristics, the difference between the packing heights achieved and the optimal height becomes larger with increasing

problem size.

Table 4: Relative distance of best solution to optimal height [%] for heuristic and meta-heuristic methods combined with BL routine

	C1	C2	C3	C4	C5	C6	C7
GA+BL	6	10	8	9	11	15	21
NE+BL	6	8	8	8	11	13	19
SA+BL	4	7	7	6	6	7	13
HC+BL	9	18	11	14	14	20	25
RS+BL	6	14	14	16	18	20	28
BL	17	31	24	18	18	21	29

The best layouts for the hybrids with the BL decoder have been obtained with simulated annealing in all problem categories. The difference between SA and the two evolutionary techniques (GA and NE) lies between 1 and 8% and is higher for the larger problems. However, the number of iterations needed by the SA is up to 5 times higher on average. Figure 1 shows that the SA converges very slowly, whereas the GA reaches the final packing height earlier.

Figure 4: GA and SA + BL for large problem

The results obtained by the methods using the sophisticated packing heuristic, i.e. BLF, show a similar ranking of the various methods (Table 5). For problems up to category C3 random search and hill-climbing perform better than the simple heuristic. Hill-climbing, however, is outperformed by random search, especially for the smaller problems. The packing heights achieved by the evolutionary algorithms, GA and NE, are very similar and outperform random search and hill-climbing. Simulated annealing yields the best results in each problem category.

Summarising Table 5, the meta-heuristic methods, which use the BLF decoder, achieve packing heights, which are very close to the optimum height (between 3 and 7%). Even the BLF heuristic on its own leads to very low packing heights. Especially, for larger problems (C2 to C7) the difference to the meta-heuristics is maximally 2%. This finding is different from the BL case, where the solution qualities obtained by the simple heuristic and the meta-heuristic differ particularly for the larger problems (Table 4). Using the BLF decoder the difference to the optimal height does not increase with the problem size as is the case with the BL decoder (see Table 4 and Table 5). The solution quality remains at the same level for each problem category (up to 7% from the optimum).

Table 5: Relative distance of best solution to optimum height [%] for heuristic and meta-

heuristic methods combined with BLF decoder

	C1	C2	C3	C4	C5	C6	C7
GA+BLF	4	7	5	3	4	4	5
NE+BLF	5	7	4	4	4	4	5
SA+BLF	4	6	5	3	3	3	4
HC+BLF	7	10	7	7	6	7	7
RS+BLF	5	8	7	7	6	7	7
BLF	11	16	12	5	5	5	5

The comparison of the hybrid algorithms shows that the combinations with the BLF placement routine produce better layouts than the combinations with the BL routine. The hybrids with the BLF routine generate layouts that are up to 16% better. The difference is especially high for the large problems. The same is true for the NE hybrids. In Table 6 the results obtained with the two different decoding algorithms are compared. The difference between the best solutions found using the meta-heuristic with the BL and the BLF decoder are stated.

Figure 5 demonstrates the performance of the two genetic algorithms. Both algorithms achieve the highest performance gain within the first 10000 iterations, i.e. 2000 generations.

Table 6: Difference between the best solutions of the hybrids with the BLF routine and the ones with the BL routine [%]

	C1	C2	C3	C4	C5	C6	C7
GA	2	3	3	6	7	11	16
NE	1	1	4	4	7	9	14
SA	0	1	2	3	3	3	9
HC	2	8	4	7	8	13	18
RS	1	6	7	9	12	13	21
heuristic	6	15	12	13	13	16	24

Figure 5: Comparison of the two GAs combined with BL and BLF routine

Although the meta-heuristics perform better in terms of solution quality, the combinations with the BLF decoder have longer run times (Table 7). Run times become extremely long for large problems (C5 to C7) due to its higher computational complexity. The BL algorithm offers an advantage in that respect. Especially, simulated annealing, which achieves the best layouts, has high execution times. However, further adjustments to the annealing schedule will most likely reduce the run time of this meta-heuristic.

Table 7: Average elapsed time for heuristics and meta-heuristics per run in [min]; for BL and BLF in [ms]

	C1	C2	C3	C4	C5	C6	C7
GA+BL	0.5	0.8	0.9	2.4	4.0	6.7	23
NE+BL	0.4	0.6	0.7	1.6	2.6	4.1	30
SA+BL	0.4	1.4	1.8	7.5	17	31	117
GA+BLF	1.0	2.0	3.0	13	36	86	777
NE+BLF	0.7	1.3	2.1	8.3	23	55	483
SA+BLF	0.7	2.4	4.0	33	115	382	4181
BL [ms]	0.8	1.3	1.6	4.7	8.7	15.3	63.7
BLF [ms]	1.2	2.2	3.3	12	34	77.0	636

7 Discussion

7.1 Packing Heuristics

The BLF packing algorithm achieves better packing patterns than the BL heuristic for our example problems. Since the BLF routine first attempts to fill the gaps in the layout, the majority of the small items will be 'absorbed' within the existing partial layout and does not contribute further to the packing height, which is mainly determined by the larger items (Figure 6). With the BL-rule, however, unused regions in layout cannot be accessed and smaller rectangles also contribute to the height. The results in section 6.1 show that the difference to optimum solution gets smaller with increasing problem size. This is due to the fact that larger problems contain a larger number of small items, which are allocated in the empty areas contained in the partial layout. In particular pre-ordered input sequences (height or width) achieve dense layouts (Figure 6). Although the execution time for the BLF algorithm is considerably larger, the performance gain especially for large packing problem justifies the application of the BLF routine. In a combination with a meta-heuristic, however, the time complexity plays a more important role.

Figure 6: Best layouts for a large problem (C6) with BL (left) and BLF (right); height-sorted sequence

7.2 Meta-Heuristics and Local Search

Looking at the results stated in section 6.2 it can be seen that the meta-heuristic methods outperform the hill-climbing algorithm due to their ability to escape from local minima. Hence meta-heuristics offer a clear advantage over the local search algorithm in that respect. Since the hill-climbing algorithm terminates in a local minimum its run time is shorter than that of

simulated annealing. The hill-climbing technique only allows exploration of a limited area of the search space and is outperformed by the meta-heuristic algorithms from the beginning of the search process (Figure 7). Although the final outcome of hill-climbing is slightly better than the random search method, a random walk through the search space results in better solutions during most of the search process. Whereas hill-climbing only explores the search space locally, simulated annealing can exploit the space more effectively and concentrates on promising areas.

Figure 7: Comparison between local search and meta-heuristic methods (+BL)

Due to the manipulation technique used in SA, which either only changes the rotation of one element at a time or the position of two elements in the permutation, the search process with the SA is slow at the beginning and even random search results in better layouts. Random search, however, is quickly outperformed, when the SA algorithm starts exploiting promising areas in the solution space and finds solutions which would not have been found on a random basis.

Figure 8: Evolutionary algorithms (GA and NE) and random search for BL for C3

Looking at Figure 7 the most successful search strategy in the beginning of the search process are genetic algorithms. Solutions found by the GA improve rapidly over the number of evaluations and only get outperformed by the SA towards the end of the search (Figure 4). Hence the technique which GAs use to explore the space is more successful than the one used by SA. One of the differences between the algorithms is that a crossover operator is used in the GA to manipulate the current best solutions. This obviously creates larger changes in the sequences than with the mutation operator used in SA and hence could explain the rapid progress the GA makes in the beginning.

Comparisons with a naïve evolution algorithm (NE), which is only based on mutation and has no crossover, show, however, that both strategies are equally successful (Figure 8). Hence the crossover operator used in this implementation is not the reason for the better exploration of the search space in the beginning of the search process. If crossover was the main contributor, then the difference between GA and NE would be higher. However, the mutation operator implemented is obviously sufficient for this task. Different from the SA method the evolutionary techniques work on a population of solutions, which they explore simultaneously. Whereas the SA operates only on one solution at a time, the recombination method in evolutionary algorithms guarantees that the most successful solutions are utilised in the following generation. Hence, they allow exploration of the solution space in parallel. The GA, however, is outperformed by the SA technique towards the end of the search process, when the population has converged. Only the SA technique, where solutions of minor quality can be accepted over a series of moves, can then lead the search into promising regions.

Figure 9: Random search and GAs for both decoders (BL and BLF)

In order to establish how well the meta-heuristic algorithms explore the search space, a random search process has been applied to the packing problems. Whereas the random search only 'explores' the solution space, the in-built search mechanisms allow the meta-heuristic strategies to 'exploit' good regions. The difference between the GA and random search for the BLF case is smaller, which indicates that the 'exploitation' of the solution space is limited (Figure 9).

The comparison between the meta-heuristics, hill-climbing and random search shows that the improvement over the BLF heuristic is largest for smaller packing problems, i.e. problems with less then 50 items (C1 to C3). Figure 11 indicates the improvement the five different methods have achieved over the best solution obtained with the BLF algorithm. For problems consisting of a higher number of items (C4 to C7) only meta-heuristic methods are successful and result in better layouts than the BLF heuristic. Random search and hill-climbing cannot explore the enormous search space sufficiently and are easily outperformed by the simple BLF heuristic. Unlike random search the simple BLF method used in this comparison also includes pre-sorted sequences, where the items are sorted according to their height or width. As it can be seen in Table 2 sorting almost always is better than random input. Since random search only stands a theoretical chance of finding a sorted sequence it can be outperformed by the simple BLF method as indicated in Figure 10. All meta-heuristic methods manage to improve the heuristic solution, however, only by a few percent. The most successful method on large problems is SA.

Figure 11: Improvement of the meta-heuristics +BLF in comparison to the best of the heuristic solutions (BLF including sorted sequences) for each problem category

7.3 Hybrid Methods

The results summarised in Table 4 and Table 5 show that the combination between the meta-heuristics and the BLF packing routine achieve the better outcomes compared to the combinations with the BL routine. The difference is higher for the larger problems (Table 6).

Figure 5 shows the performance of the two heuristics in combination with a genetic algorithm. Since the packing heights achieved with the BLF on its own are already very close to the optimum height (less than 8% from the optimum on average), the meta-heuristic cannot improve the performance of the heuristic as much as in the BL case. In other words by using the 'poorer' BL-decoder, the meta-heuristic is needed to find a good input sequence, whereas applying the better BLF-heuristic good layouts are achieved in a smaller number of iterations.

Figure 12 and Figure 13 show initial and best layouts obtained by both GAs for the largest problem.

Figure 12: Initial and best layout for GA+BL

Figure 13: Initial and best layout for GA+BLF

This questions the use of a hybrid combination between a meta-heuristic and a 'poor' decoder for this type of packing problem. In order to achieve high quality layouts it may often be sufficient to apply the BLF routine over a small number of iterations. This is especially true for larger problems (>30 items, C4 to C7), where the hybrid methods only manage to improve the heuristic solution by less than 2% (Figure 11), however, at a very high computational cost (Table 7). For smaller problems on the other hand, where the execution times are acceptable, the meta-heuristic with the BLF rule outperforms the simple heuristic by about 7%. In this case the application of a meta-heuristic algorithm offers advantages.

In order to reduce the computation time the BL heuristic could be considered in connection with the meta-heuristic. In spite of the reduction of the execution time the solution qualities achieved with this approach for large packing tasks are up to 15% worse than those obtained with the simple BLF heuristic (Figure 14). For smaller problems where combination with the BLF routine achieves better layouts, the computation time is low anyway. Summarising the hybrid approach with the 'poorer' decoder, which is more efficient in terms of computation time, cannot be justified, since it is easily outperformed by the BLF-heuristic on large problems.

Figure 14: Improvement of the meta-heuristics + BL in comparison to the best heuristic solution (BLF) for each problem category

An implementation of the BLF algorithm, which has a lower computational complexity, is of great benefit for the hybrid approach using the BLF rule. Chazelle (1983) developed an implementation of this algorithm that has a complexity of $O(N^2)$. Using a more time efficient implementation the execution times of both heuristic packing rules become comparable, and will only differ by a factor rather than an order of magnitude. This means the hybrid algorithms with the BL rule lose their major advantage over the ones using the BLF decoder.

Hybrid combinations between meta-heuristics and a heuristic packing rule as investigated in this study not only achieve high quality layouts. Their main advantage lies in simplicity of the implementation. The meta-heuristic search is hybridised with a heuristic and acts as tuner of the packing routine. The representation as a sequencing problem allows the use of well-known manipulation techniques for the search space, e.g. order-based crossover operators, rather than developing problem-specific operators that only can be used in one specific context.

On the other side it can be argued that the geometric information concerning the layout is hidden in the heuristic decoder. Hence it cannot be exploited by meta-heuristic search processes to the same extent as if a representation was used that includes more geometric information in the definition of the neighbourhood structure and chromosomes respectively.

Comparisons with the approach by Ratanapan and Dagli (1997), where the items have been represented as true geometric objects in an evolutionary search algorithm show that this does not necessarily generate better layouts. The packing densities achieved by Ratanapan and Dagli's technique for two medium-sized problems (21 and 31 items) are about 92%. Using the same data sets all of the hybrid meta-heuristic techniques (SA, GA and NE) in combination with either of the packing routines were able to outperform this result. The achieved packing densities range between 92 and 98%, with the SA obtaining the best outcomes. For the smaller problem the SA even found the optimum solution in 3 runs out of 10. Unfortunately, Ratanapan and Dagli do not state the computational effort of the rearrangements in the layout after application of the mutation operators.

Dagli and Poshyanonda (1997) developed an approach involving a hybrid between GAs and a neural network, which achieves packing densities between 95 and 97%. Applying the meta-heuristics in combination with the BLF routine densities between 95 and 96% are obtained. However, the test problem Dagli and Poshyanonda use is very special in the sense that the ratio between the width and the height of the object is very large. Even simple heuristics as the height and width sorted BLF routines achieve a packing density of 94%. Hence, the higher effort of the implementation in Dagli and Poshyanonda's approach is not reflected in a much better outcome.

The two-dimensional matrix representation Herbert and Dowsland (1996) developed for a GA for the pallet loading problem of identical boxes does not achieve better outcomes as the one-dimensional binary encoding as the authors conclude. Applying the hybrid meta heuristics combined with BL and BLF routines to the same problems shows, that they find optimal solutions more often than Herbert and Dowsland's approach for the small problems (<=16 items). For the larger problems only near-optimal solutions have been found. Only with the implementation of 'enhance' and 'improvement' operators Herbert and Dowsland manage to improve the performance for larger problems.

The GA approach developed by Kröger et al. (1991) is based on a graph structure, which allows including some geometric information in the data structure. Since the data sets used in the experiments are not published only an indirect comparison is possible. The packing heights achieved with this technique are 1 and 7% better than those obtained by the BLF heuristic and lie in the same region as our findings.

8 Conclusions

Two types of hybrid algorithms for the rectangle packing problem have been implemented consisting of a combination of a meta-heuristic algorithm (GA, NE and SA) and heuristic packing routine to allocate the items on the object. The heuristic packing routine generates the layouts in a bottom-left justified manner. Whereas one of the techniques (BL) is a time-efficient implementation based on a sliding principle, the second one (BLF) is able to fill enclosures in the layouts, however, at higher computational cost. The meta-heuristic hybrid algorithms have been tested on a number of packing problems and compared with heuristic and local search methods and also with approaches used by other researchers.

In terms of solution quality the meta-heuristic algorithms outperform the heuristic packing routines and the hill-climbing approach with SA performing best. The combinations with the more sophisticated heuristic (BLF) achieve better layouts than the ones using the BL decoder. For industrial problems the question, which technique in combination with the BLF routing to choose, is a trade-off between material cost and simulation cost. In our study SA has achieved the best layout quality over all problem categories. Its execution time, however, becomes larger with increasing problem size. The evolutionary algorithms, GA and NE, are better in terms of execution time and yield results, which are slightly worse than ones obtained by the SA. Hence if the time for solving a packing task is limited, GA and NE are appropriate. For very small time margins, only heuristic packing algorithms will be able to meet this criterion. Table 8 summarises which methods are appropriate for different sized packing tasks under limited execution time. The information in Table 8 has to be seen in connection with the computational power of the simulation equipment used. With the processing power constantly increasing, it will be possible to apply meta-heuristics efficiently even in larger problems in the future.

Table 8: Method for best results within specified time limit (hybrids are with BLF decoder)

	<1min	<10min	< 1h	< 10h	< 24h
C1	GA, SA	GA, SA	GA, SA	GA, SA	GA, SA
C2	BLF	SA	SA	SA	SA
C3	BLF	NE	NE	NE	NE
C4	BLF	NE	SA	SA	SA
C5	BLF	GA, NE	GA, NE	SA	SA
C6	BLF	BLF	NE	SA	SA
C7	BLF	BLF	BLF	NE	NE

Since the performance difference between the hybrid methods using the BL decoder and the one using the BLF decoder is only due to the improved heuristic, the decoder has a larger effect

on the outcome of the hybrid technique than the meta-heuristic technique itself. This seems to suggest approaches, where more layout specific knowledge is incorporated in the meta-heuristic rather than the decoder. However, representation schemes studied by other researchers that use more layout information did not necessarily achieve higher packing densities than hybrid techniques for the problems tested.

Concerning the methodology, hybrid algorithms are well suited for industrial demands. The layouts achieved are of similar quality as other techniques. The implementation of the hybrid algorithm is easier, since it is based well-known techniques and does not require development of problem specific algorithms. With heuristics already being applied in industry, the acceptance of research methods such as meta-heuristics certainly will be higher.

References

András P., András, A. and Zsuzsa, S., 1996. A genetic solution for the cutting stock problem. In: Chawdry P. K., Roy R. and Kant R. K. (Eds.), Proceedings of the First On-line Workshop on Soft Computing, Springer, Berlin, pp. 87-92.

Baker B. S., Coffman E. G., Jr. and Rivest R. L., 1980. Orthogonal packing in two dimensions. SIAM Journal of Computing 9, 846-855.

Burke E. and Kendall G., 1998. Comparison of Meta-Heuristic Algorithms for Clustering Rectangles. Proceedings of the 24th International Conference on Computers and Industrial Engineering, Uxbridge, UK, (to appear).

Chazelle B., 1983. The Bottom-Left Bin-Packing Heuristic: An Efficient Implementation. IEEE Transactions on Computers c32/8, 697-707.

Coffman, E. G., Garey, M. R. and Johnson, D. S., 1984. Approximation algorithms for bin-packing - an updated survey. In: Ausiello, Algorithms Design for Computer Systems Design, Springer, Vienna, pp. 49-106.

Dagli C. H. and Poshyanonda P., 1997. New approaches to nesting rectangular patterns. Journal of Intelligent Manufacturing 8, 177-190.

Davis L., 1991. Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York.

Dowsland, K.A., 1993. Some experiments with simulated annealing techniques for packing problems. European Journal of Operational Research 68, 389-399.

Dyckhoff H., 1990. Typology of cutting and packing problems. European Journal of Operational Research 44, 145-159.

Falkenauer E., 1998. Genetic algorithms and grouping problems, John Wiley & Sons, Chichester.

Falkenauer E. and Delchambre A., 1992. A genetic algorithm for bin-packing and line balancing. Proceedings of the 1992 IEEE International Conference on Robotics and Automation, 1992, Vol.2, IEEE, Piscataway, IEEE Service Center, NJ, USA , pp.1186-1192

Goldberg D. E., 1989. Genetic Algorithms in Search, Optimisation and Machine Learning, Addison-Wesley Publishing Company, Reading.

Hässler R. W. and Sweeney, P. E., 1991. Cutting stock problems and solution procedures. European Journal of Operational Research 54, 141-150.

Herbert E.A. and Dowsland K.A., 1996. A family of genetic algorithms for the pallet loading problem. Annals of Operations Research 63, 415-436.

Hinxman A. I., 1980. The trim loss and assortment problems. European Journal of Operational Research 88, 1, 8-18.

Hopper E. and Turton B. C. H., 1998. A Genetic Algorithm for a 2D Industrial Packing Problem. Proceedings of the 24th International Conference on Computers and Industrial Engineering, Uxbridge, UK, (to appear).

Hopper E. and Turton B. C. H., 1997. Application of Genetic Algorithms to Packing Problems - A Review. In: Chawdry, P. K., Roy, R. and Kant, R. K. (Eds.), Proceedings of the 2nd On-line World Conference on Soft Computing in Engineering Design and Manufacturing, Springer Verlag, London, pp. 279-288.

Hwang I., 1997. An efficient processor allocation algorithm using two-dimensional packing. Journal of Parallel and Distributed Computing 42, 75-81.

Hwang S. M., Cheng Y. K. and Horng J. T., 1994. On solving rectangle bin packing problems using genetic algorithms. Proceedings of the 1994 IEEE International Conference on Systems, Man and Cybernetics, IEEE, Piscataway, NJ, USA, pp. 1583-1590.

Jakobs S., 1996. On genetic algorithms for the packing of polygons. European Journal of Operational Research 88, 165-181.

Kröger B., Schwenderling P. and Vornberger O., 1991. Parallel genetic packing of rectangles. In: Schwefel H. P. and Männer R. (Eds.), Parallel Problem Solving from Nature 1st Workshop, Springer Verlag, Berlin, pp. 160-164.

Kröger B., 1995. Guillontineable bin-packing: A genetic approach. European Journal of Operational Research 84, 645-661.

Lai K. K. and Chan W. M., 1997. An evolutionary algorithm for the rectangular cutting stock problem. International Journal of Industrial Engineering 4, 130-139.

Mehlhorn, K., Näher S. and Uhrig C., 1998. LEDA - Library of Efficient Data types and Algorithms, Max-Planck-Institut für Informatik, Saarbrücken, http://www.mpi-sb.mpg.de/LEDA/.

Press W. H., Teuckolsky S. A., Vetterling W. T. and Flannery B. P., 1995. Numerical Recipes in C, The Art Scientific Computing, 2nd edition, Cambridge University Press, Cambridge.

Ratanapan K. and Dagli C. H., 1997. An object-based evolutionary algorithm for solving rectangular piece nesting problems. Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, IEEE, Piscataway, NJ, USA, pp. 989-994.

Sarin S. C., 1983. Two-dimensional stock cutting problems and solution methodologies. ASME Transactions, Journal of Engineering for Industry 104, 155-160.

Smith D., 1985. Bin-packing with adaptive search. In: Grefenstette, J. (Ed.), Proceedings of an International Conference on Genetic Algorithms and their Applications, Lawrence Erlbaum, pp. 202-206.

Syswerda D., 1991. Schedule optimisation using genetic algorithms. In: Davis, L. (Ed.), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, pp. 332-349.

Appendix

The packing tasks are classified by the problem size, i.e. the number of items. For each category three problems instances have been constructed. The dimensions of the items are given in the following tables. The optimum solution for each problem is known and is achieved by packing the rectangles in the order they are stated in the tables using the BLF routine.

Category 1:

16 or 17 items;

object: 20 x 20

P1		P2		P3	
w	h	w	h	W	h
2	12	4	1	4	14
7	12	4	5	5	2
8	6	9	4	2	2
3	6	3	5	9	7
3	5	3	9	5	5
5	5	1	4	2	5
3	12	5	3	7	7
3	7	4	1	3	5
5	7	5	5	6	5
2	6	7	2	3	2
3	2	9	3	6	2
4	2	3	13	4	6
3	4	2	8	6	3
4	4	15	4	10	3
9	2	5	4	6	3
11	2	10	6	10	3
		7	2		

Category 2:

25 items,

object: 40 x 15

P1		P2		P3	
w	h	w	h	w	h
11	3	11	2	12	7
13	3	2	3	7	7
9	2	10	7	7	1
7	2	8	4	5	1
9	3	9	5	3	2
7	3	7	2	6	2
11	2	4	1	7	2
13	2	6	1	5	2
11	4	4	5	3	1
13	4	8	3	6	1
3	5	1	3	12	6
11	2	5	5	9	6
2	2	3	1	12	2
11	3	12	4	7	2
2	3	6	2	10	3
5	4	2	4	4	1
6	4	11	4	5	1
12	2	10	2	16	3
1	2	3	2	5	3
3	5	11	2	4	2
13	5	3	4	5	2
12	4	26	4	10	3
1	4	8	4	9	3
5	2	3	2	16	3
6	2	6	2	5	3

Category 3:

28 or 29 items,

object: 60 x 30

P1		P2		P3	
w	h	w	h	w	h
7	5	18	6	24	9
14	5	12	2	8	9
14	8	7	10	11	9
4	8	23	4	17	9
21	13	1	4	24	4
7	11	7	7	8	4
14	11	4	11	6	1
14	5	5	6	5	1
4	5	7	2	17	4
18	3	11	6	6	3
21	3	19	10	5	3
17	11	5	11	5	12
4	11	2	4	13	12
7	4	5	7	14	14
5	4	2	4	14	2
6	7	12	7	2	2
18	5	13	7	3	8
3	5	6	3	9	8
7	3	10	6	14	12
5	3	16	9	2	12
18	4	4	1	3	6
3	4	10	4	9	6
12	2	24	6	5	2
6	2	9	9	13	2
18	5	1	2	18	3
21	5	5	8	14	3
17	3	5	3	16	3
4	3	25	7	12	3
		21	5		

Category 4:

49 items; object: 60 x 60

P1				P2				P3			
w	h	w	h	W	h	w	h	w	h	w	h
2	7	3	3	10	14	2	4	10	4	12	7
24	7	8	3	3	13	2	7	12	4	9	4
16	4	5	20	28	5	3	4	13	5	4	4
18	4	3	17	5	8	5	30	3	5	9	9
16	7	3	7	14	9	5	3	7	22	2	5
18	7	5	7	12	14	10	26	6	22	20	5
2	4	3	7	13	10	6	5	9	23	9	5
24	4	4	7	3	17	4	9	10	19	4	5
4	28	4	21	1	5	1	4	3	15	4	2
6	18	10	19	4	1	9	2	5	13	12	2
14	12	4	17	18	4	4	17	2	10	3	15
2	12	8	17	1	1	5	2	2	10	21	11
18	19	3	10	2	6	4	4	13	18	11	3
9	8	5	10	4	14	6	2	3	18	3	3
7	8	7	6	3	18	4	10	2	3	11	23
9	11	8	6	4	14	2	4	2	3	11	23
7	11	15	12	8	17	3	12	5	2	11	8
14	6	3	12	11	5	6	5	4	2	3	8
2	6	11	10	9	12	3	9	3	4	21	4
6	10	5	10	4	7	7	18	9	4	14	4
16	10	4	2	25	8	6	6	7	1	3	13
3	5	8	2	7	5	18	7	6	1	35	13
4	5	10	2	24	9	13	9	2	4	11	5
8	12	12	2	9	14	25	7	20	4	11	5
3	18			12	19			4	7		

Category 5: 72 or 73 items; object: 60 x 90

P1						P2						P3					
w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h
6	34	10	2	2	3	3	5	1	2	30	10	6	37	5	11	4	2
3	13	6	6	9	6	14	3	5	1	14	19	10	15	4	5	3	5
5	13	5	6	9	6	9	27	1	1	4	26	4	7	5	5	4	5
12	10	7	14	1	6	6	24	3	3	3	3	12	7	1	3	4	24
12	10	6	14	2	6	21	7	5	20	5	23	4	18	6	3	15	12
7	6	3	16	7	5	7	10	6	23	5	20	10	8	1	4	13	12
15	6	5	16	18	5	1	2	7	2	15	4	5	8	6	4	19	7
7	25	6	14	9	3	13	19	11	21	10	6	4	25	4	2	9	7
15	25	5	14	9	3	4	17	8	7	6	3	5	25	5	2	5	4
12	21	13	14	18	9	4	13	6	15	5	2	4	8	19	25	2	4
7	16	2	3	5	6	17	3	2	1	4	2	12	8	5	9	12	5
5	16	7	3	2	6	24	10	13	14	3	1	10	10	4	9	9	22
3	21	2	11	12	2	5	4	3	14	2	3	5	10	3	6	5	1
5	21	7	11	9	2	2	2	5	26	14	3	3	4	3	6	2	1
7	5	7	6	3	8	6	1	9	14	9	2	7	4	6	13	15	12
5	5	6	6	9	8	11	9	10	3	7	8	7	10	20	13	13	12
1	4	14	33	9	10	4	26	4	13	32	6	2	10	3	18	2	5
10	4	4	12	5	3	2	1	1	3	6	2	7	15	3	18	5	5
13	6	3	12	2	3	4	7	14	11	26	5	4	18	5	16	12	17
13	12	18	16	18	3	7	38	7	10	1	2	15	18	4	16	2	12
9	12	3	12	7	3	3	2	14	12	6	5	3	18	6	11	5	12
6	23	18	12	3	2	3	1	18	3	13	3	7	18	13	4	4	5
3	7	4	4	9	2	4	2	7	4	10	3	7	5	7	4	28	5
5	7	3	4			4	6	2	7			2	5	13	7		
1	2	1	3			2	1	7	28			4	11	3	2		

Category 6: 97 items; object: 80 x 120

P1								P2								P3							
w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h
30	19	11	6	3	7	4	26	7	39	10	33	6	5	5	2	6	35	5	6	9	4	26	20
8	5	9	30	3	14	7	26	8	33	1	1	5	13	4	4	1	6	9	21	4	2	5	7
13	5	10	8	17	7	3	9	7	6	4	4	4	14	8	2	6	6	4	21	5	2	3	7
15	23	10	8	7	7	21	9	5	3	3	3	16	8	9	11	34	13	5	23	4	25	3	15
9	4	5	4	5	7	11	31	3	5	4	4	23	9	3	2	10	7	5	23	6	25	8	2
3	4	5	4	3	7	9	31	39	6	3	6	26	8	5	11	23	7	8	6	6	6	7	2
2	11	4	14	6	6	6	28	11	13	16	7	1	6	7	9	1	7	15	6	2	6	4	19
9	7	2	14	4	6	3	19	3	4	6	4	15	26	24	30	6	7	8	5	18	10	6	19
3	7	4	22	4	2	18	8	2	2	6	2	4	25	2	11	10	62	10	5	11	24	5	25
8	14	8	14	6	2	3	8	5	2	15	3	8	45	10	8	10	33	5	4	9	4	8	13
11	6	3	14	9	61	18	11	5	30	30	5	11	50	9	2	13	33	7	2	17	4	7	13
2	6	10	22	6	8	3	11	2	1	1	1	19	5	10	2	7	22	6	2	7	9	5	8
11	8	4	20	5	8	10	18	26	11	10	4	12	55	3	11	4	15	6	4	9	5	3	8
2	8	6	20	5	2	2	6	4	5	2	6	5	20	4	22	6	8	17	4	17	5	8	5
12	12	2	7	4	2	9	6	9	2	6	23	4	13	6	9	24	8	7	2	6	4	3	5
2	12	13	2	5	28	2	12	10	29	29	8	15	5	3	3	6	7	6	2	2	4	6	8
30	10	9	2	4	28	9	12	4	3	26	5	2	6	7	18	24	7	5	8	6	8	2	8
21	10	13	5	6	29	3	9	5	5	9	17	4	26	5	15	4	7	4	4	20	8	7	12
15	6	9	5	3	20	12	2	8	2	7	3	12	6	9	13	30	7	9	4	7	42	10	7
14	6	17	11	21	20	9	2	24	4	19	9	3	1	2	10	6	34	6	8	8	14	37	7
2	9	7	11	10	39	12	7	22	7	36	6	2	3	6	5	8	17	17	8	18	14	6	4
22	9	6	18	4	13	9	7	2	9	28	6	3	2	17	5	10	17	4	6	6	34	2	4
6	16	4	8	7	13			2	2	6	20	1	1			8	16	5	6	5	9		
5	2	2	8	6	22			5	1	20	7	2	3			15	16	4	8	8	7		
5	2	5	7	5	22			9	15	11	2	3	2			5	6	4	4	7	7		

Category 7: 196 or 197 items; object: 160 x 240

Items no. 1 - 100

P1								P2								P3							
w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h
19	21	33	10	12	23	2	4	15	75	14	5	11	10	1	1	19	15	19	11	17	12	14	23
6	21	16	9	9	9	4	4	12	80	5	2	5	25	2	1	4	15	3	3	5	15	16	23
6	18	3	9	16	9	11	11	27	6	7	2	7	26	11	56	8	5	3	3	10	6	6	31
41	18	8	4	6	18	29	11	3	13	12	9	10	44	10	39	26	5	5	5	15	6	21	22
22	14	17	4	20	13	10	54	10	3	2	1	19	4	5	5	21	10	35	5	8	50	4	22
13	14	20	9	21	13	13	54	9	21	12	8	8	31	3	20	5	24	6	17	30	16	4	17
8	13	19	9	5	25	13	70	8	11	5	28	33	6	2	1	3	7	13	17	6	16	22	17
14	13	8	15	6	25	7	13	6	13	8	6	11	73	2	7	32	7	24	8	15	10	14	8
31	20	6	6	19	25	3	13	2	9	35	7	8	27	7	6	26	92	31	8	6	10	16	8
8	7	5	6	7	16	7	25	51	10	7	14	6	2	13	12	16	92	5	8	10	9	9	4
14	7	3	6	5	16	2	12	6	6	10	45	2	9	17	34	8	5	35	8	15	9	6	4
22	54	5	6	9	9	4	12	11	12	4	19	25	9	16	46	26	5	24	5	6	8	20	4
13	54	17	23	16	9	11	32	2	10	13	17	9	39	1	1	3	17	31	5	5	8	22	4
6	23	5	28	20	12	22	9	8	12	62	9	9	17	6	13	32	17	5	5	12	40	5	29
8	13	6	28	21	12	7	9	13	47	36	11	12	7	3	12	10	5	35	5	9	33	38	7
33	13	11	53	23	10	7	12	14	37	38	18	21	101	6	10	24	5	11	44	6	33	33	7
6	22	20	6	12	10	3	12	3	11	4	2	3	10	9	15	2	8	12	44	6	37	7	32
5	22	19	6	7	9	16	33	3	6	9	5	3	7	6	24	19	8	38	17	5	35	5	32
11	28	3	17	5	9	8	33	1	4	8	3	4	10	1	1	3	7	57	17	21	13	30	49
19	56	5	17	25	7	14	21	1	6	30	20	8	22	5	7	3	7	13	5	4	13	5	22
12	56	39	12	6	7	7	21	27	5	6	7	5	3	4	1	13	10	17	5	4	17	9	22
16	11	8	12	16	47	6	27	14	3	36	10	2	1	2	13	4	27	8	17	22	17	6	37
3	11	31	8	16	14	6	10	10	4	27	7	1	2	11	9	10	14	36	20	16	34	6	31
6	20	41	8	8	14	16	10	5	20	17	3	22	59	1	6	24	14	21	20	6	32	16	31
8	10	23	23	21	16	7	23	3	11	6	4	1	2	3	16	2	11	13	12	5	32	2	7

Category 7 cont.: items no.101 - 197

P1								P2								P3							
w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h	w	h
10	45	29	11	4	9	3	10	7	11	30	8	4	2	6	32	25	7	3	8	4	10	2	4
7	45	9	26	22	14	6	10	8	15	26	7	6	7	1	5	11	12	12	65	47	10	25	11
2	10	4	26	9	7	4	24	6	10	16	32	13	3	12	5	33	22	5	5	3	11	8	34
4	10	17	35	9	7	5	21	6	21	71	25	29	5	7	33	2	5	16	5	12	11	7	7
16	13	12	6	17	4	21	21	3	9	12	25	2	2	2	8	25	5	6	9	14	7	2	7
14	6	12	6	3	4	19	21	1	4	22	20	8	15	6	11	27	10	3	9	12	7	10	29
7	6	4	4	19	17	8	21	10	7	9	55	5	28	9	9	11	10	41	9	3	18	2	10
2	3	10	4	8	17	19	17	3	3	13	27	8	42	4	27	5	15	17	3	5	18	5	10
4	3	8	10	8	11	21	17	5	15	65	16	4	27	10	11	9	15	13	3	12	10	59	19
13	13	13	2	5	11	9	15	2	4	1	3	3	2	38	9	5	30	6	8	9	10	9	23
14	13	6	2	7	10	5	6	33	8	5	2	18	4	5	7	11	13	8	8	9	49	22	15
7	8	10	12	20	10	13	6	16	5	16	57	17	5	6	2	30	13	17	6	5	7	3	15
33	8	3	12	9	7	19	14	9	12	2	1	2	8	3	6	30	21	13	6	3	7	2	9
16	34	22	16	9	7	4	14	10	11	3	3	2	11	6	2	7	17	5	11	26	11	4	7
12	28	18	16	19	20	5	9	6	3	3	2	1	2	32	5	5	17	16	11	4	8	1	7
12	28	7	20	4	20	4	7	10	11	25	11	3	18	12	4	6	24	6	2	47	8	4	2
7	30	17	16	4	10	9	7	39	8	11	8	17	10	16	3	6	22	10	2	3	7	1	2
4	19	3	16	26	10	5	3	17	113	6	4	7	5	72	15	10	22	6	7	12	7	7	10
29	19	13	8	19	11	21	3	13	36	10	9	3	10	14	14	3	2	3	7	12	39	59	10
10	51	6	8	21	11	4	2	28	8	25	8	5	31	2	6	3	2	8	10	9	39	22	8
13	51	8	10	3	3	9	2	17	7	10	6	19	9	3	9	8	10	14	3	5	4	3	8
13	25	19	10	6	3			42	9	12	15	7	8	12	3	6	18	12	3	3	4		
4	21	10	8	18	13			22	57	6	19	10	3			11	8	3	3	51	8		
10	21	3	8	8	27			2	1	2	4	11	37			30	8	5	3	15	8		
4	11	9	9	5	27			15	9	4	7	1	4			3	8	10	26	7	4		

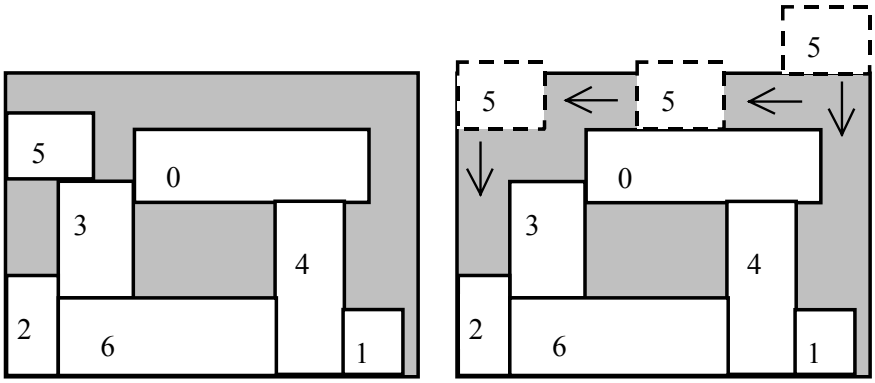

Figure 15: BL routine (Jakobs, 1996)

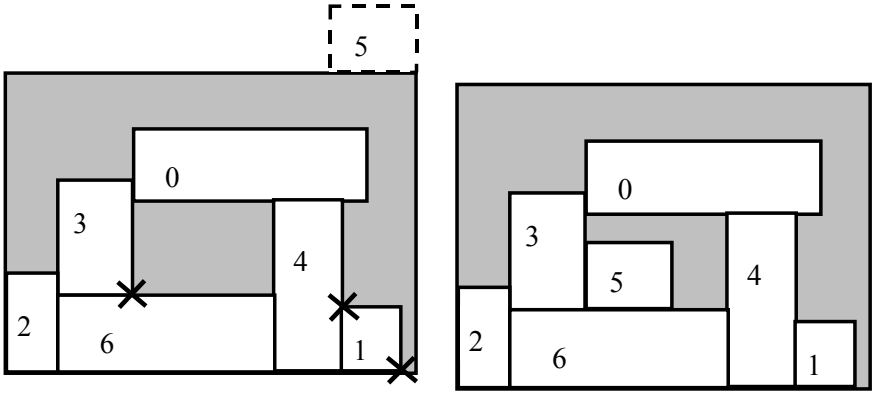

Figure 16: Bottom-Left-Fill heuristic

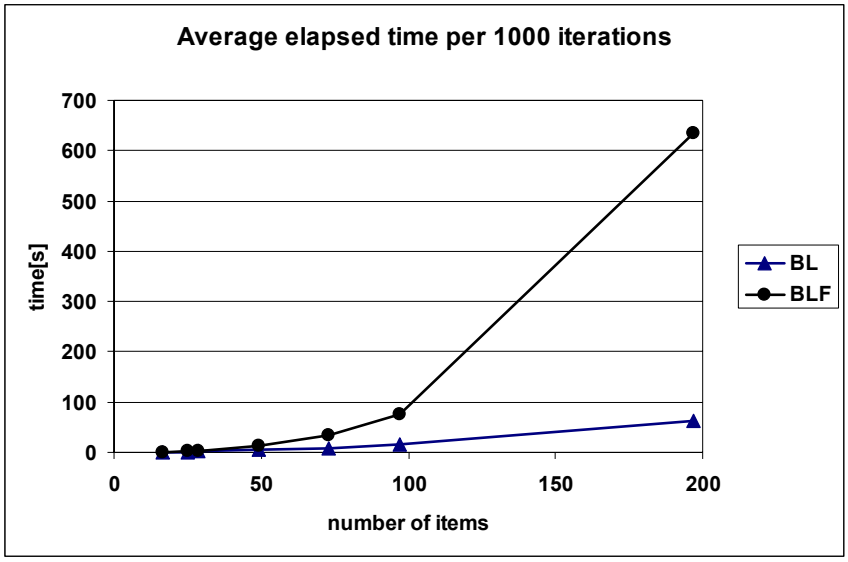

Figure 17: Average elapsed time per 1000 iterations for heuristics

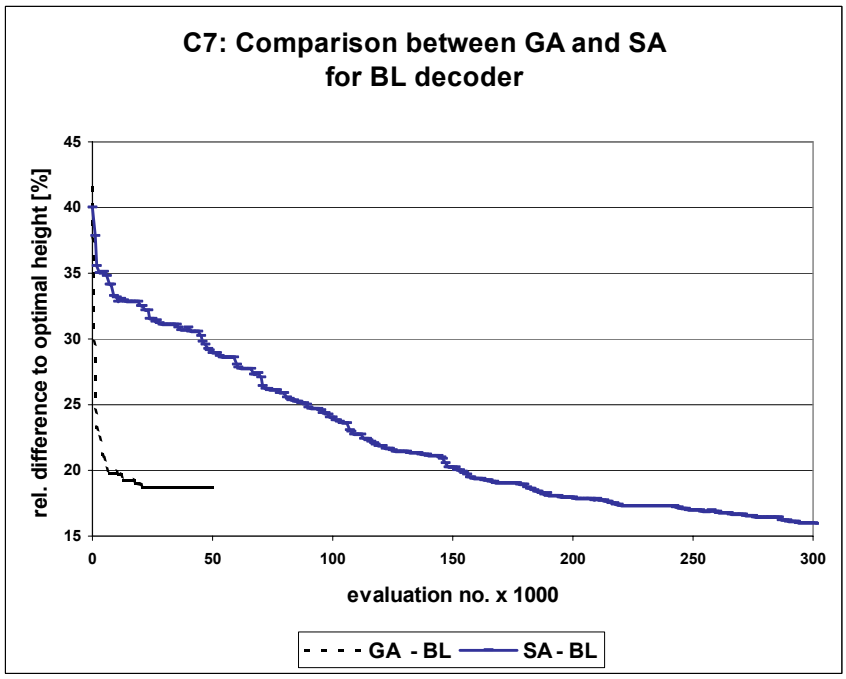

Figure 18: GA and SA + BL for a large problem

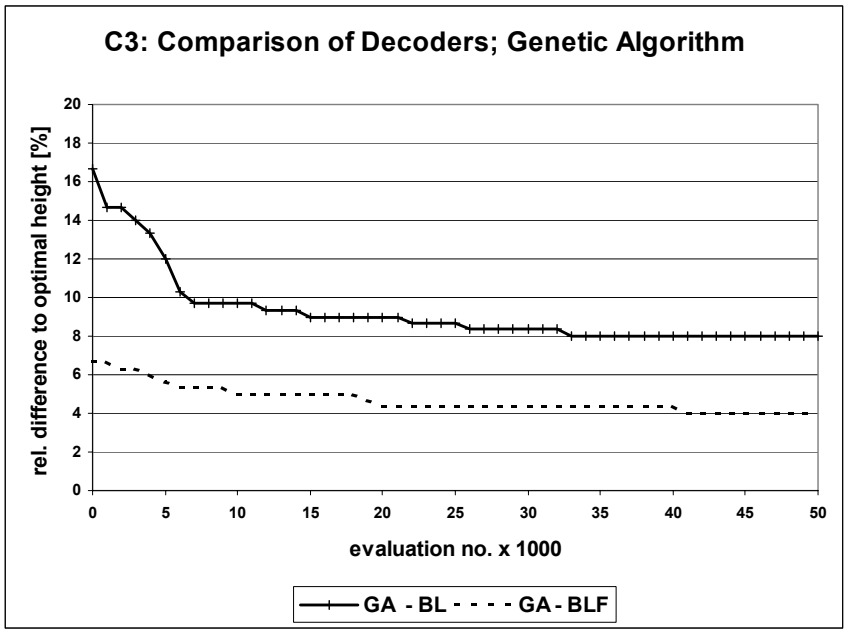

Figure 19: Comparison of the two GAs combined with BL and BLF routine

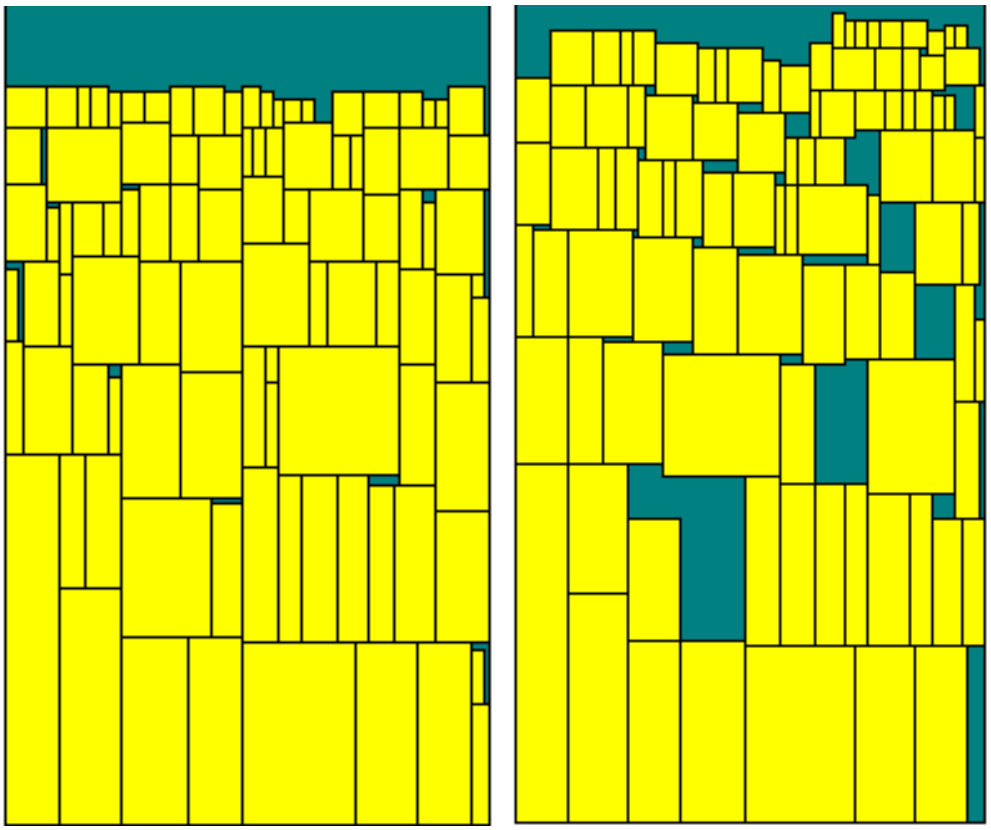

Figure 20: Best layouts for a large problem (C6) with BL (left) and BLF (right); height-sorted sequence

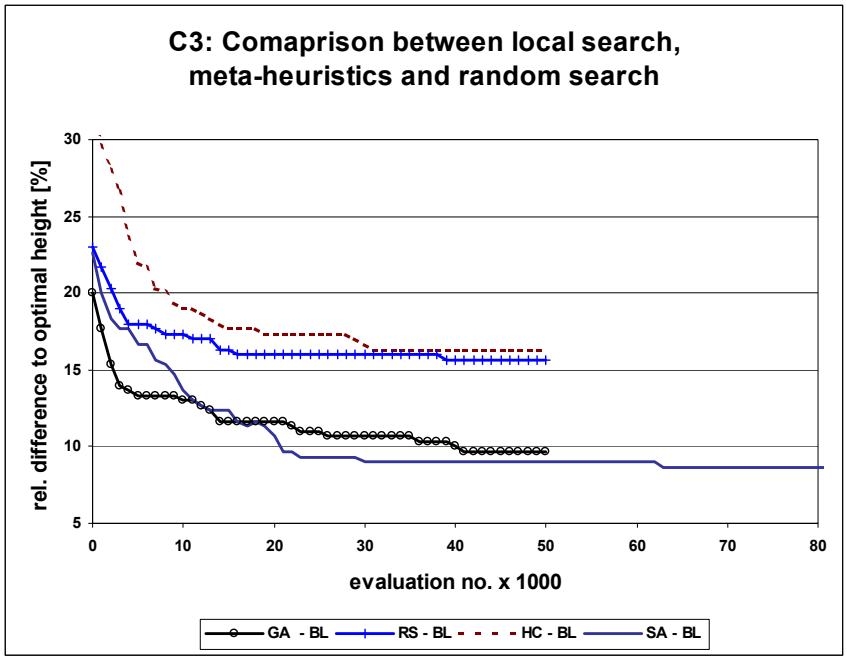

Figure 21: Comparison between local search and meta-heuristic methods (+BL)

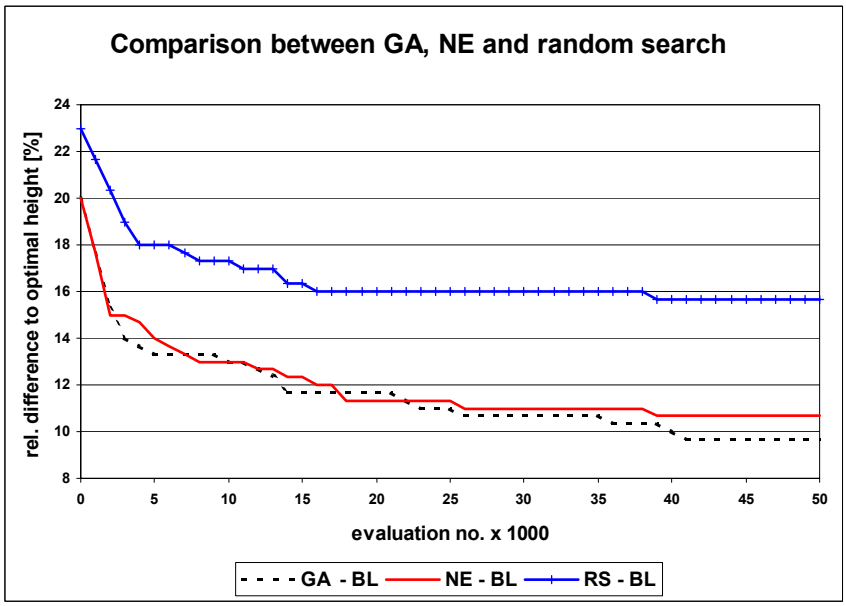

Figure 22: Evolutionary algorithms (GA and NE) and random search for BL for C3

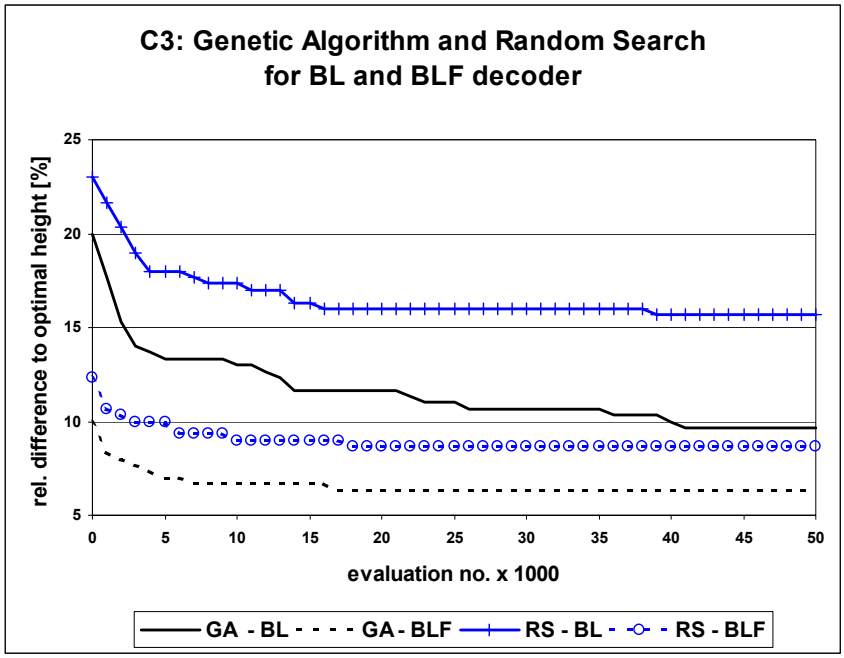

Figure 23: Random search and GAs for both decoders (BL and BLF)

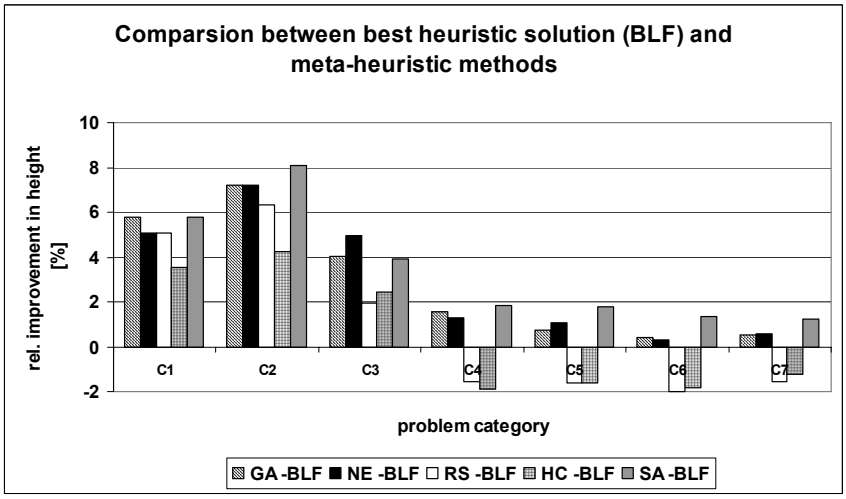

Figure 24: Improvement of the meta-heuristics +BLF in comparison to the best of the heuristic solutions (BLF including sorted sequences) for each problem category

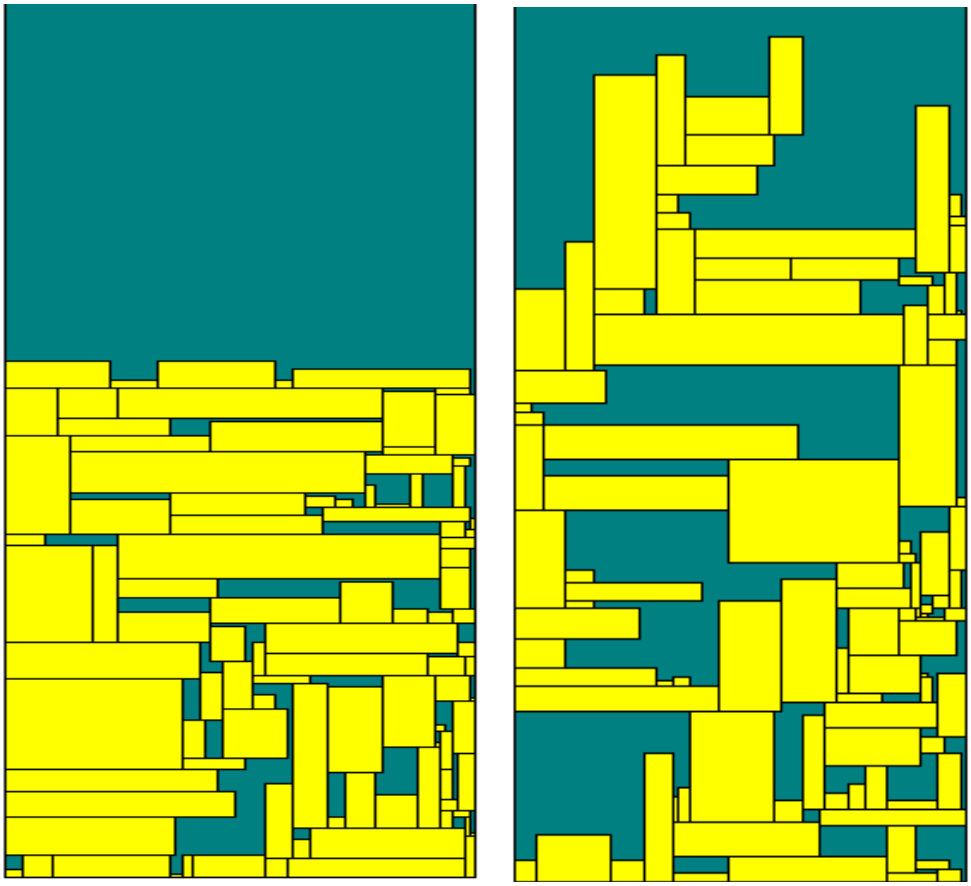

Figure 25: Initial and best layout for GA+BL

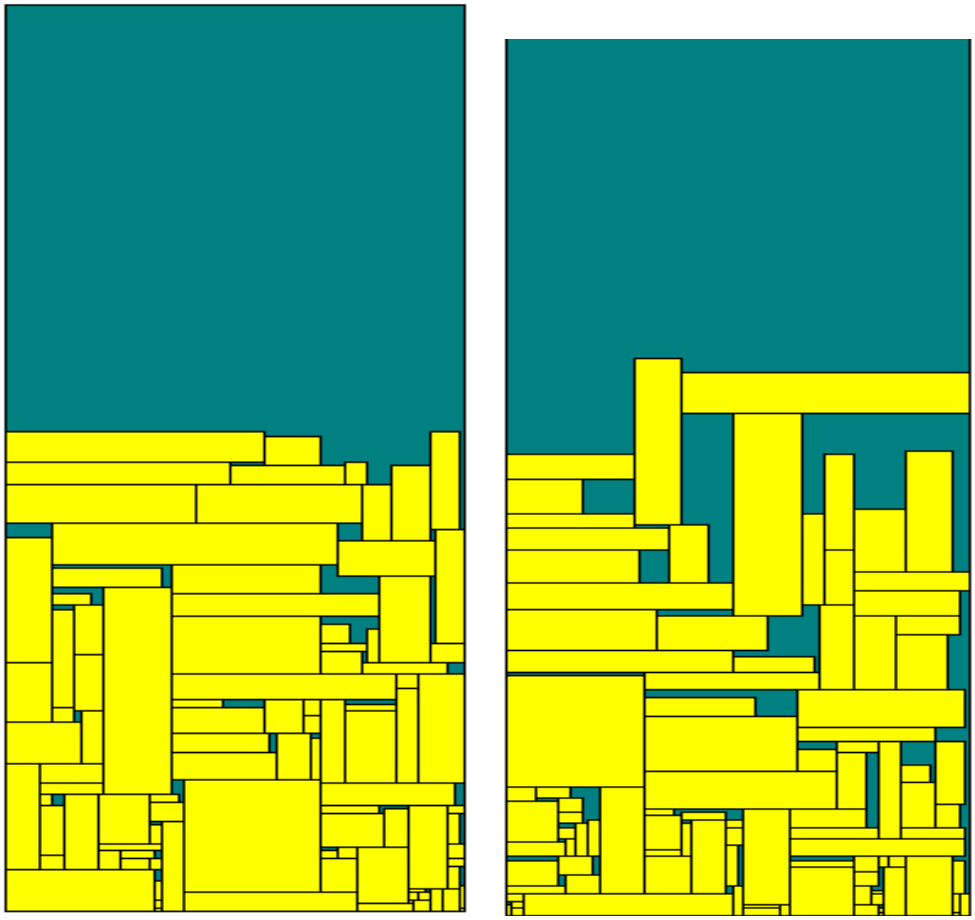

Figure 26: Initial and best layout for GA+BLF

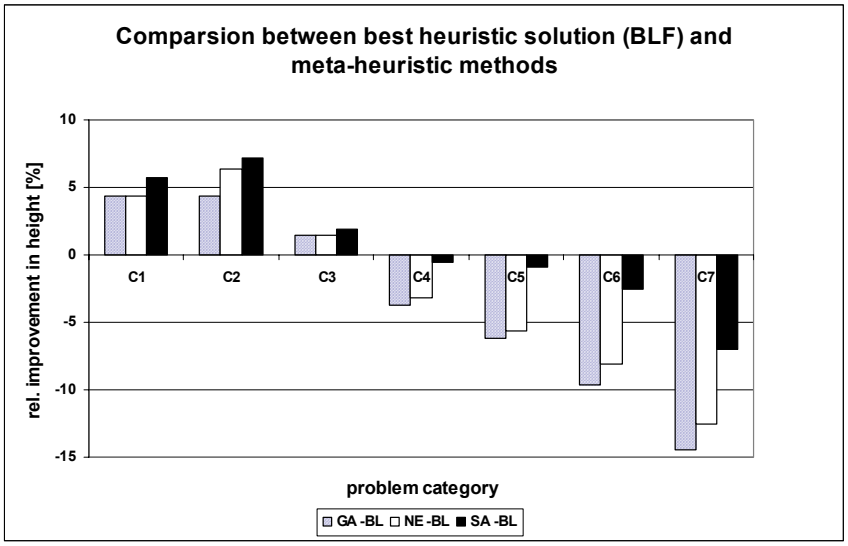

Figure 27: Improvement of the meta-heuristics + BL in comparison to the best heuristic solution (BLF) for each problem category