

A Genetic Algorithm for Solving the Container Loading Problem

H. GEHRING and A. BORTFELDT

FernUniversität Hagen, Kleine Straße 22, D-58084 Hagen, BRD

Abstract: The paper presents a genetic algorithm (GA) for the container loading problem. The main ideas of the approach are first to generate a set of disjunctive box towers and second to arrange the box towers on the floor of the container according to a given optimization criterion. The loading problem may include different practical constraints. The performance of the GA is demonstrated by a numerical test comparing the GA and several other procedures for the container loading problem.

Keywords: Packing, container, genetic algorithm, practical constraints.

1 Introduction

Container loading problems may be grouped in different ways. A basic distinction exists between cases in which a given set of goods has to be loaded completely and cases which allow some goods to be left behind. Whilst the former type of problem involves more than one container, the latter is often restricted to a single container (cf. the distinction made by DYCKHOFF, 1990). Another important distinction concerns the goods to be loaded. BORTFELDT (1994) considers the loading of rectangular goods, i.e. boxes, and defines a cargo comprising only identical boxes as 'homogeneous'. He also refers to a given set of boxes with many different types of boxes as 'strongly heterogeneous' and one with only a few different types of boxes as 'weakly heterogeneous'.

The subject of this paper is the loading of a strongly heterogeneous set of boxes into a single container. The literature references given below are focused on this type of problem and also on genetic approaches.

HAESSLER and TALBOT (1990) present a heuristic method for packing low density products in freight wagons and truck trailers. The packaged products are shipped in cases (corrugated boxes) of standardized dimensions. The core of the heuristic consists of two procedures used alternately. The first procedure forms vertical stacks of cases and the second arranges the stacks on the load area. PORTMANN (1990), GEHRING et al. (1990) and SCHEITHAUER (1992) as well as MORABITO and ARENALES (1994) suggest methods for general loading problems of the type under consideration. PORTMANN's heuristic method fills a given container successively from bottom to top, whereas that proposed by GEHRING et al. generates vertical layers parallel to the container front and fills the container layer by layer. The algorithm from SCHEITHAUER is based on the 'forward state strategy' of dynamic programming, while MORABITO and ARENALES present an AND/OR graph approach. Finally, the heuristics proposed by LOH and NEE (1992), NGOI et al. (1994), BISCHOFF et al. (1995) and BISCHOFF and RATCLIFF (1995) are tailored to weakly heterogeneous loading problems.

Until now, only a few genetic algorithms for loading and packing problems have been reported. PROSSER's hybrid genetic algorithm (1988) is aimed at the packing of a given set of metal plates on a minimum number of pallets. The genetic algorithm suggested by KRÖGER et al. (1993) solves a

two-dimensional packing problem, which corresponds to the three-dimensional problem of loading a strongly heterogeneous set of boxes into a container.

In the given case, the reasonable results which have been achieved through applying genetic approaches to problems of similar complexity, e.g. the line balancing problem (cf. GEHRING and SCHÜTZ 1994), and the ease of considering practical constraints speak in favour of a genetic approach. The genetic algorithm presented in this paper proceeds in two main steps; furthermore, it considers practical constraints. The rest of the paper is therefore structured as follows: In Section 2 the underlying problem is formulated more precisely and the two-step approach referred to above is outlined roughly. The first step treated in Section 3 serves to generate a set of box-towers using a greedy algorithm; the second step presented in Section 4 concerns the covering of the container floor with box-towers in an appropriate way. The inclusion of some practical constraints is treated in Section 5 and the performance of the resulting algorithm is illustrated by some numerical examples in Section 6.

2 Problem formulation and basic approach

In order to meet practical requirements to a greater extent, only stable box arrangements are tolerated, stable here meaning that the boxes have a priori balanced positions and must not fall over. This means that additional precautions, e.g. the use of spacing material, are not necessary. In addition, the boxes must be arranged completely within the container and parallel to its side walls. These requirements lead to the following feasibility conditions:

- (F1) Each box lies on the container floor or it is placed on the top of another box so that its centre of gravity is supported by the second box.
- (F2) Each box is positioned parallel to the side walls of the container.
- (F3) Each box lies completely within the container.

As to the first feasibility condition (F1) it is assumed that the centre of gravity of a box is always given by its geometrical centre. Some of the subsequent parts of this paper require a three-dimensional reference frame. In the following it is assumed that the given container is embedded in a three-dimensional Cartesian system of coordinates: The container lies in the first octant of the system with one of its lower corners in the system origin; the depth, the width and the height of the container are oriented in accordance with the directions of the x-, the y-, and the z-axis.

Problem formulation

The problem under consideration may now be formulated as follows: for a given container of depth x_c , width y_c , height z_c and a given set of nb rectangular boxes b , $b = 1, \dots, nb$, with dimensions $b_{dim1}(b)$, $b_{dim2}(b)$, $b_{dim3}(b)$, weight $b_{weight}(b)$ and value $b_{value}(b) > 0$, determine a feasible arrangement of a subset of boxes within the container so that the total value of the packed boxes is maximized and the relevant constraints are met.

The value of a box may denote its volume or, e.g., its freight rate. In the first case the total volume of the packed boxes is to be maximized and in the second case the total freight.

From the viewpoint of loading and handling the container feasible box positions and the distribution of the weight as well as stability aspects are of interest. In this paper the arrangement of boxes must meet the following five constraints:

- (C1) Orientation constraint:
Each box in a given subset of boxes is to be placed in a restricted way: one or two dimensions of the box may not be oriented in a vertical direction.
- (C2) Top placement constraint:
None of the boxes in a given subset may bear a weight: placing further boxes on the top of these boxes is not allowed.
- (C3) Weight constraint:
The weight of a complete cargo may not exceed a given weight limit.
- (C4) Stability constraint:
The stability of a placed box is calculated as the ratio of the bottom area in contact with the boxes below to the total bottom area of the box. It measures the degree to which the bottom of a box is supported. The stability constraint demands that the stability of all boxes does not fall below a given amount.
- (C5) Balance constraint:
The distance between the x-coordinate of the centre of gravity of the cargo and the midpoint along the container depth $xc/2$ may not exceed a given amount. The y-coordinate of the centre of gravity has to meet an analogous condition.

Basic approach

Formulation of the basic approach requires the introduction of some concepts. It is assumed that an arrangement of boxes meets the feasibility conditions (F1) to (F3). Only a single box of the arrangement, called the base box, is placed on the container floor. Each of the other boxes is placed on the top of another box of the arrangement in a way that the whole of its base is supported by the box beneath. An arrangement of this kind is called a tower. The base of the base box is called the tower base. Figure 1 shows three box arrangements in a container; only one arrangement represents a tower. Two towers are referred to as disjunctive if the corresponding box sets are disjunctive. A set of towers, which are disjunctive in pairs, is called a tower set if each of the given nb boxes is placed in one of the towers.

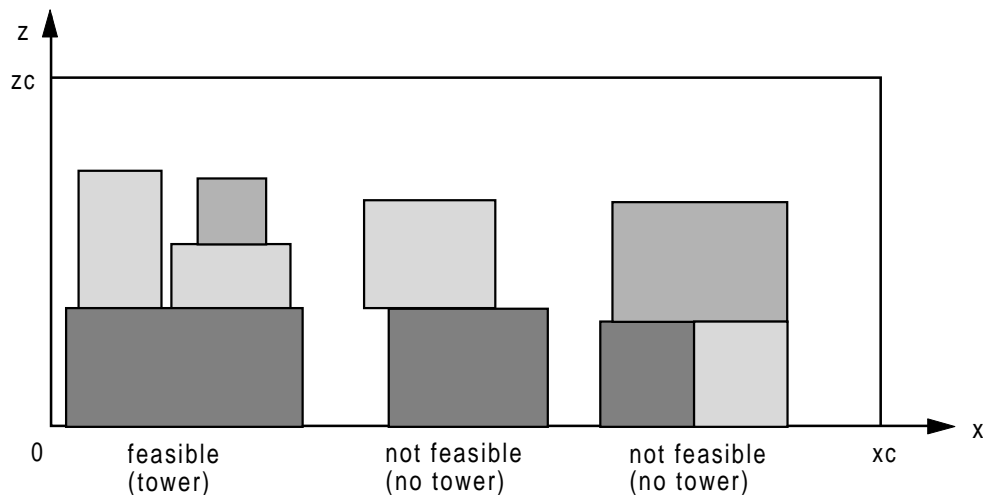


Fig. 1. Three arrangements of boxes in a container.

Using these concepts the basic approach may be formulated as follows:

- (1) In the first step, the given nb boxes are arranged in a tower set. This is done by means of a greedy algorithm that tends to minimize the spare spaces lying above the base boxes.
- (2) In the second step, the floor of the container is to be covered by the tower bases of the generated towers. The resulting two-dimensional problem is solved with a genetic algorithm, which maximizes the total value of the packed boxes. The second step is finished by combining the generated tower set and the best covering of the container floor into a solution.
- (3) Both steps are repeated several times. A repetition includes the generation of a new variant of the tower set and the covering of the container floor with this set. At the end, the best generated solution is used as solution to the problem.

This approach, and especially the chosen concept of a tower, ensures the feasibility of a generated solution.

3 Generation of tower sets

The generation of a tower set comprises the subdivision of the given set of boxes into disjunctive subsets as well as the fitting together of the boxes of each subset into a tower. A semiformal description of the generation procedure is given in Figure 2. The process within the twin-framed box in the diagram is refined in a subsequent step.

The procedure presented in Figure 2 requires some clarification:

- A tower set $TSet$ is generated in a number of $nsteps$ steps; each step includes the addition of at least one tower to $TSet$. Owing to the definition of the set of free boxes $BFree$ the generated new towers and the other towers of $TSet$ are disjunctive.
- In each step, the variation of the height dimension of the chosen base box leads to a maximum of three towers. Only the tower retaining the highest total volume of the included boxes is added to the set $TNew$.
- As long as the condition $istep < nsteps$ holds, the generation of the next tower $tnext$ for a base box $bbasis$ causes no update of the set of free boxes $BFree$. Hence, the generated new towers denoted by $TNew$ are not disjunctive in general. As a consequence, the set $TNew$ is finally reduced to a set of disjunctive towers; the reduced set contains at least one element.
- In the final step ($istep = nsteps$), the generation of the next tower $tnext$ for a base box $bbasis$ is followed by an update of the set $BFree$. For this reason, the remaining free boxes are fitted together into disjunctive towers, and this guarantees the completion of the tower set.

By varying the number of steps $nsteps$ different tower sets may be generated. Let r , $r = 1, 2, \dots$, denote the subsequent repetitions of the two main steps of the basic approach (see Section 2). Then for the r th repetition, the number of steps $nsteps$ is given by the relation $nsteps := r$. The number of repetitions is determined by the parameter $ntsets$.

The purpose of the process within the twin-framed box in Figure 2 is the generation of a tower. The description of this process requires the introduction of some data structures. They concern the description of a tower set, a tower and the positions of the boxes within a tower.

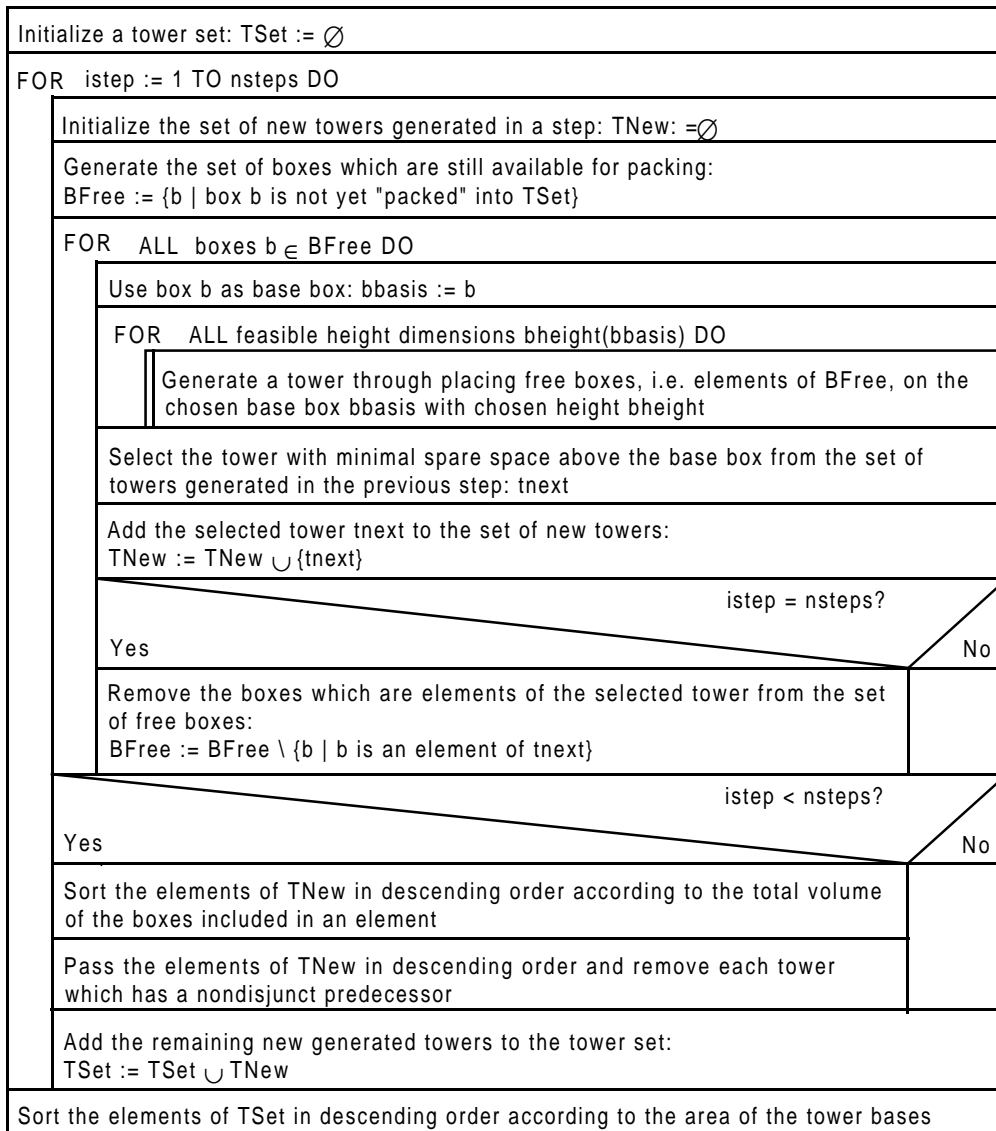


Fig. 2. Generation of a tower set.

A tower set is denoted by a vector $TSet(it)$, $it = 1, \dots, ntowers$, where $ntowers$ is the number of included towers. The elements of the tower set vector are arranged in descending order according to the area of the tower bases.

An element $TSet(it)$, i. e. a single tower, is described by a data structure composed of the following items:

- $tvalue$, the total volume or total value of the boxes defining the tower (depends on the objective function),
- $tloss$, the spare space related to the tower, i.e. difference between the cuboid defined by the tower base and the height of the container and the total volume of the placed boxes,
- $tdim1, tdim2$, the dimensions of the tower base,
- $TBoxes(j)$, $j = 1, \dots, ntboxes$, the vector indicating the placements of the boxes within the tower, where $ntboxes$ denotes the number of placed boxes.

An element $TBoxes(j)$ of the box placement vector is, again, a data structure. It contains the following data items:

- b , the index of the placed box,
- $bdimx, bdimy, bdimz$, the dimensions of the box seen in the directions of the x-axis, y-axis and z-axis,
- $bcornx, bcorny, bcornz$, the coordinates of the corner of the box which is closest to the origin of the coordinate system, i.e. the coordinates of the bottom left corner of the back of the box.

A semiformal presentation of the process generating a single tower is shown in Figure 3.

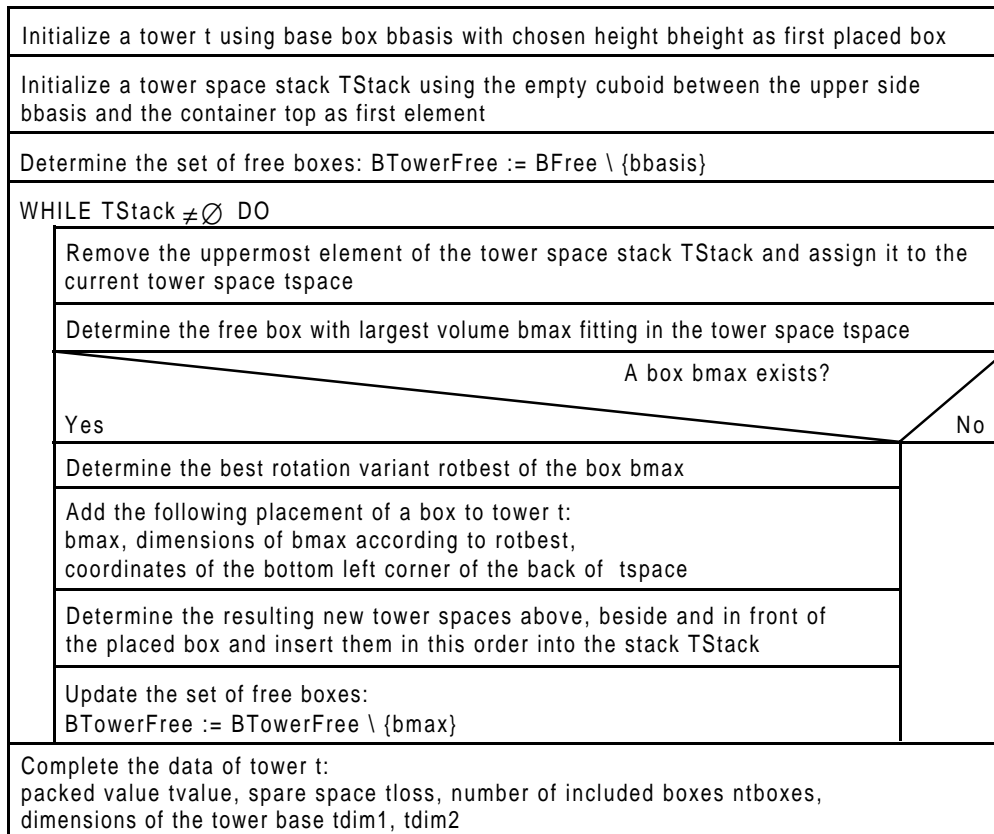


Fig. 3. Subalgorithm describing the generation of a tower.

The following remarks may make it easier to understand the diagram:

- To initialize the tower t , the base box $bbasis$ is provisionally positioned in the origin of a three-dimensional coordinate system. The directed dimensions of the box depend on the chosen height $bheight(bbasis)$ and on a placement rule. The latter implies a placement of a base box so that the larger horizontal dimension and the x-axis have the same direction.
- Additional boxes are fitted into so-called tower spaces; these are empty spaces shaped like a cuboid and lying between the tower base and the container top. A tower space $tospace$ is defined by its directed dimensions and by the coordinates of the bottom left corner of its back. Free tower spaces are kept in a stack denoted by $TStack$.
- A (fitting) box is always placed with its bottom left back corner in the bottom left back corner of a tower space. This arrangement leads to three new tower spaces above, beside and in front of the placed box. Figure 4 illustrates these tower spaces.

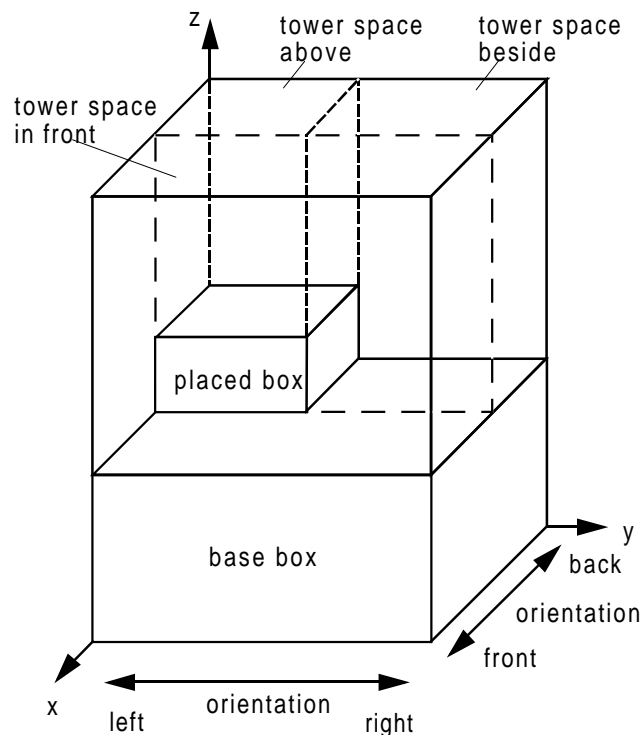


Fig. 4. Illustration of new tower spaces.

- The next tower space to be removed from the tower stack $TStack$ and, if possible, filled with the box with the largest volume is always the uppermost element of $TStack$. Generation of a tower is finished if the stack is empty.
- If it is possible to place a box in different ways in a tower space the best feasible 'rotation variant' *rotbest* is used. The determination of the best rotation variant is based on the volume of a 'test box', which, by way of experiment, may be fitted in one of the new tower spaces generated by the rotated box. Each of the free boxes may serve as test box. The best rotation variant *rotbest* is now defined as that feasible variant which permits the (experimental) placement of the test box with the largest volume.

4 Covering the container floor with a genetic algorithm

The remaining problem of arranging tower bases on the container floor is solved by means of a genetic algorithm (GA). The GA is developed in three steps: At first, a general GA using the incremental replacement concept is formulated. In the second step, this general GA is adapted to the given problem. Finally, the effectiveness of the GA is enhanced by hybridization.

4.1 A general incremental GA

The concept of genetic algorithms was introduced by HOLLAND (1975). Further developments of the concept were suggested by different authors. Systematic treatments of genetic algorithms are given by GOLDBERG (1989) and REEVES (1993).

The following Figure 5 describes a general procedure for a GA with incremental generational change. In the relevant literature, several advantages of the incremental concept are mentioned: higher performance, easier implementation and simpler prevention of the occurrence of duplicate descendants (cf. WHITLEY 1989, DAVIS 1991 and REEVES 1993). For this reason, the concept is preferred to replacement en bloc.

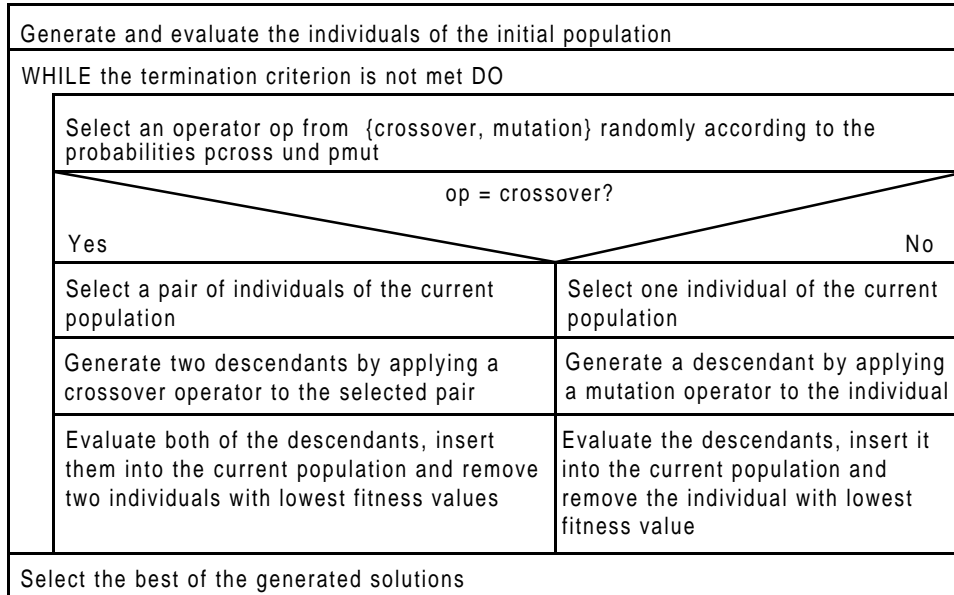


Fig. 5. Procedure of a general incremental GA.

In Figure 5 it is assumed that a crossover and a mutation operator are applied alternatively; a crossover operation leads to two descendants and a mutation operation to one (mutated) individual only. The choice of these operators is based on complementary probabilities p_{cross} and p_{mut} , i.e. $p_{cross} + p_{mut} = 1$. According to the incremental concept, generated descendants replace those individuals of the current population whose calculated fitness values are the lowest. This ensures the survival of the best solution over the whole reproduction process.

4.2 Adaptation of the general GA to the given problem

Adaptation of the general incremental GA to the underlying problem involves the following parts of the algorithm:

- Problem representation, i.e. encoding problem solutions by chromosomes (strings).
- Decoding and evaluation, i.e. assignment of problem solutions to chromosomes and calculation of fitness values.
- Genetic operators - selection, crossover and mutation.
- Initialization and configuration, i.e. determination of the initial population and definition of procedural parameters.

These parts are dealt with below.

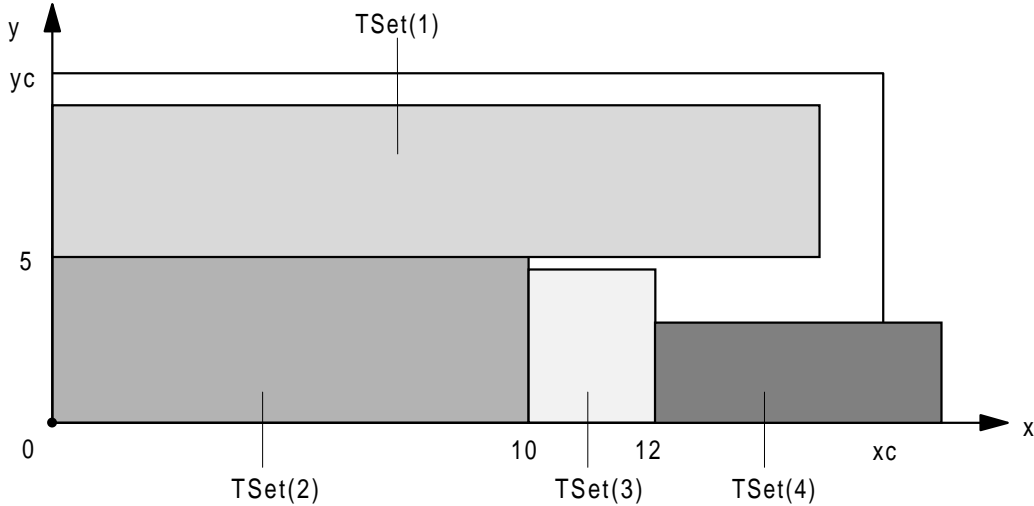
Problem representation

In order to be able to define appropriate representations of coverings of the container floor, the depth of the container is extended infinitely. In the following, the original container is referred to as the real container and the extended container as the virtual container. The same difference applies to the real and the virtual container floor. The extension of the container enables all tower bases of a generated tower set to be placed in the container. A feasible solution, i.e. covering the real container floor, includes only a subset of the total set of tower bases arranged on the virtual container floor.

A solution is now defined in terms of a chromosome as follows: A chromosome is a placement vector $chrom(i)$, $i = 1, \dots, ntowers$, which specifies the sequence of the placements of the tower bases of a tower set and which, in addition, specifies the orientation (rotation variant) of each tower base. An element $chrom(i)$ of the placement vector is a data structure containing two data items: the index it of a tower base and its rotation variant $trot$. The value $trot = 0$ indicates an orientation of the tower base so that its larger dimension is positioned parallel to the y-axis; otherwise the value is $trot = 1$. It should be noted here that the rotation variants of individual boxes must be stipulated during tower generation, whereby a maximum of 6 possibilities are to be differentiated (three-dimensional rotation). In contrast, when a complete tower is arranged on the container floor, only two rotation variants are possible in each case, and these start from one another by means of a vertical rotation of the tower by 90° (two-dimensional rotation).

Decoding and evaluation

The decoding of a chromosome is aimed at the generation of a corresponding feasible problem solution. The description of the decoding process requires a precise definition of a solution as well as the introduction of some concepts.



Legend:

The corresponding solution $sol(i)$, $i = 1, 2, 3, 4$ is composed of the elements

- $sol(1) = \{it = 2; trot = 1; tcornx = 0; tcorny = 0\}$
- $sol(2) = \{it = 1; trot = 1; tcornx = 0; tcorny = 5\}$
- $sol(3) = \{it = 3; trot = 0; tcornx = 10; tcorny = 0\}$
- $sol(4) = \{it = 4; trot = 1; tcornx = 12; tcorny = 0\}$

Fig. 6. Illustration of a solution.

A solution is, analogous to a chromosome, defined by a placement vector $sol(i)$, $i = 1, \dots, ntowers$. An element $sol(i)$ is again a data structure; this structure is now composed of four data items instead of two: the index it of a tower base, its rotation variant $trot$ and, in addition, the x- and y-coordinates of the rear left corner of the tower base $tcornx$ and $tcorny$. As Figure 6 illustrates, a placement vector gives a complete description of the positions of towers in the (virtual) container. The concepts to be introduced are referred to as placement corner, free width and set of placement corners.

Let i ($i \geq 0$) tower bases be placed on the virtual container floor. A point $corner$ of the virtual container floor with the x- and y-coordinates $cornerx$ and $cornery$ is called a placement corner, if the following three conditions hold:

- (1) $cornerx = 0$ or $cornerx$ is equal to the x-coordinate of the front edge of a placed tower base.
- (2) $cornery = 0$ or $cornery$ is equal to the y-coordinate of the right edge of a placed tower base, where it is presupposed that $cornery$ is smaller than the container width yc .
- (3) If a sufficiently small rectangle is placed with its back left corner at the point denoted as $corner$, the rectangle will not overlap with any tower base already placed.

The free width $cwidth$, $cwidth > 0$, of a placement corner $corner$ denotes the difference between the y-coordinate of the placement corner and the y-coordinate of either the left edge of the next tower base placed on the right of the point $corner$ or the right edge of the container floor. In the following, a placement corner is described as triple $corner = (cornerx, cornery, cwidth)$.

The set of all placement corners of a given arrangement of i ($i > 0$) tower bases is denoted by $CSet$. On the elements of the set $CSet$ a relation $<_c$ is defined so that for two placement corners $corner1$ and $corner2$ the following conditions hold:

$$corner1 <_c corner2, \quad \text{if} \quad \begin{array}{l} corner1x < corner2x \\ \text{or if} \quad corner1x = corner2x \text{ and } corner1y < corner2y. \end{array}$$

Hence, $<_c$ represents a linear order and the first element of $CSet$ is the closest to the container back.

The concepts introduced here are illustrated in Figure 7.

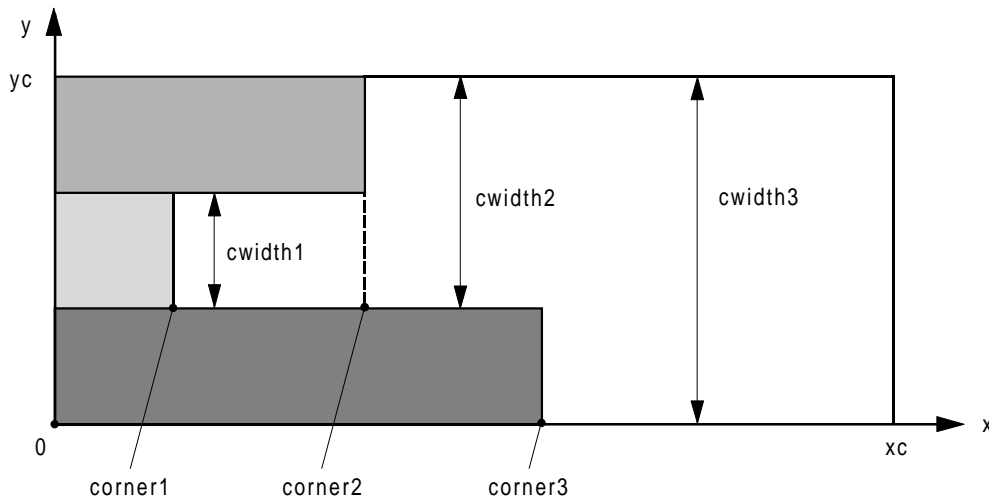


Fig. 7. Arrangement of tower bases including three placement corners.

For i ($i \geq 0$) placed tower bases the set of all placement corners $CSet$ is finite and not empty. In addition, $CSet$ always contains a placement corner $corner^*$, whose width $cwidth^*$ is equal to the width of the container yc . $CSet$ is certainly finite, since the coordinates of a placement corner are restricted to discrete values. Moreover, if a tower base has not yet been placed, then the placement corner $corner^* = (0, 0, yc)$ meets the required properties. Otherwise, let x_{max} denote the maximum of the x-coordinates of the front edges of all placed tower bases; in this case, $corner^* = (x_{max}, 0, yc)$ is a placement corner with a free width yc .

After these preparations the process of decoding chromosomes can now be formulated. A semiformal description of the decoding procedure is given in Figure 8.

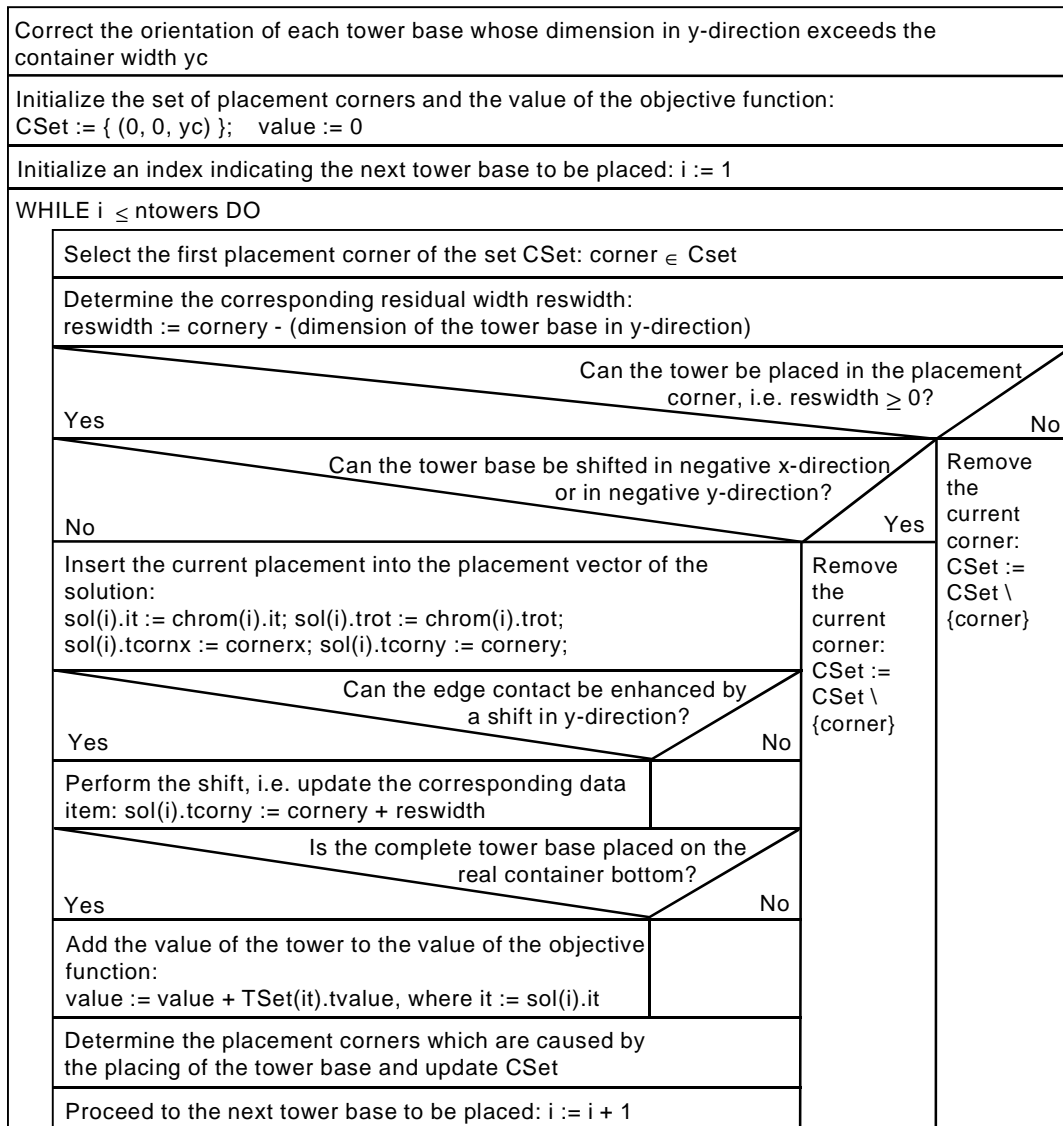


Fig. 8. Structure chart of the decoding process.

The decoding procedure shown in Figure 8 requires some explanation:

- At first, a 90° turn is applied to all tower bases whose dimensions in y-direction exceed the container width. All tower bases of the chromosome to be decoded may now be placed on the virtual container floor in accordance with their assigned rotation variants. 90° rotations lead to corresponding changes to the chromosome.

- The tower bases of the tower set are to be placed in the sequence defined by the placement vector of the chromosome. Placing the i th tower base, $i = 1, \dots, ntowers$, is to be preceded by the determination of the set of placement corners. The placement corners are now tested in the sequence defined by the $<_c$ - relation and the first feasible corner, i.e. the corner which meets the condition $reswidth \geq 0$, is used for placing the i th tower base.
- If a shift in y-direction of a tower base that has already been placed would enhance its contact with an edge of another box or of the container, then the position of the tower base is changed. This situation is illustrated in Figure 9, where a shift of the 4th tower base leads to the desired result.

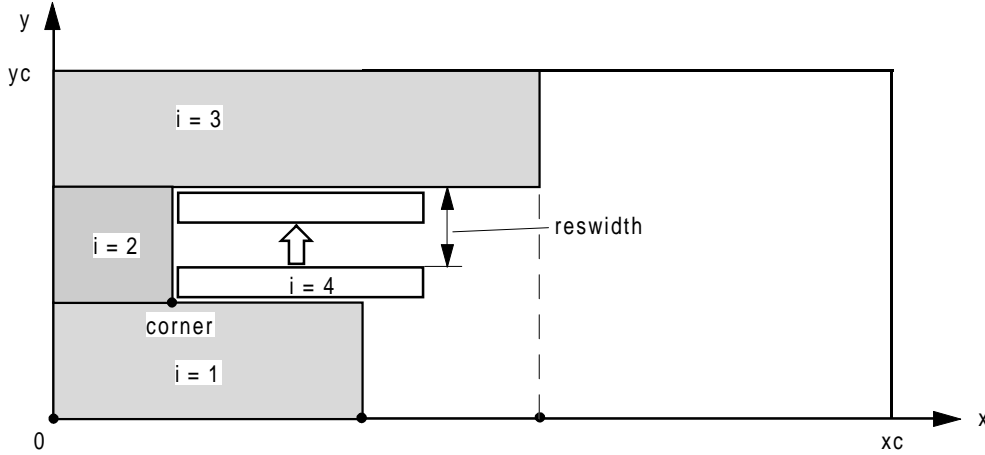


Fig. 9. Enhancement of the edge contact by shifting.

- The decoding process already includes an evaluation step, the determination of the value of the objective function. In this step only those towers are considered whose bases are placed completely on the real container floor. The value of a tower is taken from the tower set vector; the matching vector element is indicated by the index it .

The decoding procedure shown here transforms each chromosome of a population unambiguously into one solution.

The purpose of the evaluation is to calculate a fitness value associated with each chromosome of the population. A naive concept would be the use of the corresponding values of the objective function, which have already been calculated (see Figure 9). Since this concept favours a premature convergence to a local optimum, a ranking procedure is applied instead (cf. e.g. REEVES 1993).

Let $npop$ denote the number of individuals of the population to be evaluated and let sol_k , $k = 1, \dots, npop$, denote the solution associated with the k th chromosome of the population. The chromosomes are then sorted in descending order with respect to the values of the objective function and the fitness $f(sol_k)$ of the k th chromosome, $k = 1, \dots, npop$, is defined as follows:

$$f(sol_k) = f_{max} - (k - 1) \cdot d, \quad (1)$$

where f_{max} ($f_{max} > 0$) is a maximal fitness value and d ($d > 0$) is a fitness decrement. The corresponding definitions are:

$$f_{max} = 2/(npop + 1), \quad (2a)$$

$$d = 2/((n_{pop} + 1) \cdot n_{pop}). \quad (2b)$$

Since the fitness values of all individuals of a population calculated in this manner are positive and add up to one, they may immediately be used as selection probabilities.

Genetic operators

The selection of individuals is based on the ranking method presented above, i.e. the fitness values calculated according to equation (1) are used as selection probabilities. This method combines several advantages (cf. e.g. GOLDBERG 1989, WHITLEY 1989, REEVES 1993): One of them has already been mentioned, the prevention of a premature convergence to a (poor) local optimum. In addition, the selection probabilities are not directly connected with the corresponding objective function values. Owing to this relative independence, selection probabilities may be spread to a sufficient extent at any state of the solution process.

The choice of the crossover and the mutation operators may be greatly influenced by specific properties of the problem representation. The chromosomes used here are permutations of tower bases. Hence, permutation preserving operators should be applied, since they favour the generation of descendants representing feasible solutions. In this paper, alternative operators are chosen and tested. The reason is that it is almost impossible to know a priori which operators will generate the best coverings of the container base. The chosen operators are:

- the uniform order-based crossover (see DAVIS 1991),
- the partially mapped crossover, also called PMX (see GOLDBERG and LINGLE 1985),
- the cycle crossover (see OLIVER et al. 1987),
- the scramble sublist mutation (see DAVIS 1991) and
- mutation by inversion.

As to the application of these operators it has to be considered that each position of a chromosome encodes not only a tower base, but also a rotation variant. For this reason, crossovers or mutations will be carried out in accordance with the following schema:

- With a crossover, descendants *O1* and *O2* are generated from two selected parents *P1* and *P2* in accordance with the selection of the operator, whereby the rotation variants of the descendants remain undetermined. This means that with descendants *O1* and *O2* only two permutations of all tower bases for the given tower set are available at first.
- The rotation variants are now determined in descendant *O1* for all tower bases. The exact rotation variant is assigned to each tower base for *O1* which was allocated to the tower base with the same index in the parent individual *P1*. The rotation variants from the second parent *P2* are transferred analogously to the second descendant *O2*.
- With the mutation of an individual as well, depending on the selection of one of the two mutation operators, at first only a permutation of all tower bases is generated. The rotation variants of the mutated individual are then selected randomly in the set $\{0,1\}$.

Initialization and configuration

The initial population is generated as follows: For each chromosome of the initial population the placement sequence of the tower bases is determined as a random permutation of the tower base indices $it = 1, \dots, ntowers$. The rotation variants of the tower bases are selected randomly from the set $\{0,1\}$.

The configuration is concerned with the choice of the procedural parameters and of the termination criterion. The procedural parameters are fixed as follows:

- The number of tower sets to be generated is constantly $ntsets = 3$.
- The population size is constantly $npop = 50$.
- The probabilities p_{cross} and p_{mut} are constantly $p_{cross} = 0.5$ and $p_{mut} = 0.5$.

The number of generated populations serves as the termination criterion: It is fixed to $ngen = 500$ for a specific test case and to $ngen = 300$ for the rest of the test cases.

The parameters given here were determined with the help of a low-range provisional test which generated the following results:

- the use of three tower sets ($ntsets = 3$) instead of just one or two sets has a favourable effect on the quality of the solution;
- the selection of population sizes $npop < 50$ leads to poorer capacity results;
- increasing the number of calculated generations $ngen$ still further has no positive effect;
- the required calculation time can be kept within reasonable limits with the selected set of parameters.

After the parameter set shown above was determined it was not altered in any way to verify with extensive tests that the procedure generates high-quality results for very different loading problems with a fixed set of parameters and behaves robustly for these purposes.

4.3 Hybridization of the GA

In order to improve its efficiency the GA presented in Section 4.2 has been supplied with additional problem-based knowledge. Hybridization here concerns the generation of the initial population and the decoding of the chromosomes.

Generation of the initial population

Instead of the random method described in Section 4.2 another method is used to generate the initial population. The method is similar to the decoding procedure and proceeds as follows:

- A chromosome is generated in $ntowers$ steps; in a step i , $i = 1, \dots, ntowers$, one tower base is placed on the virtual container floor.

- The arrangement of $i-1$ ($i-1 \geq 0$) tower bases, which have already been placed on the container floor, defines the starting-point for the placement of a tower base in the i th step. For this arrangement, the set $CSet$ of placement corners is determined; for $i = 1$ the set $CSet$ contains only the origin of the system of coordinates.
- The arrangement corners of $CSet$ are tested in the sequence defined by the $<_c$ -relation, which was introduced above. For each element of $CSet$ the set P of all potential placements of tower bases not yet placed is determined.
- The serial testing is stopped as soon as a placement corner is reached which is associated with a non-empty set P . Finally, an element, i.e. a placement of a tower base, is selected randomly from set P and inserted into the placement vector of the chromosome.

Decoding process

During the decoding process the individuals of a generated population are improved, i.e. the sequence of tower bases or their rotation variants is changed in an advantageous way. The improvement concept is based on four heuristic rules that are applied to each tower base of the individual to be decoded in the predetermined sequence of rule 1 to rule 4.

Rule 1:

If it is not possible to place the current tower base with the assigned rotation variant in a given placement corner and if, instead, a 90° turn of the current tower base allows placement, a 90° turn of the current tower base is carried out.

Rule 2:

If the current tower base may only be placed partly on the real container floor and if there exist feasible placements of tower bases not yet used, the alternative with the largest area is realized.

Rule 3:

If the current tower base lies completely on the real container floor and if there exists an alternative tower base with a box tower with a greater volume but with at least the same area which could be arranged completely on the real container floor, the current tower base is replaced by the alternative tower base.

Rule 4:

If

- the current tower base lies completely on the real container floor independent of the assigned rotation variant and if
- there exists no alternative tower base that can replace the current tower base under rule 3 and if
- an additional tower base may not be placed on the right side of the current tower base,

the current tower base is positioned with its larger dimension parallel to the y-axis.

All applications of rules 1 to 4 are correspondingly traced in the chromosomes. In the case of rules 1 and 4, this means that the rotation variant of the tower bases is altered if necessary; in the case of rules 2 and 3, two tower bases in the chromosome are exchanged if necessary.

Rules 1 to 4 have the effect that the container floor is covered in a way which leaves only a few small uncovered areas. In addition, they favour the placing of box towers with high stowed values.

5 Inclusion of practical constraints

Constraints (C1) to (C5) formulated in Section 2 require some modifications of the developed GA.

The orientation constraint (C1) is easily met: Dimensions of boxes which may not be arranged vertically are marked and not used as box heights.

The top placement constraint (C2) implies two different cases depending on the decision whether the constrained box is to be placed above a base box only, or whether it must be used as a base box itself. In the first case, the tower space above the box is locked; i.e. no further boxes may be placed in this space. In the second case, a tower is generated by stacking one box on top of another. Finally, the constrained box is placed on the top of the stack.

The two steps of the basic approach are modified to take account of the weight constraint (C3). In the first step, the weight of the tower is calculated additionally for each of the generated towers. In the second step, the decoding of a chromosome is restricted: As long as the weight limit restricting the total weight of the cargo is not exceeded, placed tower bases are included in the corresponding solution step by step.

The stability constraint (C4) is met automatically for all boxes of a tower if the base box is not affected by a top placement constraint. Otherwise the boxes of the tower and their rotation variants are to be chosen so that the ratio of the bottom area of the base box and the top area of each other box is not less than the required box stability.

To meet the balance constraint (C5), an elementary transformation is finally applied to the best solution: At first, the container is subdivided into vertical, rectangular and non-overlapping layers parallel to its back so that each box lies completely within one of the layers. In general the cumulated depth of all layers falls below the container depth; the resulting difference denotes the extent to which the layers may be shifted in an x-direction. The x-coordinate of the centre of gravity of the cargo is now moved towards the middle of the container as follows: the x-coordinates of the centres of gravity are calculated and the best arrangement is chosen for different arrangements of layers, which are obtained by feasible shifts in an x-direction in combination with permutations of the layer sequence. The y-coordinate of the centre of gravity is moved towards the middle of the container in a similar way: For different arrangements of layers, which are obtained by the reflection of one or more layers with regard to the plane $y = y_c/2$, the y-coordinates of the centres of gravity are calculated and the best arrangement is chosen. Both chosen arrangements are combined.

The outlined elementary transformation concept permits the conclusion that the balance constraint cannot be satisfied in a strict sense. In contrast to the balance constraint, the other constraints can always be satisfied strictly.

6 Computational results

The developed GA has been implemented in C. The configuration of the GA allows problem examples with up to 500 boxes to be computed. The test of the GA, described below, has been subdivided into three phases. All calculations have been performed on a Pentium/130 MHz PC.

6.1 Test of alternative operators

The developed GA has been tested in a kind of pre-test using different combinations of the operators mentioned in Section 4.2. Some of the results may be summarized as follows:

- The different pairs of operators lead to values of the objective function which differ less than 1%; and the respective computing times also differ insignificantly. The mutation operator "scramble sublist" is slightly superior to the inversion. The best results are achieved by the operator pair "uniform order-based crossover/scramble sublist mutation".
- Hybridization causes a significant increase in the values of the objective function and of the computing times. Whilst the growth of the objective function values amounts to nearly 5%, computing times are nearly tripled.
- Each tested operator pair leads to results which are significantly better than the results generated by a random method. The objective function values achieved by the random method are at least about 3% smaller. On the other hand, the computing times caused by the random method are only insignificantly shorter. The random method "simulates" a GA in the sense that all populations of individuals are generated in the same way as the initial population.

The developed GA is always used below with the operator pair "uniform order-based crossover/scramble sublist mutation" and the hybridization concept outlined in Section 4.3. This version of the GA is denoted by CBGAT.

6.2 First comparison with other methods

In the second phase of the test the GA was applied to loading problems which include different constraints. By means of the generator specified in the appendix six series of numerical examples, i.e. loading problems, have been generated randomly; each of the series represents a test case. Each test case consists of 100 numerical examples. The test cases are denoted by GB1 to GB6 and vary with respect to the considered constraints and the container size:

- All test cases include a stability constraint (C4). Test case GB1 includes no further constraint. An orientation constraint (C1) has to be met in test cases GB2 and GB6, and a top placement constraint (C2) in test case GB3. Test case GB4 requires the meeting of a weight constraint (C3). Finally, an orientation constraint (C1), a top placement constraint (C2) and a weight constraint (C3) have to be met simultaneously in test case GB5.
- Test cases GB1 to GB5 are based on a 40' standard container and test case GB6 on a smaller container of the same proportions.

The parameters characterizing the test cases are shown in Table 1. The parameters serve as input data for the generator referred to.

Tab. 1. Parameters of the test cases GB1 - GB6.

Parameter	Test case					
	1	2	3	4	5	6
Number of problems npr	100	100	100	100	100	100
Container depth xc [cm]	1207	1207	1207	1207	1207	121
Container width yc [cm]	237	237	237	237	237	24
Container height zc [cm]	240	240	240	240	240	24
Number of boxes nb	50	50	50	50	100	50
Volume covering $bvol_{cov}$ [%]	100	100	100	100	100	150
Dimension deviation $bdim_{dev}$ [%]	50	50	50	50	50	50
Number of boxes subject to an orientation constraint nb_{orient}	-	50	-	-	50	50
Number of boxes subject to a top placement constraint nb_{top}	-	-	25	-	25	-
Maximum weight of the cargo $load$ [kg]	-	-	-	10000	10000	-
Weight covering $bweight_{cov}$ [%]	-	-	-	250	150	-
Weight deviation $bweight_{dev}$ [%]	-	-	-	10	10	-
Minimum stability of the cargo load min_{stab} [%]	70	70	70	70	70	70

In this phase the GA CBGAT was tested against the heuristic method suggested by GEHRING et al. (1990) (denoted by CBGMM) and the method developed by SCHEITHAUER(1992) (denoted by CL). Method CL was applied with its most efficient version "Variant 1". The comparison of CBGAT with CBGMM is based on test cases GB1 to GB5 and with CL on test case GB6. The reason is that the available configuration of CL was tailored to loading problems with items of smaller dimensions.

The test results are presented in Table 2.

It should be noted that only the GA CBGAT may meet a stability constraint (C4), whereas the methods CBGMM and CL ignore this constraint.

With regard to test cases GB1 to GB5 the method CBGAT leads to a higher utilization of the container volume than the heuristic method CBGMM. Depending on the constraints to be met, the enhancement of the mean utilization lies between 1.19% and 5.33% of the container volume. For test case GB6 the method CBGAT achieves a mean utilization of the container volume which exceeds the mean utilization determined by the CL method by 5.03%. The mean computing time of CBGAT amounts to less than six minutes for all test cases.

Apart from test case GB4, the resulting mean distance between the centre of gravity of the cargo and the half container depth $xc/2$ and the half container width $yc/2$ respectively is less than 5% of the container depth and width respectively. In the case of a 40' standard container these 5% correspond to an absolute distance of 60 cm of depth and 12 cm of width. In test case GB4, with a relatively

higher box density, the achieved mean distance between the centre of gravity and the middle of the container amounts to only about 10% of both directions.

Tab. 2. Results obtained for the test cases GB1 - GB6.

Test case	Method	Volume utilization [%]			
		mean	minimum	maximum	SD
GB1	CBGAT	88.14	80.20	91.82	1.70
	CBGMM	86.95	80.20	90.31	1.67
GB2	CBGAT	82.68	78.33	86.12	1.64
	CBGMM	80.49	70.85	85.08	2.04
GB3	CBGAT	85.45	77.60	91.24	2.48
	CBGMM	80.65	73.23	85.72	1.79
GB4	CBGAT	55.74	45.78	66.18	5.23
	CBGMM	52.91	45.26	64.52	5.08
GB5	CBGAT	80.81	66.51	88.69	3.63
	CBGMM	79.17	66.68	83.81	3.11
GB6	CBGAT	86.74	81.92	90.59	1.64
	CL	81.71	69.23	89.17	2.54

6.3 Second comparison with other methods

In the third phase of the test the GA CBGAT was applied to problems provided by LOH and NEE (1992) and by BISCHOFF and RATCLIFF (1995).

Table 3 shows the results obtained for the 15 problems provided by LOH and NEE (1992), where CBGAT was compared with the methods of LOH and NEE (1992), NGOI et al. (1994), BISCHOFF et al. (1995) and BISCHOFF and RATCLIFF (1995).

It should be noted that the results presented for LOH and NEE's method (1992) under the heading "packing density" are not directly comparable with the volume utilization ratios generated by the other methods. The utilization criterion "packing density" used by LOH and NEE generally overstates the volume utilization (cf. BISCHOFF and RATCLIFF 1995, p. 382).

The GA CBGAT calculates an optimal solution for 12 of the 15 problems and achieves a slightly better average volume utilization than the other methods. For problems LN06 and LN13 CBGAT outperforms all other approaches, and one or two of the other procedures perform better than CBGAT for the two problems LN07 and LN12 only. For each of the problems the computing time of CBGAT amounts to less than four minutes.

Tab. 3. Results obtained for the 15 problems of LOH and NEE (1992).

Problem	Loh and Nee (1992)		Ngoi et al. (1994)		Bischoff et al. (1995)		Bischoff and Ratcliff (1995)		CBGAT	
	No. of boxes left out	Packing density	No. of boxes left out	Volume utilization	No. of boxes left out	Volume utilization	No. of boxes left out	Volume utilization	No. of boxes left out	Volume utilization
LN01	0	78.1	0	62.5	0	62.5	0	62.5	0	62.5
LN02	32	76.8	54	80.7	23	89.7	35	90.0	39	89.5
LN03	0	69.5	0	53.4	0	53.4	0	53.4	0	53.4
LN04	0	59.2	0	55.0	0	55.0	0	55.0	0	55.0
LN05	1	85.8	0	77.2	0	77.2	0	77.2	0	77.2
LN06	45	88.6	48	88.7	24	89.5	77	83.1	32	91.1
LN07	21	78.2	10	81.8	1	83.9	18	78.7	7	83.3
LN08	7	67.6	0	59.4	0	59.4	0	59.4	0	59.4
LN09	0	84.2	0	61.9	0	61.9	0	61.9	0	61.9
LN10	0	70.1	0	67.3	0	67.3	0	67.3	0	67.3
LN11	0	63.8	0	62.2	0	62.2	0	62.2	0	62.2
LN12	0	79.3	0	78.5	3	76.5	0	78.5	0	78.5
LN13	15	77.0	2	84.1	5	82.3	20	78.1	0	85.6
LN14	0	69.1	0	62.8	0	62.8	0	62.8	0	62.8
LN15	0	65.6	0	59.6	0	59.5	0	59.5	0	59.5
Average		74.2		69.0		69.5		68.6		69.9

BISCHOFF and RATCLIFF (1995) provided 700 numerical examples by means of a procedure generating loading problems with a given number of box types. The problems were subdivided into seven test cases, denoted here by BR1 to BR7. The test cases vary with respect to the number of box types; each of the test cases consists of 100 numerical examples.

CBGAT was tested against the methods of BISCHOFF et al. (1995) and of BISCHOFF and RATCLIFF (1995) on the basis of test cases BR1 to BR7. The test results are presented in Table 4. For each test case the number of different box types is shown in brackets. In addition, for each test case the results shown in the lines entitled "Bischoff and Ratcliff - Combined" refer to the maximum of the utilization ratios gained by the methods BISCHOFF et al. (1995) and BISCHOFF and RATCLIFF (1995), respectively (cf. BISCHOFF and RATCLIFF 1995, p. 387).

The computational results obtained for test cases BR1 to BR7 may be interpreted as follows. For all seven test cases the GA CBGAT achieves a higher mean container utilization than the other tested procedures. Enhancement of the mean utilization varies between 0.4% and 4.73% if the combination of the methods of BISCHOFF et al. (1995) and BISCHOFF and RATCLIFF (1995) is considered. However, if CBGAT is compared with the single methods, enhancement of the mean utilization lies between 2.01% and 7.17%. The improvement achieved by CBGAT increases clearly with the number of different box types to be packed. The mean computing time required by CBGAT amounts to less than three minutes for all test cases.

Tab. 4. Results obtained for the test cases BR1 - BR7 of BISCHOFF and RATCLIFF (1995).

Test case	Method	Volume utilization [%]			
		mean	minimum	maximum	SD
BR1 (3)	Bischoff et al. (1995)	81.76	64.76	94.36	5.98
	Bischoff and Ratcliff (1995)	83.79	72.05	93.58	4.67
	Bischoff and Ratcliff - Combined	85.40	73.72	94.36	4.30
	CBGAT	85.80	76.67	94.32	4.40
BR2 (5)	Bischoff et al. (1995)	81.70	66.70	93.76	6.54
	Bischoff and Ratcliff (1995)	84.44	67.98	91.85	3.97
	Bischoff and Ratcliff - Combined	86.25	73.79	93.76	3.49
	CBGAT	87.26	78.43	95.22	3.03
BR3 (8)	Bischoff et al. (1995)	82.98	66.91	92.63	5.44
	Bischoff and Ratcliff (1995)	83.94	75.33	89.65	3.12
	Bischoff and Ratcliff - Combined	85.86	75.33	92.63	3.27
	CBGAT	88.10	81.12	92.86	2.20
BR4 (10)	Bischoff et al. (1995)	82.60	66.46	88.89	4.11
	Bischoff and Ratcliff (1995)	83.71	73.11	90.06	3.07
	Bischoff and Ratcliff - Combined	85.08	78.38	90.06	2.47
	CBGAT	88.04	82.69	91.59	2.05
BR5 (12)	Bischoff et al. (1995)	82.76	70.38	90.39	4.03
	Bischoff and Ratcliff (1995)	83.80	74.87	89.87	2.88
	Bischoff and Ratcliff - Combined	85.21	78.71	90.39	2.49
	CBGAT	87.86	81.70	92.55	2.02
BR6 (15)	Bischoff et al. (1995)	81.50	64.86	89.15	3.88
	Bischoff and Ratcliff (1995)	82.44	72.29	88.39	3.05
	Bischoff and Ratcliff - Combined	83.84	75.22	89.15	2.62
	CBGAT	87.85	84.14	92.47	1.82
BR7 (20)	Bischoff et al. (1995)	80.51	70.50	88.28	3.75
	Bischoff and Ratcliff (1995)	82.01	75.57	86.88	2.58
	Bischoff and Ratcliff - Combined	82.95	75.73	88.28	2.43
	CBGAT	87.68	84.38	90.74	1.43

The method used by BISCHOFF et al. (1995) addresses loading problems with strong stability requirements, e.g. pallet loading problems. This means that a comparison with this method should include an evaluation of the GA from the aspect of stability. In the given case two stability dimensions are of interest, namely vertical and horizontal stability.

Vertical stability concerns support for the box bases and horizontal stability the avoidance of possible lateral shifts of positioned boxes. The developed GA meets vertical stability requirements flexibly, as overhanging boxes are avoided and each box base is supported to a predetermined

extent at least. Horizontal stability requirements may easily be met if there are only a few box types which allow boxes to be placed like bricks in a wall. If, on the other hand, strongly heterogeneous assortments of boxes have to be arranged, it will be difficult to avoid vertical gaps between boxes. Horizontal stability requirements have therefore not been taken into account.

7 Concluding remarks

The developed GA seems to be suitable for container loading problems where simple stability requirements are sufficient. The method promises high container utilization for problems with both weakly heterogeneous and strongly heterogeneous assortments of boxes. Designed for the latter case, however, the GA performs particularly well for higher numbers of box types. The procedure meets some practical constraints and the required computing times appear to be acceptable with respect to practical requirements.

Acknowledgement

The authors wish to thank Dr. E. Bischoff and a further anonymous referee for their constructive comments.

Appendix: Generation of numerical examples for the test cases GB1 to GB6

The numerical examples are generated in series. A series, also called a test case, is defined by the data specified in Table 1. Some of this data requires further comments:

- The dimensions of the container x_c , y_c , z_c must satisfy the inequality $x_c \geq \max(y_c, z_c)$.
- The volume covering $bvol_{cov}$ is a quotient expressing the ratio of the total volume of all boxes to be loaded to the volume of the container.
- The dimension deviation $bdim_{dev}$ denotes the non-negative ratio of the (positive or negative) maximum permitted difference between a box dimension and the mean dimension of all boxes to be loaded to the mean dimension of all boxes to be loaded.
- If the orientation of a box is restricted and if the condition $y_c > z_c$ applies, only the smallest box dimension may be used as box height; if, on the other hand, the condition $y_c \leq z_c$ applies, only the mean dimension may be used as box height.
- The weight covering $bweight_{cov}$ is a quotient expressing the ratio of the total weight of all boxes to be loaded to the maximum permitted load.
- The weight deviation $bweight_{dev}$ denotes the non-negative ratio of the (positive or negative) maximum permitted difference between the weight of a box and the mean weight of all boxes to be loaded to the mean weight of all boxes to be loaded.

Generation of a test case includes the following steps:

- (1) Container volume $cvol := xc \cdot yc \cdot zc$;
- (2) Mean box volume $bvol_{med} := (1/nb) \cdot cvol \cdot (bvol_{cov}/100)$;
- (3) Mean box dimension $bdim_{med} := bvol_{med}^{1/3}$;
- (4) Minimum box dimension $bdim_{min} := 0.05 \cdot \min(yc, zc)$;
- (5) Mean box weight $bweight_{med} := (1/nb) \cdot load \cdot (bweight_{cov}/100)$
- (6) FOR $i := 1$ TO npr
 - FOR $b := 1$ TO nb
 - choose 3 dimensions of box b at random from the interval
 $[\max(bdim_{min}, bdim_{med}(1 - bdim_{dev}/100)), bdim_{med}(1 + bdim_{var}/100)]$;
 - choose the weight of box b at random from the interval
 $[bweight_{med}(1 - bweight_{dev}/100), bweight_{med}(1 + bweight_{dev}/100)]$;
 - ENDFOR
 - choose at random nb_{orient} boxes affected by an orientation constraint;
 - choose at random nb_{top} boxes affected by a top placement constraint;
- ENDFOR.

References

- BISCHOFF, E. E., JANETZ, F. & RATCLIFF, M. S. W. (1995). Loading pallets with non-identical items. *European Journal of Operational Research*, Vol. 84, pp. 681 - 692.
- BISCHOFF, E. E. & RATCLIFF, M. S. W. (1995). Issues in the development of approaches to container loading. *Omega*, Vol. 23, pp. 377-390.
- BORTFELDT, A. (1994). A genetic algorithm for the container loading problem. *Proceedings of the Conference on Adaptive Computing and Information Processing*, London, Vol. 2, pp. 25-32.
- DAVIS, L. (Ed) (1991). *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold.
- DYCKHOFF, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, Vol. 44, pp. 145-159.
- GEHRING, H., MENSCHNER, K. & MEYER, M. (1990). A computer-based heuristic for packing pooled shipment containers. *European Journal of Operational Research*, Vol. 44, pp. 277-288.
- GEHRING, H. & SCHÜTZ, G. (1994). Zwei genetische Algorithmen zur Lösung des Bandabgleichproblems. In B. WERNERS & R. GABRIEL (Eds) *Operations Research, Reflexionen aus Theorie und Praxis*, pp. 85-128. Berlin: Springer.
- GOLDBERG, D. E. & LINGLE, R. (1985). Alleles, loci, and the traveling-salesman problem. In J. J. GREFENSTETTE (Ed) *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 154 - 159. Hillsdale: Lawrence Erlbaum Ass.

- GOLDBERG, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Reading: Addison-Wesley.
- HAESSLER, R. W. & TALBOT, F. B. (1990). Load planning for shipments of low density products. *European Journal of Operational Research*, Vol. 44, pp. 289-299.
- HOLLAND, J. H. (1975). *Adaption in natural and artificial systems*. Ann Arbor: The University of Michigan Press.
- KRÖGER, B., SCHWENDERLING, P. & VORNBERGER, O. (1993). Parallel genetic packing on transputers. In J. STENDER (Ed) *Parallel genetic algorithms: theory and applications*, pp. 151 - 186. Amsterdam: IOS Press.
- LIN, J.-L., FOOTE, B., PULAT, S. CHANG, C.-H. & CHEUNG, J.-Y. (1993). Hybrid genetic algorithm for container packing in three dimensions. *Proceedings of the 9th IEEE conference on Artificial Intelligence*, Washington DC, IEEE Computer Society Press, pp. 353 - 358.
- LOH, T. H. & NEE, A. Y. C. (1992). A packing algorithm for hexahedral boxes. *Proceedings of the Conference of Industrial Automation*, Singapore 1992, pp. 115 - 126.
- MORABITO, R. & ARENALES, M. (1994). An AND/OR-graph approach to the container loading problem. *International Transactions of Operational Research*. Vol. 1, No. 1, pp. 59 - 73.
- NGOI, B. K. A., TAY, M. L. & CHUA, E. S. (1994). Applying spatial representation techniques to the container packing problem. *International Journal of Production Research*, Vol. 32, pp. 111 - 123.
- OLIVER, L. M., SMITH, D. K. & HOLLAND, J. R. C. (1987). A study for permutation crossover operators on the traveling salesman problem. In J. J. GREFENSTETTE (Ed) *Proceedings of the 2nd International Conference on Genetic Algorithms*, pp. 224 - 230. Hillsdale: Lawrence Erlbaum Ass.
- PORTMANN, M. C. (1990). An efficient algorithm for container loading. *Methods of Operations Research*. Vol. 64, pp. 563-572.
- PROSSER, P. (1988). A hybrid genetic algorithm for container loading. *Proceedings of the 8th European Conference on Artificial Intelligence*, pp. 159 - 164. London: Pitman.
- REEVES, C. R. (Ed) (1993). *Modern heuristic techniques for combinatorial problems*. Oxford: Blackwell Scientific Publications.
- SCHEITHAUER, G. (1992). Algorithms for the container loading problem. *Operational Research Proceedings 1991*, pp. 445-452. Berlin: Springer.
- SMITH, D. (1985). Bin Packing with adaptive search. In J. J. GREFENSTETTE (Ed) *Proceedings of the First International Conference on Genetic Algorithms* (pp. 202 - 206). Hillsdale: Lawrence Erlbaum Ass.
- WHITLEY, D. (1989). The GENITOR algorithm and selective pressure: why rank-based allocation of reproductive trials is best. In J. D. SCHAFFER (Ed) *Proceedings of the 3rd International Conference on Genetic Algorithms* (pp. 133 - 140). Los Altos: Morgan Kaufmann.