# Parallel Coevolutionary Algorithm for Three-Dimensional Bin Packing Problem

Wojciech Bożejko[1(✉)], Łukasz Kacprzak[1] and Mieczysław Wodecki[2]

[1] Department of Automatics, Mechatronics and Control Systems
Faculty of Electronics, Wrocław University of Technology
Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland
wojciech.bozejko@pwr.edu.pl
lukasz.kacprzak@pwr.edu.pl
[2] Institute of Computer Science, University of Wrocław
Joliot-Curie 15, 50-383 Wrocław, Poland
mieczyslaw.wodecki@ii.uni.wroc.pl

**Abstract.** The work considers the problem of three-dimensional bin packaging (3D-BPP), where a load of maximum volume is put in a single container. To solve the above mentioned problem there was a coevolutionary parallel algorithm used basing on the separate evolution of cooperating subpopulations of possible solutions. Computational experiments were conducted in a neighbourhood of clusters and aimed to examine the impact of parallelization algorithm on the computation time and the quality of the obtained solutions.[1]

## 1 Introduction

The three-dimensional bin packing problem (3D-BPP) is a major industrial issue which has a number of practical applications. In a variant of maximizing, the problem boils down to filling the empty space inside a single rectangular cuboid (called *bin*, hereafter referred to as container) with boxes of the greatest volume. With proper management of the available space it is possible to reduce the costs associated with the storage of goods and their transport. The three-dimensional bin packing problem belongs to difficult optimization problems, which is therefore ranked amongst the class of NP-hard problems.

Many heuristic approaches , especially evolutionary algorithms, are used to find approximate solutions to problems, both for one-, two- and three-dimensional packing. However, in connection to the fact that these methods are general and do not use detailed information about the problem, many researchers supplement them with additional heuristic, more suitable for the specific nature of the issue and accepted limitations. Such heuristics are usually responsible for finding acceptable positions for objects in containers (2D packing), or boxes in containers (3D packing) and are (called *decoding procedure*). Method of action

[1] The work was partially supported by the OPUS grant DEC-2012/05/B/ST7/00102 of Polish National Centre of Science.


L. Rutkowski et al. (Eds.): ICAISC 2015, Part I, LNAI 9119, pp. 319–328, 2015.
DOI: 10.1007/978-3-319-19324-3_29

in decoding procedure determines the (*packing strategy*). The role of an evolutionary algorithm, used together with a decoding procedure, may boil down to find the best order (method) of packing, in which items or boxes, will be used by heuristic.

Stawowy [12] developed an evolutionary algorithm for one-dimensional packing problem. The aim of this study was to create a possibly simple algorithm, solving the problem at a level similar to more specialized algorithms. The offspring is created with the use of mutation of operators, without a crossover phase. Tan et al. [11] proposed an evolutionary algorithm for two-dimensional packing problem. The authors do not represent a feasible solution as a vector, but as a structure, called a particle, which creates a set of swarm (called *particle swarm*). A single particle contains information not only on the quantity of used containers (called *bin*) but also on arranged in them rectangular objects, and the best results. The work formulates a mathematical model for two-dimensional packing problem with additional restrictions. The center of gravity of the total load was taken into account.

The works of He et al. [6], Wu et al. [13] or Karabulut and Mustafa [9] presented the problem of three-dimensional packing, wherein boxes are placed in a container of fixed length and width but of varying height. In a variant of the problem considered by He et al. [6] and Wu et al. [13] boxes can have different sizes and arbitrary rotation. A single solution is represented as a chromosome, containing information about the order of packing boxes and a kind of rotation of each of them. A decoding procedure uses concept of reference points (called*extreme points*, *reference points*). The authors [6] improved the idea proposed in [13]. The framework has been created (*global search framework*- GSF) using the concept of the *evolutionary gradient*. Karabulut and Mustafa [9] proposed a method for finding positions of boxes in a container, called *Deepest Bottom Left with Fill Method* (DBLF), which is an extension of the procedure given by Hopper [8], for two-dimensional packing problem . The method inserts a box on the first allowed position, located in the deepest possible (*deepest*), the lowest (*bottom*) and the left-side of the container (*left*). Boxes do not have the possibility of rotation.

In their works Goncalves and Resende [7], Bortfeld and Gehring [2], Kang et al. [10] discuss the problem of filling in a single container with fixed dimensions. Kang et al. [10] developed a packing strategy presented by Karabulut and Mustafa [9]. There has been introduced the concept of a rectangular spatial object (ang. *cuboid space object*). The object represents the space inside the container, inside which it is possible to pack the boxes. Using of objects is aimed at, among others, reducing computation time. The boxes of volume and sizes which do not fit into the space occupied by the object are ignored at this stage of algorithm's acting. Goncalves and Resende [7] have developed a parallel algorithm in which the chromosome contains the information about the order of packing boxes into the container and a layer in which the box can be placed. Layer is a rectangular structure (vertical or horizontal), consisting of boxes of the same type used for filling the empty space inside the container (called *maximal*

*spaces*). The authors present a procedure for coupling of free space -(*MaxJoin*). Bortfeld and Gehring [2] presented a genetic algorithm for which the packing strategy is based on a layered technique. The work considers several practical constraints of the problem, i.e. the number of boxes' rotations, restrictions on the load stability, its weight and balance.

Bischoff and Ratcliff [1] held a discussion on the methodology of solving the problem of container filling (called *Container loading problem*). They discussed a wide range of restrictions of the above mentioned issue, designed the procedure that generates multiple instances of the problem, leading to better assessment of the quality of the algorithms developed by researchers who furtherly extended the problem.

## 2 Problem Definition

The three-dimensional bin packing problem (3D-BPP), in the considered variant, relies in packing of such a number of boxes into a single container that, with the fulfillment of all adopted restrictions, the total volume of load was as large as possible. The container $K$(see Figure 1), has a form of a rectangular cuboid

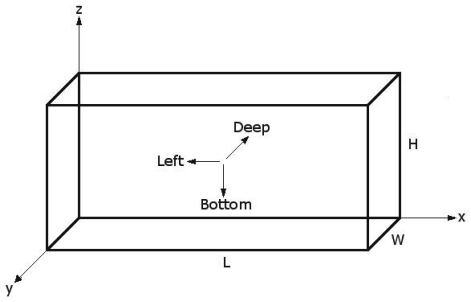


**Fig. 1.** Container

with fixed dimensions: length $L$, width $W$, height$H$ and volume $V$. There is a set of boxes $P = \{p_0, p_1, \ldots, p_{n-1}\}$, type $T = \{t_0, t_1, \ldots, t_{m-1}\}$ and rotation $R = \{0, 1, 2, 3, 4, 5\}$, representing possible rotations of boxes. Each box $p_i \in P$ has a certain type $u_i$, defining its dimensions and possible rotation, i.e. $u_i = t_k$, dla $k \in \{0, 1, \ldots, m-1\}$, $i = 0, 1, \ldots, n-1$, $u \in U = \{u_0, u_1, \ldots, u_{n-1}\}$. A given type $t_k \in T$ can be written in the form of: $t_k = (l_k, w_k, h_k, MR_k)$, where $MR_k \subseteq R$ and determines available rotations, whereas $l_k, w_k, h_k$ denote respectively: the length, width and height of the box. The set $MR_k$ may contain from one to six rotations. A solution $e = (PZ_e, q_e)$ of the packing problem under consideration defines a set of packed boxes $PZ_e \subseteq P$, $PZ_e = \{c_0, c_1, \ldots, c_{q_e}\}$, where $q_e$ is the number of boxes in the set $PZ_e$. The solution $e$ to be found must take the form in which the sum of the volume of packed boxes:

$$\max_{0 \le c_0 \le c_1 \le \ldots \le c_{q_e-1} \le n-1} \sum_{z=0}^{q_e-1} v_z$$

was maximum and fulfilling the following constraints:

- box $c_z$ must lie entirely within the container, parallel to the side walls, in one of the available type of rotations,
- box $c_z$ cannot occupy the space previously occupied by the packed boxes,
- box $c_z$ must be placed on the bottom of the container, or on top of another.

To determine the position of boxes there are used coordinates in the Cartesian coordiante system of reference. A set of boxes $P$ may be of unrestricted character, ranging from slightly heterogeneous(*weakly heterogeneous*) to highly heterogeneous (*strongly heterogeneous*). If the set is weakly heterogeneous, it means that the instance of the problem has few types, with plenty of boxes for each of them, whereas a strongly heterogeneous set consists of many types of boxes, with a small quantity of each type.

## 3 Algorithm Descripion

The proposed algorithm combines the features of a genetic algorithm and heuristic used as a decoding procedure. This section describes the representation adopted for the solution, the method of determining the position of boxes in a container (decoding procedure) and the coevolutionary algorithm.

**Representation of Solution.** An individual is represented as a chromosome composed of two parts: Sequence and Rotation (see Figure 2). The first one contains the sequence of boxes, the second one corresponding to each box - rotation number, used to pack the boxes into a container. Each part of the chromosome has a length equal to a number of boxes in the set $P$, regardless of how many of them will be used in the solution. For a number of boxes equal to $n$ the chromosome length is $2n$. The rotation number stored at position $d_s$ corresponds to a box number stored in a place $a_s$ of a chromosome. The division of the chromosome into two parts is also used in the work [6,13]. On the basis of the chromosome using the decoding procedure, the actual form of a solution $e$ is calculated. In reference to the $n \geq q_e$, which means that not all decoded in the chromosome boxes must be included in the set $PZ_e$, an individual represents the solution $e$ in an approximate way. A case, in which $q_e = n$ means that the solution $e$ is optimal. In the further part the terms: solution, chromosome and individual will be the same.

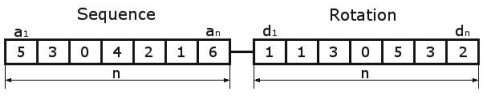


**Fig. 2.** Chromosome

**Decoding Procedure.** The task of the decoding procedure is to find possible positions in the container for the greatest number of boxes, whose sequence

is decoded by the chromosome. In practice, for solution $e$, this means finding elements $q_e$ forming the set $PZ_e$. Heuristic used for this purpose was based on the version of the method presented in the work [9], for bin packing problem with variable height of the container (called *strip packing*). Block diagram of the procedure is shown in Figure 3. In the first step (Initial calculations), the boxes are turned, depending on specific rotation defined in the second part of the chromosome. Then the procedure leads us to place boxes in the container, according to the given sequence. An important element is the position list used to hold the currently available coordinates that are considered when looking for a free place into which $b_s$ box can be packed. If there are no boxes in the container, the position list contains only the coordinates (0,0,0), which is the beginning of the container. In case when the list contains more items, they are

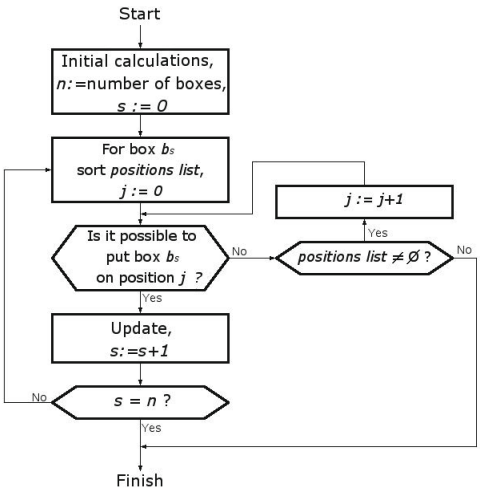


**Fig. 3.** Scheme of decoding procedure

arranged according to the following order: depth (*deep*, a minimum value on the y-axis), bottom ( textit bottom, the smallest possible value of the z-axis), left ( textit left, and the lowest value on the x-axis). In order to determine whether it is possible to insert the considered $b_s$ box in the selected position $j$, heuristic checks if it fits completely within the container and whether it does not disturb the space occupied by the pre-packaged boxes. If the insertion is possible, the position $j$ is removed from the list, and in its place there are three new ones added, created by the location of the box $b_s$. These positions have coordinates $(x_j + l_s, y_j, z_j)(x_j, y_j + w_s, z_j)(x_j, y_j, z_j + h_s)$ and are referred to as *reference points* or *extreme points*) [6, 13] . Then, in case of the remaining boxes, the procedure takes more of them into consideration, sorting an updated list of available positions and checking the opportunity to insert them in one of the positions, otherwise, the procedure is terminated. If you insert a $b_s$ box in

$j$ position is not possible, the heuristic tries to put them in different positions from the list until its termination. The fulfillment of the condition *position list* $\neq \emptyset$ (see Figure 3) means that in the *position list* there are coordinates not taken into consideration within the context of $b_s$ box. When it is not possible to place the box on any of the available positions, the procedure stops.

At the end of working of decoding procedure, the adaptation (cost) function value is calculated for the given solution $e$:

$$F(e) = \frac{\sum_{z=0}^{q_e-1} v_z}{V} 100\%,$$

for $e = (PZ_e, q_e)$, where $v_z$ is the volume of subsequently packed boxes $c_z \in PZ_e$, $|PZ_e| = q_e$ a $V$ is the volume of container $K$. Information concerning the value of adaptation function and packed boxes (their dimensions, the coordinates in the container, the type of rotation) are stored and directly related to the concerned individual, therefore, the procedure is restarted for a given chromosome only in case of its modification.

**Coevolutionary Algorithm.** A coevolutionary algorithm is a parallel multi-track algorithm, based on the island migration model. It explores the space of solutions using parallel threads in the search. The processes exchange between themselves information obtained during the exploration of their own trajectories, with the result that the algorithm is ranked among the subclass of co-operating algorithms [3–5].

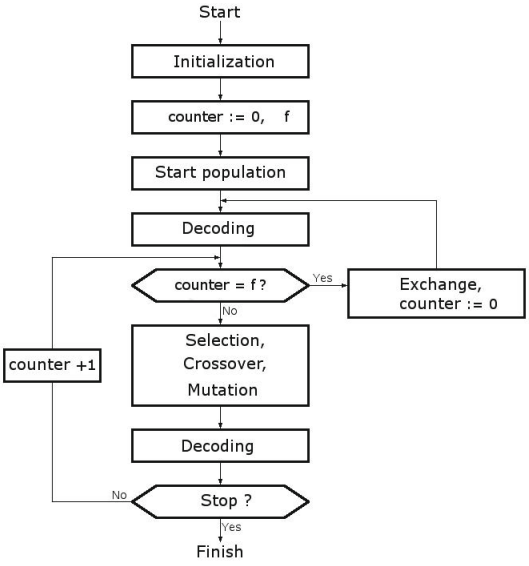


**Fig. 4.** Scheme of coevolutionary algorithm

A block scheme algorithm, for a single process, was illustrated in Figure 4. All stages of its operation, apart from an exchange step, are accomplished by the given processor independently of the others. The algorithm can also be run in a sequential version and in such a case, it is treated as a parallel algorithm with a single process (the exchange is omitted).

In the first step (Initialization) there are entered data concerning the problem and the initial values are given to necessary variables. Then, each process creates its own starting subpopulation of random individuals. For all chromosomes of each initial subpopulation there is a separate decoding decoding procedure. In the next step, there are two possible situations: (1) $counter \neq f$, which means that there was no moment of exchange of individuals between processes and there are genetic operations executed, such as Selection, Crossover and Mutation, after which the new or modified individuals are subject to decoding, (2) $counter = f$, and therefore there is a migration of individuals between the individual threads, and if necessary it leads to Decoding.

In the Selection step, each process examines the average quality of solutions of its subpopulation and rejects all having the value of the adaptation function which is below average. In the algorithm there have been implemented two types of Crossover operator: one point and two point with the use of PMX method. In most cases, parents are chosen from a pool of individuals that survived the selection process. However, with certain probability, the choice of parent from the group of rejected solutions may be possible, which is a one of the methods of prevention stagnation.

The mutation may occur in two ways. Since the sequence of boxes decoded in the chromosome is important for the decoding procedure, the first one is the random swapping of two fields with the numbers of boxes and the corresponding rotations. The second method changes the value of the rotation field in a single chromosome, therefore, the associated with it box has a changed rotation when trying to place it in a container by heuristic. The new value of the rotation may not be prohibited by the type to which the box belongs to. The exchange is a key element of the coevolutionary algorithm. In view of the fact that the algorithm was created with the cluster environment, it has been implemented with the use of MPI library. The processes are combined in pairs, in which communication occurs. Each of the two processors emits part of the sub-populations and sends it to another thread. The solutions sent are replaced with the new arrivals. The communication between processors is a non-locking one (functions: *MPI_Isend()*,*MPI_Irecv())*, and groups of individuals are transferred between them in the form of compressed data packets (functions: *MPI_Pack()*,*MPI_Unpack())* ). The main task of the Exchange, is the diversification of the various sub-populations.

## 4 Computational Experiments

Calculations were conducted on the Supernova cluster placed in the Wrocław Centre for Networking and Supercomputing, which consists of 573 nodes and

6368 computation cores. Benchmark problem instances were generated basing on procedure proposed in [1]. Fourteen test instances were used in computational experiments, for the number of possible boxes from 3 to 20, with the constant size of the contatiner. The algorithm was executed up to 20 times for each instances:

- sequentially on 1 processor,
- parallelly, with using 2,4 and 8 processors.

Each processor was placed in separate node of the cluster, with 2GB of the local RAM memory per node.

**Table 1.** Results of computations for 1 and 2 processors

| problem | types | 1 processor | | | 2 processors | | |
|---|---|---|---|---|---|---|---|
| | | $F_{\text{aver}}$[%] | $F_{\text{best}}$[%] | $t_{\text{aver}}$[min] | $F_{\text{aver}}$[%] | $F_{\text{best}}$[%] | $t_{\text{aver}}$[min] |
| bench1_03 | 3 | 72.0 | 72.7 | 80.4 | 72.9 | 74.2 | 65.8 |
| bench1_31 | 3 | 77.3 | 79.7 | 209.6 | 76.9 | 79.4 | 176.4 |
| bench2_01 | 5 | 79.7 | 80.7 | 31.4 | 80.1 | 83.4 | 27.5 |
| bench2_02 | 5 | 79.7 | 82.3 | 85.8 | 79.2 | 81.2 | 68.5 |
| bench3_01 | 8 | 78.8 | 82.0 | 75.3 | 80.0 | 82.9 | 43.1 |
| bench3_25 | 8 | 74.1 | 75.8 | 160.0 | 74.6 | 76.9 | 124.5 |
| bench4_01 | 10 | 78.2 | 80.1 | 53.7 | 78.5 | 81.4 | 39.7 |
| bench4_02 | 10 | 77.6 | 79.9 | 185.2 | 77.9 | 79.4 | 157.1 |
| bench5_04 | 12 | 76.5 | 79.6 | 234.2 | 76.5 | 78.8 | 191.9 |
| bench5_17 | 12 | 77.6 | 78.6 | 128.8 | 78.1 | 80.5 | 38.1 |
| bench6_01 | 15 | 77.4 | 78.1 | 146.0 | 76.7 | 78.5 | 107.7 |
| bench6_02 | 15 | 76.7 | 78.3 | 257.8 | 76.7 | 78.2 | 112.6 |
| bench7_01 | 20 | 77.0 | 78.5 | 69.8 | 77.1 | 78.8 | 45.1 |
| bench7_02 | 20 | 76.6 | 77.9 | 119.2 | 76.3 | 78.7 | 83.3 |
| **Average** | | **77.1** | **78.9** | **131.2** | **77,3** | **79.5** | **91.3** |

Tables 1 and 2 show results of computations for different number of processors. $F_{\text{aver}}$ and $F_{\text{best}}$ denote the average and the best found adaptation (cost) function value for each benchmark instance, respectively. The average time of computations for each instance is denoted by $t_{\text{aver}}$. As we can see, quality of solutions measured by the average and the best found solution increase with growing of the number of processors used.

Table 3 presents average speedups comparison. Growing of the speedup can observed for the processor number increasing. For test instances bench6_02 and bench5_17 for 2 processors, as well as for bench5_17 for 4 processors the superlinear speedup was observed. Detailed explaining of this anomaly will be an issue of the further research, it is probably connected with the specify of the cluster environment.

**Table 2.** Results of computations for 4 and 8 processors

| problem | types | 4 processors | | | 8 processors | | |
|---|---|---|---|---|---|---|---|
| | | $F_{\text{aver}}[\%]$ | $F_{\text{best}}[\%]$ | $t_{\text{aver}}[\text{min}]$ | $F_{\text{aver}}[\%]$ | $F_{\text{best}}[\%]$ | $t_{\text{aver}}[\text{min}]$ |
| bench1_03 | 3 | 72.6 | 75.0 | 50.0 | 71.7 | 72.7 | 13.8 |
| bench1_31 | 3 | 76.4 | 78.6 | 72.9 | 76.5 | 78.3 | 55.3 |
| bench2_01 | 5 | 79.9 | 84.0 | 12.5 | 79.2 | 81.3 | 8.5 |
| bench2_02 | 5 | 79.5 | 81.1 | 42.8 | 79.1 | 81.5 | 22.1 |
| bench3_01 | 8 | 78.9 | 82.3 | 33. | 78.2 | 79.1 | 17.9 |
| bench3_25 | 8 | 73.6 | 74.5 | 45.9 | 72.8 | 74.6 | 38.0 |
| bench4_01 | 10 | 78.5 | 82.1 | 27.3 | 77.4 | 79.4 | 15.0 |
| bench4_02 | 10 | 77.5 | 80.3 | 69.2 | 76.4 | 77.7 | 29.9 |
| bench5_04 | 12 | 76.3 | 79.0 | 71.3 | 75.5 | 78.1 | 49.8 |
| bench5_17 | 12 | 77.8 | 79.2 | 30.2 | 76.8 | 79.7 | 18.4 |
| bench6_01 | 15 | 77.1 | 79.0 | 51.0 | 75.9 | 78.2 | 41.2 |
| bench6_02 | 15 | 76.5 | 78.7 | 91.2 | 75.3 | 77.4 | 38.3 |
| bench7_01 | 20 | 76.0 | 78.2 | 39.7 | 76.0 | 77.2 | 39.8 |
| bench7_02 | 20 | 76.1 | 78.5 | 61.7 | 75.2 | 76.5 | 46.9 |
| **Average** | | **76.9** | **79.3** | **49.9** | **76.1** | **78.0** | **31.1** |

**Table 3.** Obtained speedups for a given processors number

| instance | 2 processors | 4 processors | 8 processors |
|---|---|---|---|
| bench1_03 | 1.22 | 1.61 | 5.83 |
| bench1_31 | 1.19 | 2.88 | 3.79 |
| bench2_01 | 1.14 | 2.51 | 3.69 |
| bench2_02 | 1.25 | 2.00 | 3.88 |
| bench3_01 | 1.75 | 2.23 | 4.21 |
| bench3_25 | 1.29 | 3.49 | 4.21 |
| bench4_01 | 1.35 | 1.97 | 3.58 |
| bench4_02 | 1.18 | 2.31 | 6.19 |
| bench5_04 | 1.22 | 3.28 | 4.70 |
| bench5_17 | 3.38 | 4.26 | 7.00 |
| bench6_01 | 1.36 | 2.86 | 3.54 |
| bench6_02 | 2.29 | 2.83 | 6.74 |
| bench7_01 | 1.55 | 1.76 | 1.76 |
| bench7_02 | 1.43 | 1.93 | 2.54 |
| **Average** | **1.54** | **2.57** | **4.40** |

## 5 Conclusions

Parallel coevolutionary algorithm for the three-dimensional bin packaging problem is proposed in the paper. Proposed algorithm is based on the island model, which makes possible of communication and solutions exchange between subpopulations. Parallelization of the algorithm gives an effect of processing time shortening (average time for all benchmark instances 131.2 seconds for 1 processor vs. 31.1 seconds for 8 processors) with comparable solutions quality.

## References

1. Bischoff, E.E., Ratcliff, M.S.W.: Issues in the Development of Approaches to Container Loading. Omega, Int. J. Mgmt. Sci. 23(4), 377–390 (1995)
2. Bortfeld, A., Gehring, H.: A hybryd genetic algorithm for the container loading problem. European Journal of Operational Research 131, 143–161 (2001)
3. Bożejko, W., Wodecki, M.: Parallel genetic algorithm for minimizing total weighted completion time. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 400–405. Springer, Heidelberg (2004)
4. Bożejko, W., Wodecki, M.: Parallel Evolutionary Algorithm for the Traveling Salesman Problem. Journal of Numerical Analysis, Industrial and Applied Mathematics 2(3-4), 129–137 (2007)
5. Bożejko, W., Wodecki, M.: Solving Permutational Routing Problems by Population-Based Metaheuristics. Computers & Industrial Engineering 57, 269–276 (2009)
6. He, Y., Wu, Y., de Souza, R.: A global search framework for practical three-dimensional packing with variable carton orientations. Computers & Operations Research 39(2012), 2395–2414 (2012)
7. Goncalves, J.F., Resende, M.G.C.: A parallel multi-population biased random-key genetic algorithm for a container loading problem. Computers & Operations Research 39, 179–190 (2012)
8. Hopper, E.: Two-dimensional Packing Utilising Evolutionary Algorithms and other Meta- Heuristic Methods, PhD Thesis, University of Wales (2000)
9. Karabulut, K., İnceoğlu, M.M.: A hybrid genetic algorithm for packing in 3D with deepest bottom left with fill method. In: Yakhno, T. (ed.) ADVIS 2004. LNCS, vol. 3261, pp. 441–450. Springer, Heidelberg (2004)
10. Kang, K., Moon, I., Wang, H.: A hybrid genetic algorithm with a n packing problem. Applied Mathematics and Computation 219, 1287–1299 (2012)
11. Liu, D.S., Tan, K.C., Huang, S.Y., Goh, C.K., Ho, W.K.: On solving multiobjective bin packing problems using evolutionary particle swarm optimization. European Journal of Operational Research 190, 357–382 (2008)
12. Stawowy, A.: Evolutionary based heuristic for bin packing problem. Computers & Industrial Engineering 55, 465–474 (2008)
13. Wu, Y., Li, W., Goh, M., de Souza, R.: Three-dimensional bin packing problem with variable bin height. European Journal of Operational Research 202, 347–355 (2010)