International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 6 (2017) pp. 1009-1019

Algorithms and Methods in Multidimensional Orthogonal Packing Problems

Vladislav A. Chekanin[1*] and Alexander V. Chekanin[2]

[1,2] Moscow State University of Technology «STANKIN»
Department of Theoretical Mechanics and Strength of Materials
3a Vadkovsky lane, Moscow, 127055, Russian Federation.

[*]Corresponding author
[1]ORCID: 0000-0001-9448-0583, Scopus Author ID: 55889934400
[2]ORCID: 0000-0003-0788-469X, Scopus Author ID: 55889047300

Abstract

The paper is devoted to developed algorithms and methods for managing of objects in solving orthogonal packing problems of any dimensions. For description of packing schemes we use a model of potential containers which allows controlling of all existing free spaces in containers. Fast placement of objects is provided by using of a multilevel linked data structure the depth of which is equal to dimension of a considered packing problem. We propose a method of deleting orthogonal objects in any dimensional orthogonal container which will be used in realization of rearranging packing procedures. All methods and algorithms presented in the paper provide the maximal fast and right formation of placement schemes on decoding solutions obtained with any algorithms used for optimization of the cutting and packing problems. The described algorithms and methods are realized in developed applied software which will be used in future research devoted to analysis of heuristic and metaheuristic algorithms intended for optimization of one-, two- and three-dimensional orthogonal packing and rectangular cutting problems of any types.

Keywords: orthogonal packing problem, data structure, optimization, software.

INTRODUCTION

All packing and cutting problems are combined into one class of classical problems of the mathematical theory of operational research [1]. This class of discrete optimization problems unites a large set of different practical NP-completed problems related with the searching of the most suitable schemes of placement of given sets of small items named by objects into sets of larger items named by containers. The first classification of the packing and cutting problems was presented by Dyckhoff [2]. Today the most fully classification of the packing and cutting problems is considered a classification offered by Wascher et al. [3] during the preparation of which were reviewed 445 articles devoted to different problems of resources allocation optimization.

Many cutting and packing problems take a place in solving actual optimization problems such as bin packing problem, knapsack problem, strip packing problem, cutting stock problem, trim loss problem, nesting problem, pallet and vehicle loading, assembly line balancing problem, partitioning problem, capital budgeting problem, computer memory allocation problem as well as many other problems in industry, engineering, computer science and economics [4–8]. Widespread practical application of these problems in a variety of areas makes urgent the development of effective methods for solving them. Optimization of the packing and cutting problems leads to more efficient usage of resources in practice.

Solution of a large set of practical optimization problems of cutting and packing, including resources saving problems in industry, optimization problems in logistics (including truck, ship and air cargo loading), scheduling and planning, comes down to solution of a orthogonal packing problem which is to find the most suitable placement of a given set of orthogonal objects into a set of orthogonal containers [4, 9–11]. All methods for orthogonal packing problems which considered in this paper are applicable for solving of all bin packing, cutting stock and knapsack problems dealt with the objects and containers in a form of orthogonal parallelepipeds of any dimension.

Problems with the dimension higher than three aside from only spatial dimensions additionally have often non-spatial dimensions as time for example. These problems take place as multidimensional orthogonal knapsack and packing problems usually in computing and scheduling [12–14]. The most popular in practice are orthogonal packing problems with the dimension no higher than three [3].

STATEMENT OF THE ORTHOGONAL PACKING PROBLEM

Consider the statement of the D-dimensional orthogonal packing problem in common case. Is given a set of N orthogonal containers (D-dimensional parallelepipeds) with

dimensions as follows:

$$\left\{W_j^1, W_j^2, \ldots, W_j^D\right\}, \quad j \in \{1, \ldots, N\}.$$

Also is given a set of n orthogonal objects (D-dimensional parallelepipeds) with dimensions as follows:

$$\left\{w_i^1, w_i^2, \ldots, w_i^D\right\}, i \in \{1, \ldots, n\}.$$

The goal is to find a placement of all objects into a minimum number of given containers under the following conditions of correct placement:

(1) all edges of objects and containers are parallel,

(2) all packed objects do not overlap with each other, i.e.

$$\forall j \in \{1, \ldots, N\}, \forall d \in \{1, \ldots, D\}, \forall i, k \in \{1, \ldots, n\}, i \neq k$$
$$\left(x_{ij}^d \geq x_{kj}^d + w_k^d\right) \vee \left(x_{kj}^d \geq x_{ij}^d + w_i^d\right),$$

(3) all packed objects are within the bounds of the containers, i.e.

$$\forall j \in \{1, \ldots, N\}, \forall d \in \{1, \ldots, D\}, \forall i \in \{1, \ldots, n\}$$
$$\left(x_{ij}^d \geq 0\right) \wedge \left(x_{ij}^d + w_i^d \leq W_j^d\right).$$

The statement of D-dimensional orthogonal packing problem includes specifying a load direction of containers as a priority selection list of the coordinate axes:

$$L_P = \{P_1; P_2; \ldots; P_D\}, \quad P_d \in [1; D] \forall d \in \{1, \ldots, D\}.$$

As an example on Fig. 1 are shown placement schemes obtained for all possible load directions of a three-dimensional orthogonal container.

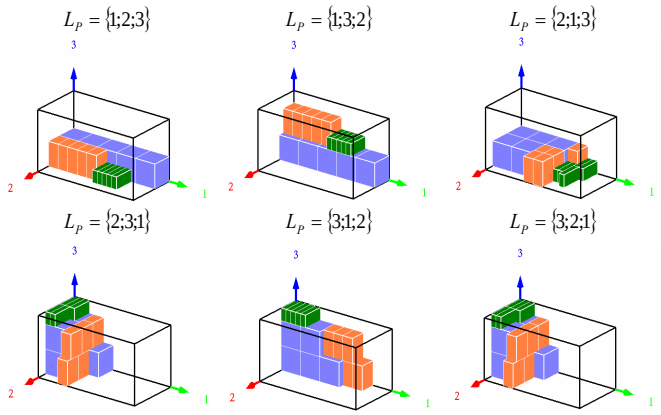

Figure 1. Load directions of a three-dimensional orthogonal container.

A solution of any dimensional packing problem is represented by a placement string (also known as a chromosome) which contains a sequence of objects selected for placing into containers. Before packing all the objects are classified by different types. All objects belonging to one type have the same geometrical and physical characteristics. Geometrical characteristics of objects include dimensions, physical – weight, color and etc.

The solving process of a packing problem includes three mandatory consecutive steps which are showed on the Fig. 2. On the first step is generated a placement string which contains a sequence of objects to be packed into containers. In practice the most commonly accepted methods of generation the solutions and the corresponding placement strings are multiobjective optimization algorithms, mainly the heuristic and metaheuristic algorithms including evolutionary algorithms [15–20].

Decoding of a placement string is performed by a corresponding decoder. It selects objects from a placement string one by one and places them into container on a particular algorithm. The quality of the resulting packing is estimated as the relative deviation of the solution from the lower bound of the considered problem [10].

The quality of a placement can be increased by its local improvement which performed by local replacing and deleting of some objects from containers with subsequent placing of them into them.

Obviously the major influence on the quality of a resulting packing render optimization algorithms that are applied for generation placement strings. Nevertheless time and speed of placement of objects from a given placement string are defined by used packing representation model as well as by algorithms used for generation a packing and for its improvement. We consider only algorithms for management objects which are used for decoding of placement strings and which provide the possibility of placing objects and their subsequent replacement in solving of any orthogonal packing problem of arbitrary dimension.

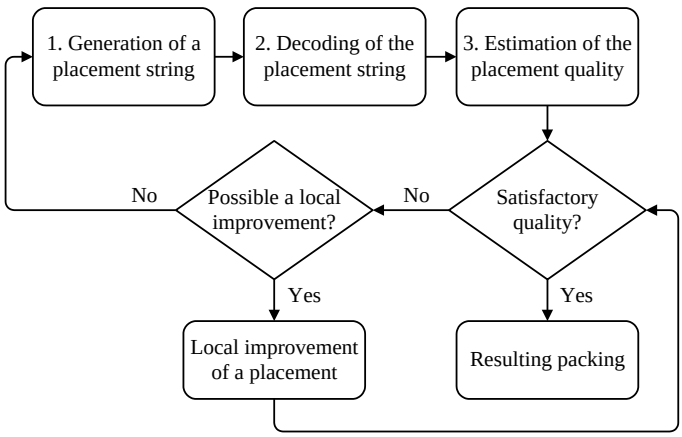

Figure 2. The solving process of a packing problem.

ALGORITHMS AND METHODS FOR MANAGEMENT OBJECTS

Packing representation model

To describe placement schemes of orthogonal objects in containers was chosen a packing representation model developed by authors of this paper – model of potential containers. Below is given a general description of this model.

Any placement scheme of objects in a container is represented by a set of objects attached to points (nodes) of a container [9, 10]. Under a potential container (PC) placed in a container at some its point is understood an imaginary orthogonal object with the largest possible dimensions that can be placed at this point without overlapping with any packed into the container object and edges of the container. Each potential container with number k is described with a vector

$$\left\{p_k^1; p_k^2; \ldots; p_k^D\right\},$$

containing dimensions of this potential container and a vector

$$\left\{x_k^1; x_k^2; \ldots; x_k^D\right\},$$

containing coordinates of a point of the potential container which is nearest to the origin of the container its containing.

All existing free orthogonal spaces located in a container are described by a set of potential containers. When a new object is put into a container it is necessary to verify the correctness of the placement. The model of potential containers guarantees the correct placement of an object if this object overlaps no borders of the potential container in which it is located. In this case when an object is put at some point of a container instead of checking on the intersection with all placed into the container objects is required to check only one condition of placement of this object entirely within the potential container, located at this point. This ensures a higher speed of placement objects on decoding a placement string. The condition of correct placement of a D-dimensional object i in a potential container k is expressed with the inequality:

$$\left(w_i^d \le p_k^d\right) \forall d \in \{1, \ldots, D\}.$$

Algorithm of placing objects

When an object is put into a potential container it divides this potential container on a set of smaller potential containers.

Theorem 3.1. No more than $2D$ of new potential containers are formed in a D-dimensional potential container when a D-dimensional orthogonal object is put into it or overlaps it.

Proof. Each face of an object cuts off a new space from an original potential container. Because the total number of faces in a D-dimensional orthogonal object is equal $2D$, hence the maximal number of new formed potential containers is also equal $2D$. □

In common case all new potential containers formed in a potential container k when a D-dimensional orthogonal object i is packed into it can be separated into two sets:

(1) a set of D potential containers with the dimensions

$$\left\{p_k^1; p_k^2; \ldots; p_k^{d-1}; x_i^d - x_k^d; p_k^{d+1}; \ldots, p_k^D\right\}$$

located at an origin of coordinates of the original potential container k:

$$\left\{x_k^1; x_k^2; \ldots; x_k^d; \ldots; x_k^D\right\}$$

which are produced under the following conditions of overlap:

$$x_i^d > x_k^d \text{ and } x_i^d < x_k^d + p_k^d \quad \forall d \in \{1, \ldots, D\};$$

(2) a set of D potential containers with the dimensions

$$\left\{p_k^1; p_k^2; \ldots; p_k^{d-1}; x_k^d + p_k^d - x_i^d - w_i^d; p_k^{d+1}; \ldots; p_k^D\right\}$$

located at D points with coordinates

$$\left\{x_k^1; x_k^2; \ldots; x_k^{d-1}; x_i^d + w_i^d; x_k^{d+1}; \ldots; x_k^D\right\}$$

which are produced under the following conditions of overlap:

$$x_i^d + w_i^d > x_k^d \text{ and } x_i^d + w_i^d < x_k^d + p_k^d \quad \forall d \in \{1, \ldots, D\}.$$

As an example on Fig. 3 are shown all potential containers which are formed in a three-dimensional orthogonal container (by dots on the figure are marked origins of the coordinates of corresponding new potential containers).

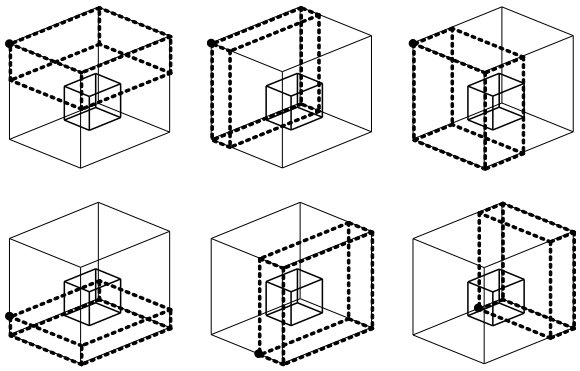

Figure 3. New potential containers formed in a three-dimensional container.

The algorithm which decodes a placement string provides placement of all objects into containers according to a given priority selection list. Below are given all steps of the decoding algorithm for the model of potential containers.

(1) Select the next object i from a placement string. If all the objects are selected, jump to step 4. Select a first not fully filled container j containing at least one potential container.

(2) In the current container j produce a serial procedure

of finding a potential container *k* nearest to origin of coordinates of the container *j* that is possible to correctly put into it the current object *i*. If the target potential container is not found then select the next container $j := j+1$ and retry step 2. If the target potential container was not found among all the containers, jump to step 1.

(3) Pack the object *i* into the found potential container *k* of the container *j*. Create a set of new potential containers and perform a procedure of search and remove of embedded potential containers. Sort all potential containers by decreasing the priority of pack of new objects to them, i.e. for any potential container *k* of the container *j* the following inequality must be satisfied:

$$\sum_{h=1}^{D}\left(x_k^{P_h}\prod_{d=P_{h+1}}^{D}W_j^d\right)\le\sum_{h=1}^{D}\left(x_{k+1}^{P_h}\prod_{d=P_{h+1}}^{D}W_j^d\right). \qquad (3.1)$$

Jump to step 1.

(4) End of the algorithm.

Fully the decoding algorithm is shown on Fig. 4. This adapted to the model of potential containers algorithm provides forming of a packing by a given placement string.

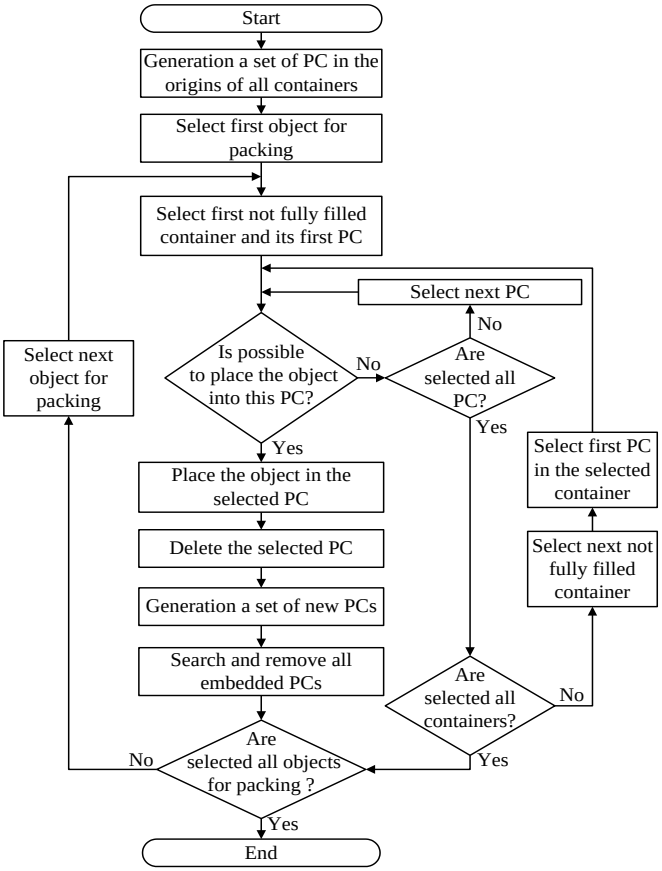

Figure 4. Decoding algorithm designed to the model of potential containers.

One of the major parts of the algorithm of placing objects is an algorithm of search and removal of embedded potential containers that is performed after placing of each object into an orthogonal container. This algorithm allows escaping from all potential containers which are embedded each to other. The potential container k_1 is embedded into the potential container k_2 (i.e. it is contained entirely within the potential container k_2) if the conditions:

$$x_{k_1}^d \ge x_{k_2}^d \quad \text{and} \quad x_{k_1}^d + p_{k_1}^d \le x_{k_2}^d + p_{k_2}^d \quad \forall d \in \{1,\ldots,D\}. \qquad (3.2)$$

The algorithm of search and removal of embedded potential containers includes the following steps.

(1) Generate in a current container *j* after placing into it an object *i* a list $\{L_0\}$ consisting of potential containers *k* that satisfy condition:

$$\exists d \in [1;D]: x_k^d \le x_i^d + w_i^d.$$

Generate an empty list of embedded potential containers $\{L_1\}$.

(2) Check Eq. (3.2) to determine the embedding of a potential container k_1 into k_2 for each pair of potential containers:

$$k_1, k_2 \in \{L_0\},\ k_1 \ne k_2 \quad \forall d \in \{1,\ldots,D\}.$$

Every found embedded potential container k_1 to copy to the list $\{L_1\}$.

(3) Remove all potential containers stored in the list $\{L_1\}$ from the contained them container *j*.

Storing the potential containers

The most time consuming step of the algorithm of placing objects is sorting of coordinates of all potential containers which runs after placing of each new object into a container to provide performing the Eq. (3.1).

To increase the time effectiveness of the algorithm of placing objects is proposed a new data structure – multilevel linked data structure. The basis of this data structure is the idea of presenting a set of coordinates of potential containers as a set of recursively embedded each to other linear queues that allows increasing the speed of access to potential containers during packing process.

The multilevel linked data structure presents a set *K* of potential containers located in points

$$\{x_k^1; x_k^2; \ldots; x_k^D\},\ k \in K$$

as a *D*-levels recursively embedded each to other linear queues which are ordered by increase of their items (Fig. 5).

Each item j of a queue i on a level P_d contains a coordinate

$$s_{i,j}^{P_d} = x_k^{P_d}$$

of a such potential container k that within each queue is satisfied the inequality:

$$s_{i,j}^{P_d} < s_{i,j+1}^{P_d} \forall P_d \in L_P .$$

$$\begin{aligned} P_d &: \quad s_{1,1}^{P_d} \to \ldots \to s_{1,i}^{P_d} \to \ldots \\ &\qquad\qquad\quad \downarrow \\ P_{d+1} &: \qquad\quad s_{i,1}^{P_{d+1}} \to \ldots \to s_{i,j}^{P_{d+1}} \to \ldots \\ &\qquad\qquad\qquad\quad \downarrow \\ P_{d+2} &: \qquad\qquad\quad s_{j,1}^{P_{d+2}} \to \ldots \to s_{j,k}^{P_{d+2}} \to \ldots \end{aligned}$$

Figure 5. Ordered lists in multilevel linked data structure.

As an example we consider a set of points of a three-dimensional orthogonal container (parallelepiped) j that is given in Table 1. For a load direction $L_P = \{1;2;3\}$ this set of points must first be ordered by non-decreasing along the coordinate axis 1, then – along the axis 2, and finally – along the axis 3 (see Table 2).

Table 1. Original coordinates of points of a container

Number of a point (k)	1	2	3	4	5	6	7	8	9	10
Coordinate (axis 1)	0	2	2	2	4	0	4	0	4	2
Coordinate (axis 2)	1	3	7	9	1	2	1	2	1	7
Coordinate (axis 3)	0	1	5	6	2	1	3	3	1	2

Table 2. Coordinates of points after sorting for the load direction {1;2;3}

Number of a point (k)	1	2	3	4	5	6	7	8	9	10
Coordinate (axis 1)	0	0	0	2	2	2	2	4	4	4
Coordinate (axis 2)	1	2	2	3	7	7	9	1	1	1
Coordinate (axis 3)	0	1	3	1	2	5	6	1	2	3

The multilevel linked data structure for a load direction $L_P = \{1;2;3\}$ is shown on Fig. 6.

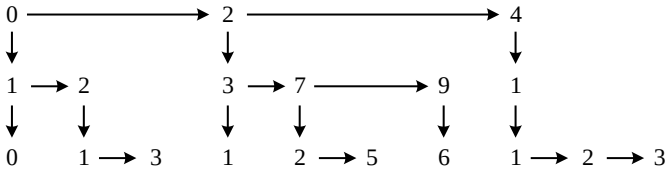

Figure 6. Multilevel linked data structure of the three-dimensional container for a load direction {1;2;3}.

After placing a new object into a container to provide the performing of the Eq. (3.1) is necessary only put coordinates of new formed potential containers in the corresponding positions of the multilevel linked data structure without sorting of coordinates of all existing potential containers.

One of the major obvious advantages of the multilevel linked data structure is an independence of time for packing of one object on the total number of packed objects. This data structure is the most effective to place a large number of objects of several slightly different in size object types. Fig. 7–10 show the relative placement time of a set of three-dimensional objects of five different types ($10 \times 10 \times 10$, $20 \times 20 \times 20$, $10 \times 40 \times 20$, $30 \times 5 \times 10$, $5 \times 10 \times 15$) uniformly distributed among the total number of objects (which takes values 25, 50, 100, 150, 200, 250, 300, 400, 800, 1600, 3200, 6400, 12800) into three-dimensional containers with the dimensions $200 \times 1000 \times 1000$, $500 \times 800 \times 800$, $1000 \times 400 \times 400$ and $2000 \times 200 \times 200$, respectively. The time efficiency is estimated by relative placement time which is calculated as the ratio of the total time spent on obtaining a placement scheme of all objects to their total number. This time characterizes the average time spent on placement of one object into a container. Time efficiency of the multilevel linked data structure is stabilized with growing the total number of packed objects.

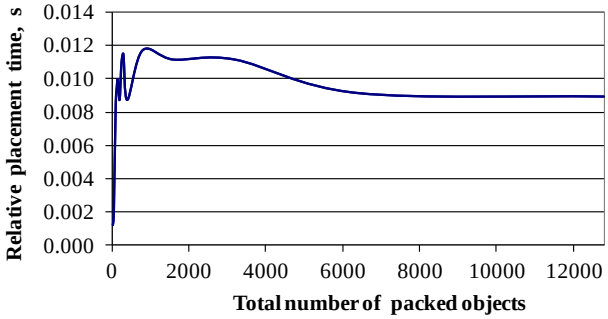

Figure 7. Relative placement time of packing into container with dimensions $200 \times 1000 \times 1000$.

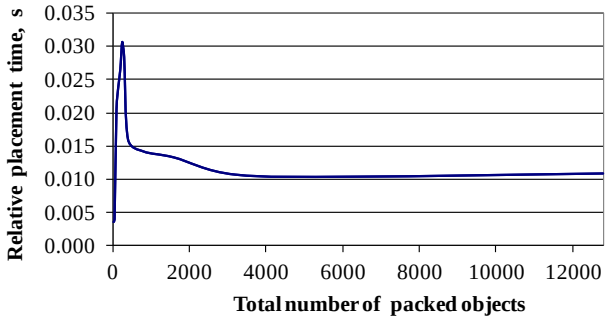

Figure 8. Relative placement time of packing into container with dimensions $500 \times 800 \times 800$.

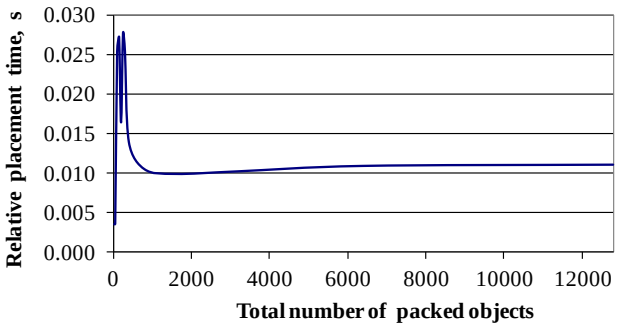

Figure 9. Relative placement time of packing into container with dimensions 1000 × 400 × 400.

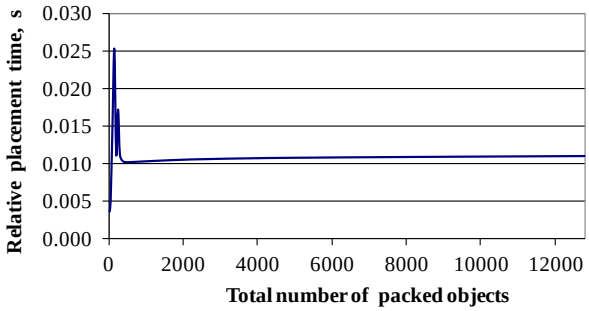

Figure 10. Relative placement time of packing into container with dimensions 2000 × 200 × 200.

1.1. *Algorithm of deleting objects*

One of the effective methods applied for increasing the quality of a resulting packing is its local improvement, which can be realized by deleting of one or more packed objects from a container with a subsequent more rational filling of the freed space by other objects. When an object is deleted from a container it is necessary to reorganize all potential containers around this object.

The developed algorithm of deleting a *D*-dimensional orthogonal object *i* from a *D*-dimensional orthogonal container *j* includes the following steps.

(1) Create a new free *D*-dimensional orthogonal container j' with the dimensions equal to the dimensions of the original container *j*.

(2) Put into the container j' an object i' with the dimensions

$$w_{i'}^d = w_i^d \quad \forall d \in \{1,\ldots,D\}$$

at a point with the coordinates equal to coordinates of the object *i* placed into the container *j*:

$$x_{i'}^d = x_i^d \quad \forall d \in \{1,\ldots,D\}.$$

(3) Create a list $\{L'\}$ containing potential containers which positions and dimensions can be modified in container *j* after deleting the object *i*. This list includes all potential containers k' under the condition

$$x_{k'}^d \le x_i^d + w_i^d \quad \forall d \in \{1,\ldots,D\}. \tag{3.3}$$

(4) Put into the container j' at points $x_{k'}^d$ a set of objects with the dimensions

$$w_{k'}^d = p_{k'}^d \quad \forall d \in \{1,\ldots,D\}, \quad k' \in \{L'\}. \tag{3.4}$$

When objects are placed into the container j' is necessary to allow them overlap each other. Free orthogonal spaces remained in the container j' are described by a set of potential containers placed in a list $\{L''\}$.

(5) Create a new free *D*-dimensional orthogonal container j'' with the dimensions equal to the dimensions of the original containers *j*.

(6) Put into the container j'' at points $x_{k''}^d$ a set of objects with the dimensions

$$w_{k''}^d = p_{k''}^d \quad \forall d \in \{1,\ldots,D\}, \quad k'' \in \{L''\}. \tag{3.5}$$

When objects are placed into the container j'' is necessary to allow them overlap each other. Free orthogonal spaces remained in the container j'' are described by a set of potential containers placed in a list $\{L'''\}$. This list of potential containers also describes a freed space of the original container *j* formed in the area of the object *i* which is to be deleted.

(7) Delete the object *i* from the container *j*.

(8) Change in the container *j* the list of its potential containers $\{L'\}$ to the obtained list of potential containers $\{L'''\}$.

As an example we consider a rectangular two-dimensional container with the dimensions $W^1 = W^2 = 100$ showed on Fig. 11. Parameters of packed objects and potential containers are given in Table 3 and Table 4, respectively.

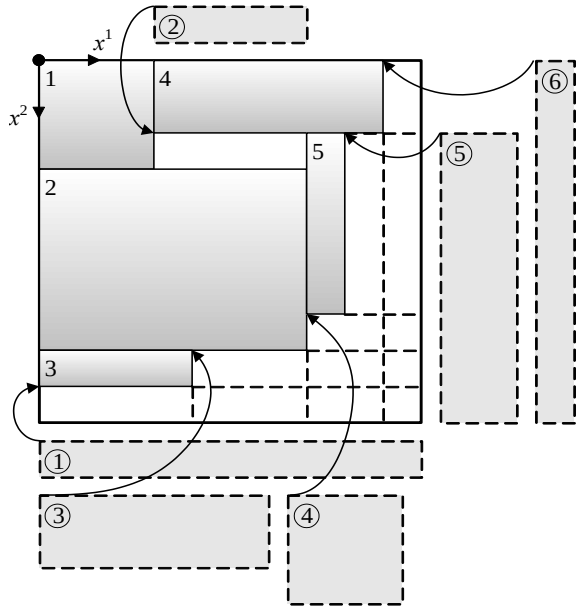

Figure 11. Container 1 before deleting an object with number 2.

Table 3. Objects packed into the container 1

Number	Coordinate 1	Coordinate 2	Dimension 1	Dimension 2
1	0	0	30	30
2	0	30	70	50
3	0	80	40	10
4	30	0	60	20
5	70	30	10	50

Table 4. Potential containers located in the container 1

Number	Coordinate 1	Coordinate 2	Dimension 1	Dimension 2
1	0	90	100	10
2	30	20	40	10
3	40	80	60	20
4	70	70	30	30
5	80	20	20	80
6	90	0	10	100

When an object with the number 2 is deleted necessary to reorganize all potential containers from the list $\{L'\}$ under the condition Eq. (3.3), i.e. potential containers with the numbers 2, 3 and 4 (see Table 4).

The object 2 as well as objects with the dimensions satisfying the Eq. (3.4) which are given in Table 5, are placed into a new container 1'. In this container is formed a list of potential containers $\{L''\}$ which is given in Table 6 and shown on Fig. 12.

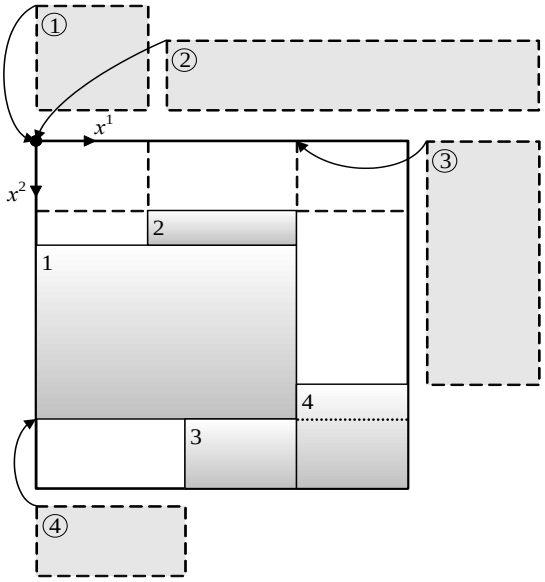

Figure 12. Container 1'.

Table 5. Objects packed into the container 1'

Number	Coordinate 1	Coordinate 2	Dimension 1	Dimension 2
1	0	30	70	50
2	30	20	40	10
3	40	80	60	20
4	70	70	30	30

Table 6. Potential containers located in the container 1'

Number	Coordinate 1	Coordinate 2	Dimension 1	Dimension 2
1	0	0	30	30
2	0	0	100	20
3	70	0	30	70
4	0	80	40	20

All objects with the dimensions satisfying the Eq. (3.5) are given in Table 7. These objects are placed in a new container with number 1" (see Fig. 13). In this container is formed a set of potential containers $\{L'''\}$ given in Table 8.

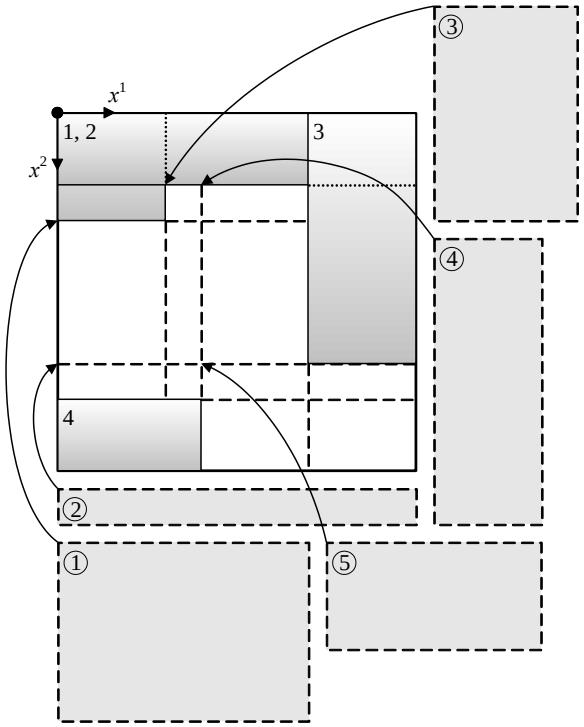

Figure 13. Container 1".

Table 7. Objects packed into the container 1"

Number	Coordinate 1	Coordinate 2	Dimension 1	Dimension 2
1	0	0	30	30
2	0	0	100	20
3	70	0	30	70
4	0	80	40	20

Table 8. Potential containers located in the container 1"

Number	Coordinate 1	Coordinate 2	Dimension 1	Dimension 2
1	0	30	70	50
2	0	70	100	10
3	30	20	40	60
4	40	20	30	80
5	40	70	60	30

After deleting the object 2 from the container 1 is necessary to replace all its potential containers from the list $\{L'\}$ (potential containers with numbers 2, 3 and 4 in Table 4) to potential containers from the list $\{L'''\}$ (given in Table 8). All the potential containers describing the free space remaining after deleting of the object are given in Table 9 and shown on the Fig. 14.

Table 9. Potential containers located in the container 1 after deleting the object

Number	Coordinate 1	Coordinate 2	Dimension 1	Dimension 2
1	0	30	70	50
2	0	70	100	10
3	0	90	100	10
4	30	20	40	60
5	40	20	30	80
6	40	70	60	30
7	80	20	20	80
8	90	0	10	100

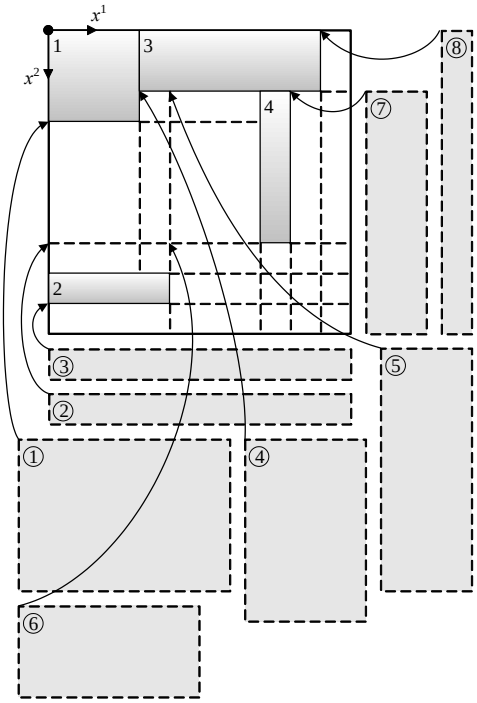

Figure 14. Container 1 after deleting the object.

The algorithm of deleting in result provides a set of potential containers which fully describes all freely orthogonal spaces of a container after deleting an object from it. This algorithm is applicable in solving of any orthogonal packing problem of arbitrary dimension.

COMPUTATIONAL EXPERIMENTS

This article contains algorithms for management objects the most of which were approved at several International conferences with publication of results of passed computation experiments in the corresponding papers. Below are given the major results of the performed computational experiments with the links on papers containing all details of the experiments.

The model of potential containers provides for four of six classes of three-dimensional orthogonal test problems a higher density of packing compared with the heuristic algorithm for the placement with the Extreme Points rule used the graph model [10] and with the heuristic algorithm proposed by Lodi et al. [19] for the node model [21].

The proposed multilevel linked data structure in practice more than twice increases access to the potential containers compared with the simple linear linked list which needs in sorting [22]. This data structure is the most effective when it is using for packing of a large number of objects of several slightly different in size object types. Efficiency of application of the multilevel linked data structure increases with the number of packed objects on solving as two-dimensional and three-dimensional orthogonal packing problems [23, 24].

SOFTWARE FOR ORTHOGONAL PACKING PROBLEMS

All the described algorithms were implemented in a class library for solving the cutting and packing problems. The UML diagram of the the class library realized with the high-level object-oriented programming language C++ is shown on Fig. 15.

On the basis of the designed class library was developed an applied software intended to solve the optimization orthogonal packing and rectangular cutting problems [25]. The software is intended for solving the following problems:

- one-dimensional bin packing problem (1DBPP),
- rectangular non-guillotine cutting problem,
- strip packing problem (SPP),
- two-dimensional bin packing problem (2DBPP),
- three-dimensional bin packing problem (3DBPP).

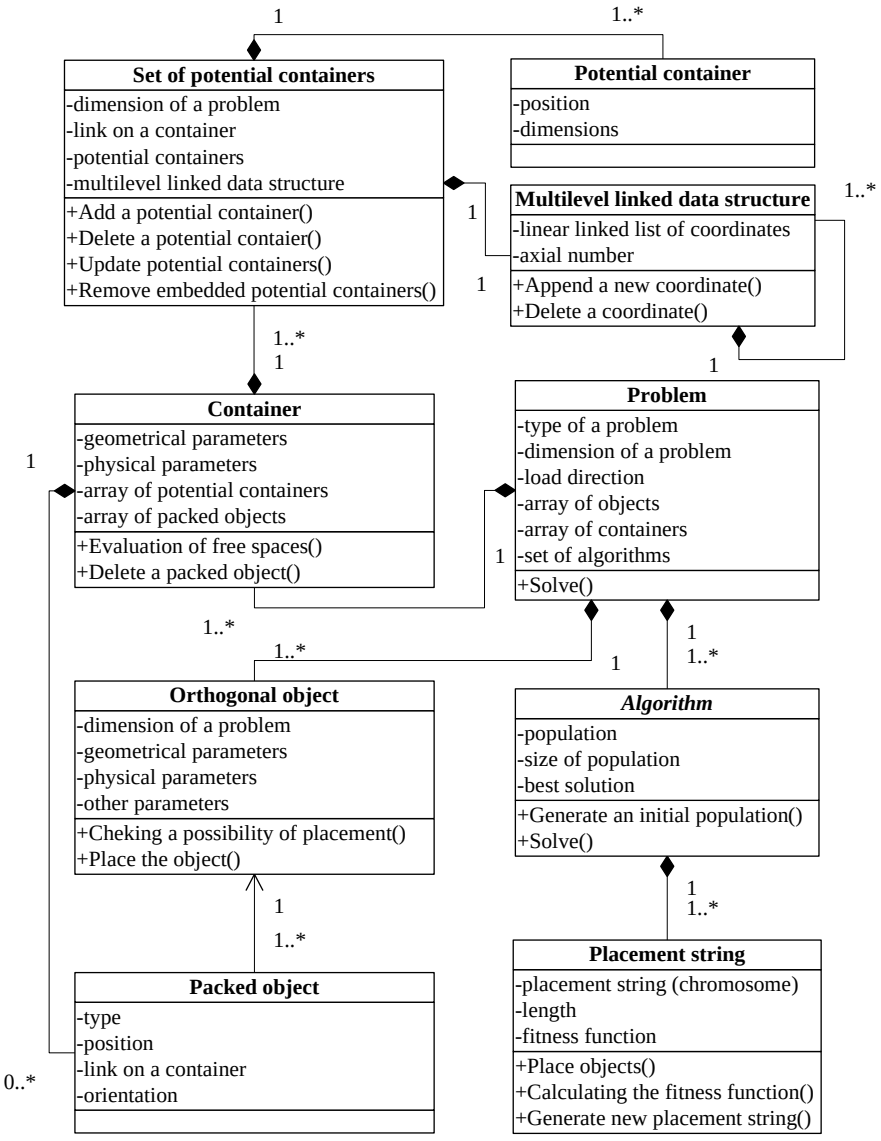

Figure 15. UML diagram of the developed class library for orthogonal packing problems.

The developed software contains a library of standard test instances for a variety of orthogonal packing and cutting problems. This library includes the following standard problem instances:

- SPP instances (classes C1-C6) from Berkey and Wang [26],
- SPP instances (classes C7-C10) from Martello and Vigo [27],
- 2DBPP instances form Fekete and Schepers [28],
- 3DBPP instances from Martello et al. [11].

The main window of the packing software demonstrates placement schemes and contains a list of all containers as well as a list of all potential containers of a selected container, as it shown on Fig. 16.

Today this software we use in carrying out computational experiments on developed algorithms and methods of solving orthogonal packing problems. The prospective area of the further research in this field includes expanding of a list of standard cutting and packing problems supported by the developed software.

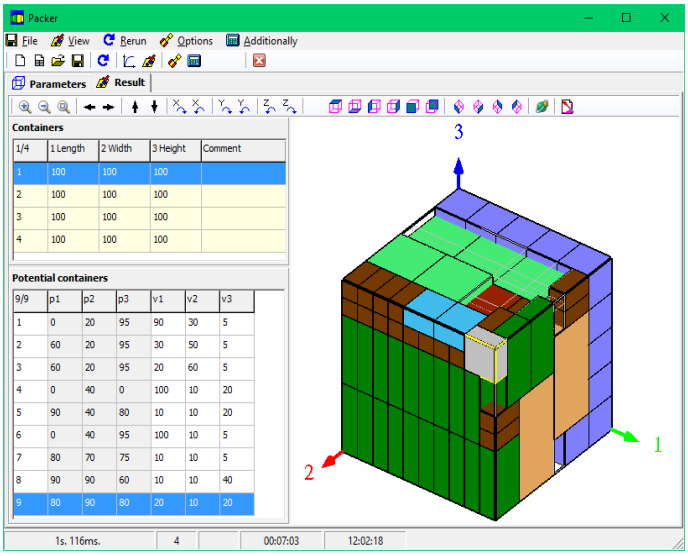

Figure 16. Main window of the developed packing software.

CONCLUSION

In this paper we have presented a set of algorithms for management objects in multidimensional orthogonal packing problems. The described algorithms and methods have the following obvious advantages:

- possibility of solving of any types of orthogonal packing or rectangular cutting problems,
- possibility of packing in any load directions,
- possibility of solving orthogonal packing problems of arbitrary dimensions,
- full description of all free spaces in containers allows accurately estimate its real possibilities for packing,
- possibility of any manipulations with objects during local improvement of a result packing,
- fast placing of objects into containers due to usage the multilevel linked data structure.

The algorithm of deleting objects will be applied in algorithms of local improvement of resulting packing which are to be developed in our future research.

All the developed algorithms realized in the designed software will be used by us in analysis and creation of new heuristic and metaheuristic algorithms for optimization the orthogonal packing and cutting problems.

ACKNOWLEDGMENTS

This work was carried out with the financial support of the Ministry of Education and Science of Russian Federation in the framework of the state task in the field of scientific activity of MSTU "STANKIN" (No. 1.7706.2017, basic part).

REFERENCES

[1] Johnson, D. S., 2012, "A Brief History of NP-Completeness, 1954–2012," Documenta Mathematica, Extra Volume ISMP, pp. 359–376.

[2] Dyckhoff, H., 1990, "A typology of cutting and packing problems," Eur. J. Oper. Res., 44, pp. 145–159.

[3] Wascher, G., Haubner, H., and Schumann, H., 2007, "An improved typology of cutting and packing problems," Eur. J. Oper. Res., 183(3), pp. 1109–1130.

[4] Bortfeldt, A., and Wascher, G, 2013, "Constraints in container loading – a state-of-the-art review," Eur. J. Oper. Res., 229(1), pp. 1–20.

[5] Boschetti, M. A., 2014, "New lower bounds for the finite three-dimensional bin packing problem," Discrete Appl. Math., 140, pp. 241–258.

[6] Gao, Y. Q., Guan, H. B., Qi, Z. W., Hou, Y., and Liu, L., 2013, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," J. Comput. Syst. Sci., 79(8), pp. 1230–1242.

[7] Leung, S. C. H., Zhang, D. F., Zhou, C. L., and Wu, T., 2012, "A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem," Comput. Oper. Res., 39(1), pp. 64–73.

[8] Lodi, A., Martello, S., and Monaci, M., 2002, "Two-dimensional packing problems: A survey," Eur. J. Oper. Res., 141(2), pp. 241–252.

[9] Chekanin, V. A., and Chekanin, A. V., 2013, "Efficient Algorithms for Orthogonal Packing Problems," Comp. Math. Math. Phys., 53(10), pp. 1457–1465.

[10] Crainic, T. G., Perboli, G., and Tadei, R., 2008, "Extreme point-based heuristics for three-dimensional bin packing," Informs J. Comput., 20(3), pp. 368–384.

[11] Martello, S., Pisinger, D., and Vigo, D., 2000, "The three-dimensional bin packing problem," Oper. Res., 48(2), pp. 256–267.

[12] Fekete, S. P., Schepers, J., and van der Veen, J. C., 2007, "An exact algorithm for higher-dimensional orthogonal packing," Oper. Res., 55(3), pp. 569–587.

[13] Lins, L., Lins, S., and Morabito, R., 2002, "An n-tet graph approach for non-guillotine packings of n-dimensional boxes into an n-container," Eur. J. Oper. Res., 141(2), pp. 421–439.

[14] Westerlund, J., Papageorgiou, L. G., and Westerlund, T., 2007, "A MILP model for N-dimensional allocation," Comput. Chem. Eng., 31(12), pp. 1702–1714.

[15] Chekanin, V. A., and Chekanin, A. V., 2014, "Development of the multimethod genetic algorithm for the strip packing problem," Applied Mechanics and Materials, 598, pp. 377–381.

[16] Chen, D., Liu, J., Fu, Y., and Shang, M., 2010, "An efficient heuristic algorithm for arbitrary shaped rectilinear block packing problem," Comput. Oper. Res., 37(6), pp. 1068–1074.

[17] Goncalves, J. F., and Resende, M. G. C., 2013, "A biased random key genetic algorithm for 2d and 3d bin packing problems," Int. J. Prod. Econ., 145(2), pp. 500–510.

[18] Kierkosz, I., and Luczak, M., 2014, "A hybrid evolutionary algorithm for the two-dimensional packing problem," Central European Journal of Operations Research, 22(4), pp. 729–753.

[19] Lodi, A., Martello, S., and Vigo, D., 2002, "Heuristic algorithms for the three-dimensional bin packing problem," Eur. J. Oper. Res., 141(2), pp. 410–420.

[20] Riff, M. C., Bonnaire, X., and Neveu, B., 2009, "A revision of recent approaches for two-dimensional strip-packing problems," Eng. Appl. Artif. Intel., 22(4–5), pp. 823–827.

[21] Chekanin, A. V., and Chekanin, V. A., 2013, "Improved packing representation model for the orthogonal packing problem," Applied Mechanics and Materials, 390, pp. 591–595.

[22] M. A. Weiss, 2014, Data Structures and Algorithm Analysis in C++, Pearson Education, Boston, USA.

[23] Chekanin, V. A., and Chekanin, A. V., 2014, "Multilevel linked data structure for the multidimensional orthogonal packing problem," Applied Mechanics and Materials, 598, pp. 387–391.

[24] Chekanin, A. V., and Chekanin, V. A., 2014, "Effective data structure for the multidimensional orthogonal bin packing problems," Advanced Materials Research, 962–965, pp. 2868–2871.

[25] Chekanin, V. A., and Chekanin, A. V., 2015, "Development of optimization software to solve practical packing and cutting problems," Advances in Intelligent Systems Research, 123, pp. 379–382.

[26] Berkey, J. O., and Wang, P. Y., 1987, "Two-dimensional finite bin-packing algorithms," J. Oper. Res. Soc., 38(5), pp. 423–429.

[27] Martello, S., and Vigo, D., 1998, "Exact solution of the two-dimensional finite bin packing problem," Manage. Sci., vol. 44(3), pp. 388–399.

[28] Fekete, S. P., and Schepers, J., 1998, "New classes of lower bounds for bin packing problems," Lect. Notes Comput. Sc., 1412, pp. 257–270.

ABOUT THE AUTHORS:

Chekanin Vladislav A. is Candidate of Science in Engineering (since 2011), Associate professor of the Sub-department of theoretical mechanics and strength of materials in Moscow State University of Technology «STANKIN». His interests are in information technologies, computer science and artificial intelligence. He has published more than 10 research papers in international scientific journals and more than 40 research papers in national journals and conference proceedings.

Chekanin Alexander V. is Doctor of Science in Engineering (since 1999), Full professor, Head of the sub-department of theoretical mechanics and strength of materials in Moscow State University of Technology «STANKIN». His interests are in information technologies, computer science, optimization methods and automation of strength calculations. He has published more than 100 research papers in international and national scientific journals and conference proceedings.