**Journal of Theoretical and Applied Information Technology**
30th June 2016. Vol.88. No.3
© 2005 - 2016 JATIT & LLS. All rights reserved·

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# IMPLEMENTATION OF PACKING METHODS FOR THE ORTHOGONAL PACKING PROBLEMS

**[1]CHEKANIN VLADISLAV A., [2]CHEKANIN ALEXANDER V.**
Moscow State University of Technology «STANKIN»,
Department Of Theoretical Mechanics And Strength Of Materials,
3a Vadkovsky lane, Moscow, 127055, Russian Federation
E-mail: [1]vladchekanin@rambler.ru, [2]avchekanin@rambler.ru

**ABSTRACT**

In the paper is considered a multidimensional orthogonal packing problem in general case. Are described packing methods developed for formation of placement schemes during solving the rectangular cutting and orthogonal packing problems of arbitrary dimensions. The presented packing model allows to describe all existing free spaces in containers. Are offered methods of placing and deleting of orthogonal objects, application of which in optimization packing algorithms will improve the quality of packing. Is described a new data structure providing increasing speed of packing formation. Efficiency of application of the packing methods is investigated on standard instances of three-dimensional orthogonal packing problems. All designed packing methods are realized in a form of an applied software which can be used in solving problems of resources allocation including container loading, cutting stock, scheduling, knapsack problems and many other practical important cutting and packing problems.

**Keywords:** *Packing, Orthogonal Packing Problem, Data Structure, Object-Oriented Programming, Applied Software*

## 1. INTRODUCTION

The optimization packing problems combine a large set of various practical problems related with finding of the most optimal allocation of resources of one type into resources of another type. The first type of the resources is named by objects and the second is named by containers. In general case in most of packing problems it is necessary to pack all given objects into minimal number of containers. Optimization of the packing problems leads to more efficient usage of resources in practice.

The extensive survey of packing problems was prepared in 2007 by G. Wascher, H. Haubner and H. Schumann. It was performed with consideration of more than four hundred papers related with different packing and cutting problems [1]. The most frequently the packing problems are occurred in solution of such practical problems as waste minimization in cutting, pallet building in logistics, filling up containers (including vehicle, air and ship cargo loading), traffic planning in computing and network systems, calendar planning, capital budgeting and many other important problems which deal with allocation and reallocation of resources [2–6].

Among the most popular types of packing problems in industry and manufacturing can be distinguished orthogonal bin packing problems, cutting stock problems and knapsack problems [2,5,7–9]. In these problems are considered containers and objects in a form of parallelepipeds.

Packing 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 [10–12]. The most popular in practice are orthogonal packing problems with the dimension no higher than three [1].

## 2. STATEMENT OF THE PROBLEM

Consider the statement of the orthogonal *D*-dimensional packing problem in general case. Is given a set of *N* orthogonal containers (*D*-dimensional parallelepipeds) with dimensions as follows: $\{W_j^1, W_j^2, \ldots, W_j^D\}$, $j \in \{1, \ldots, N\}$. Also is given a set of *n* orthogonal objects (*D*-dimensional parallelepipeds) with dimensions $\{w_i^1, w_i^2, \ldots, w_i^D\}$, $i \in \{1, \ldots, n\}$.

Before packing all the objects are classified by different types. All objects that are belong to one type have all identical geometrical and physical characteristics. Geometrical characteristics of objects include dimensions, physical include weight, color, fragility and etc. [13]

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

- all edges of objects and containers are parallel;
- all packed objects do not overlap with each other:

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

- all packed objects are within the bounds of the containers:

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

The statement of $D$-dimensional orthogonal packing problem includes specifying a load direction of containers as the priority selection list of the coordinate axes: $L_P = \{P_1; P_2; \ldots; P_D\}$, $P_d \in [1; D] \forall d \in \{1, \ldots, D\}$.

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

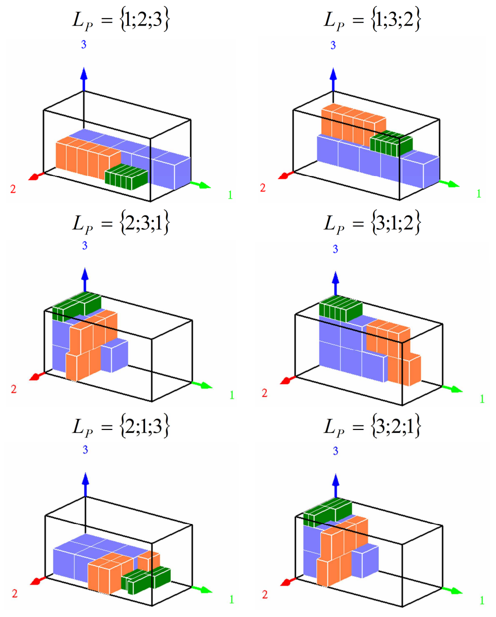


Figure 1: Load Directions Of A Three-Dimensional Orthogonal Container.

A solution of any dimensional packing problem is represented by a so-called placement string (also known as a chromosome) which contains a sequence of objects selected for placing into containers.

The solving process of a packing problem includes three mandatory consecutive steps which are showed on the figure 2.

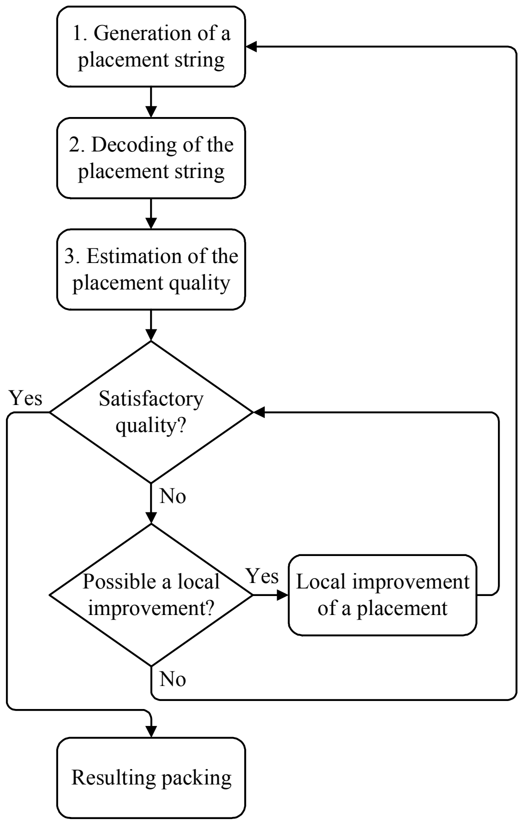

1. Generation of a placement string
2. Decoding of the placement string
3. Estimation of the placement quality
Satisfactory quality?
Yes
No
Possible a local improvement?
Yes
Local improvement of a placement
No
Resulting packing

Figure 2: The Solving Process Of A Packing Problem.

The first step is generation of a placement string which contains a sequence of objects to be packed into containers. In practice the most commonly accepted methods of generation of the solutions and the corresponding placement strings are the multiobjective optimization algorithms, mainly the heuristic and metaheuristic algorithms including evolutionary algorithms [14-19]. Need of application of such algorithms provides obtaining of suboptimal solutions is due to the fact that all optimization packing and cutting problems including orthogonal packing problems are belong to NP-completed problems so they cannot be solved in a time that polynomial depends on the dimension of the problem [20].

Decoding of a placement string is performed by a corresponding decoder. It selects objects from a placement string one by one and place 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 [8].

The quality of a placement can be increased by its local improvement which performed by local replacing and deleting of sets of objects with subsequent new placing of all deleted objects into freed spaces.

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

## 3. PACKING METHODS

### 3.1 Packing Representation Model

To describe placement schemes of orthogonal objects in containers was chosen previously developed packing representation model – model of potential containers. The efficiency of application of this model for orthogonal packing problems is shown in paper [21]. Below is given a general description of the model of potential containers.

Any placement scheme of objects in a container is represented by a set of objects attached to points (nodes) of a container [7,8]. We understand by a potential container placed in a container at some its point is 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 $\{p_k^1; p_k^2; \ldots; p_k^D\}$, containing dimensions of this potential container and a vector $\{x_k^1; x_k^2; \ldots; x_k^D\}$, containing coordinates of a point of the potential container which is closest to the origin of coordinates of the containing its container.

All existing free orthogonal spaces located in a container are described by a set of potential containers. When a object is put into a free space of a container is necessary to verify the correctness of this placement. This model guarantees the correct placement of an object if this object overlap 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 inequality: $\left(w_i^d \le p_k^d\right) \forall d \in \{1, \ldots, D\}$.

### 3.2 Placing Objects

When an object is put into a potential container it divides this potential container on a set of smaller potential containers. Any *D*-dimensional orthogonal potential container is divided at 2*D* potential containers in the worst case. Any new formed potential container has a common face with the packed object. Because the total number of faces in a *D*-dimensional orthogonal object is equal 2*D*, hence the maximal number of new formed potential containers is also equal 2*D*.

In common case when a *D*-dimensional orthogonal object *i* is packed into a potential container *k* all new potential containers formed in it can be separated into two sets [22]:

1. Set of *D* potential containers $\{p_k^1; p_k^2; \ldots; p_k^{d-1}; x_i^d - x_k^d; p_k^{d+1}; \ldots, p_k^D\}$ located at an origin of coordinates of the original potential container *k*: $\{x_k^1; x_k^2; \ldots; x_k^d; \ldots; x_k^D\}$ produced under the conditions of overlap: $\left(x_i^d > x_k^d\right) \wedge \left(x_i^d < x_k^d + p_k^d\right) \ \forall d \in \{1, \ldots, D\}$;
2. Set of *D* potential containers $p_k^1; p_k^2; \ldots; p_k^{d-1}; x_k^d + p_k^d - x_i^d - w_i^d; p_k^{d+1}; \ldots; p_k^D$ located at *D* points with coordinates $\{x_k^1; x_k^2; \ldots; x_k^{d-1}; x_i^d + w_i^d; x_k^{d+1}; \ldots; x_k^D\}$ which are produced under the following conditions of overlap: $\left(x_i^d + w_i^d > x_k^d\right) \wedge \left(x_i^d + w_i^d < x_k^d + p_k^d\right)$ $\forall d \in \{1, \ldots, D\}$.

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

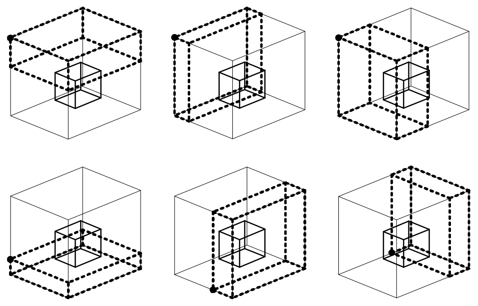


*Figure 3: New Potential Containers Formed In A Three-Dimensional Container.*

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 considered packing representation model [23].

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 this step. If the target potential container was not found among all the containers, jump to step 1.
3. Place the object $i$ into the found potential container $k$ in the container $j$. Create a set of new potential containers and perform a method of searching and removing 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:
$$\sum_{h=1}^{D}\left(x_k^{P_h}\prod_{d=P_{h+1}}^{D}W_j^d\right) \le \sum_{h=1}^{D}\left(x_{k+1}^{P_h}\prod_{d=P_{h+1}}^{D}W_j^d\right). \quad (1)$$
Jump to step 1.
4. End of the algorithm.

This adapted to the model of potential containers algorithm provides forming of a packing by a given placement string.

One of the major part of the algorithm of placing objects is a method of searching and removing of embedded potential containers that is performed after placing of each object into an orthogonal container. This algorithm allows to escape 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 wholly entirely within the potential container $k_2$) if $\forall d \in \{1,\ldots,D\}$ the conditions are:
$$\left(x_{k_1}^d \ge x_{k_2}^d\right) \wedge \left(x_{k_1}^d + p_{k_1}^d \le x_{k_2}^d + p_{k_2}^d\right). \quad (2)$$
The method of searching and removing of embedded potential containers includes the following steps.

1. Generate in a current container $j$ after placing into it an object $i$ a list $\{L_0\}$ containing a set of potential containers k if the condition is:
$$\exists d \in [1;D] : x_k^d \le x_i^d + w_i^d .$$
Generate an empty list of embedded potential containers $\{L_1\}$.
2. Check Eq. (2) to determine the embedding of a potential container $k_1$ into $k_2$ for each pair $k_1, k_2 \in \{L_0\},\ k_1 \ne k_2 \quad \forall d \in \{1,\ldots,D\}$. Copy all found embedded potential container $k_1$ to the list $\{L_1\}$.
3. Remove all potential containers stored in the list $\{L_1\}$ from the contained them container $j$.

## 3.3 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. (1).

To increase the time effectiveness of the algorithm of placing objects we offer for usage a proposed by us new data structure for packing problems – multilevel linked data structure that was firstly described in [23]. 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 to increase the speed of access to potential containers during packing process.

Using the multilevel linked data structure a set $K$ of potential containers located in points $\{x_k^1; x_k^2; \ldots; x_k^D\},\ k \in K$ is represented as a $D$-levels recursively embedded each to other linear queues which are ordered by increase of their items (see figure 4). 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 potential container $k$ that within each queue the inequality is satisfied [24]: $s_{i,j}^{P_d} < s_{i,j+1}^{P_d} \forall P_d \in L_P$.

$$\begin{tikzcd}
P_d : & s^{P_d}_{1,1} \arrow[r] & \ldots \arrow[r] & s^{P_d}_{1,i} \arrow[r] \arrow[d] & \ldots & & & & \\
P_{d+1} : & & & s^{P_{d+1}}_{i,1} \arrow[r] & \ldots \arrow[r] & s^{P_{d+1}}_{i,j} \arrow[r] \arrow[d] & \ldots & & \\
P_{d+2} : & & & & & s^{P_{d+2}}_{j,1} \arrow[r] & \ldots \arrow[r] & s^{P_{d+2}}_{j,k} \arrow[r] & \ldots
\end{tikzcd}$$

*Figure 4: Multilevel Linked Data Structure.*

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

*Table 1: Original Coordinates Of Points.*

| Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Coordinate 1 | 0 | 2 | 2 | 2 | 4 | 0 | 4 | 0 | 4 | 2 |
| Coordinate 2 | 1 | 3 | 7 | 9 | 1 | 2 | 1 | 2 | 1 | 7 |
| Coordinate 3 | 0 | 1 | 5 | 6 | 2 | 1 | 3 | 3 | 1 | 2 |

*Table 2: Coordinates Of Points After Sorting For The Load Direction {2;1;3}.*

| Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Coordinate 2 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 7 | 7 | 9 |
| Coordinate 1 | 0 | 4 | 4 | 4 | 0 | 0 | 2 | 2 | 2 | 2 |
| Coordinate 3 | 0 | 1 | 2 | 3 | 1 | 3 | 1 | 2 | 5 | 6 |

The multilevel linked data structures for all possible load directions of the considered container are shown on figures 5–10.

$$\begin{tikzcd}
0 \arrow[rrr] \arrow[d] & & & 2 \arrow[rrrr] \arrow[d] & & & & 4 \arrow[d] & & \\
1 \arrow[r] \arrow[d] & 2 \arrow[d] & & 3 \arrow[r] \arrow[d] & 7 \arrow[rr] \arrow[d] & & 9 \arrow[d] & 1 \arrow[d] & & \\
0 & 1 \arrow[r] & 3 & 1 & 2 \arrow[r] & 5 & 6 & 1 \arrow[r] & 2 \arrow[r] & 3
\end{tikzcd}$$

*Figure 5: Multilevel linked data structure of the three-dimensional container for a load direction {1;2;3}.*

$$\begin{tikzcd}
0 \arrow[rrr] \arrow[d] & & & 2 \arrow[rrrr] \arrow[d] & & & & 4 \arrow[d] & & \\
0 \arrow[r] \arrow[d] & 1 \arrow[r] \arrow[d] & 3 \arrow[d] & 1 \arrow[r] \arrow[d] & 2 \arrow[r] \arrow[d] & 5 \arrow[r] \arrow[d] & 6 \arrow[d] & 1 \arrow[r] \arrow[d] & 2 \arrow[r] \arrow[d] & 3 \arrow[d] \\
1 & 2 & 2 & 3 & 7 & 7 & 9 & 1 & 1 & 1
\end{tikzcd}$$

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

$$\begin{tikzcd}
1 \arrow[rrrr] \arrow[d] & & & & 2 \arrow[rr] \arrow[d] & & 3 \arrow[r] \arrow[d] & 7 \arrow[rr] \arrow[d] & & 9 \arrow[d] \\
0 \arrow[r] \arrow[d] & 4 \arrow[d] & & & 0 \arrow[d] & & 2 \arrow[d] & 2 \arrow[d] & & 2 \arrow[d] \\
0 & 1 \arrow[r] & 2 \arrow[r] & 3 & 1 \arrow[r] & 3 & 1 & 2 \arrow[r] & 5 & 6
\end{tikzcd}$$

*Figure 7: Multilevel linked data structure of the three-dimensional container for a load direction {2;1;3}.*

$$\begin{tikzcd}
1 \arrow[rrrr] \arrow[d] & & & & 2 \arrow[rr] \arrow[d] & & 3 \arrow[r] \arrow[d] & 7 \arrow[rr] \arrow[d] & & 9 \arrow[d] \\
0 \arrow[r] \arrow[d] & 1 \arrow[r] \arrow[d] & 2 \arrow[r] \arrow[d] & 3 \arrow[d] & 1 \arrow[r] \arrow[d] & 3 \arrow[d] & 1 \arrow[d] & 2 \arrow[r] \arrow[d] & 5 \arrow[d] & 6 \arrow[d] \\
0 & 4 & 4 & 4 & 0 & 0 & 2 & 2 & 2 & 2
\end{tikzcd}$$

*Figure 8: Multilevel linked data structure of the three-dimensional container for a load direction {2;3;1}.*

$$\begin{tikzcd}
0 \arrow[r] \arrow[d] & 1 \arrow[rrr] \arrow[d] & & & 2 \arrow[rr] \arrow[d] & & 3 \arrow[rr] \arrow[d] & & 5 \arrow[r] \arrow[d] & 6 \arrow[d] \\
0 \arrow[d] & 0 \arrow[r] \arrow[d] & 2 \arrow[r] \arrow[d] & 4 \arrow[d] & 2 \arrow[r] \arrow[d] & 4 \arrow[d] & 0 \arrow[r] \arrow[d] & 4 \arrow[d] & 2 \arrow[d] & 2 \arrow[d] \\
1 & 2 & 3 & 1 & 7 & 1 & 2 & 1 & 1 & 9
\end{tikzcd}$$

*Figure 9: Multilevel linked data structure of the three-dimensional container for a load direction {3;1;2}.*

$$\begin{tikzcd}
0 \arrow[r] \arrow[d] & 1 \arrow[rrr] \arrow[d] & & & 2 \arrow[rr] \arrow[d] & & 3 \arrow[rr] \arrow[d] & & 5 \arrow[r] \arrow[d] & 6 \arrow[d] \\
1 \arrow[d] & 1 \arrow[r] \arrow[d] & 2 \arrow[r] \arrow[d] & 3 \arrow[d] & 1 \arrow[r] \arrow[d] & 7 \arrow[d] & 1 \arrow[r] \arrow[d] & 2 \arrow[d] & 7 \arrow[d] & 9 \arrow[d] \\
0 & 4 & 0 & 2 & 4 & 2 & 4 & 0 & 2 & 2
\end{tikzcd}$$

*Figure 10: Multilevel linked data structure of the three-dimensional container for a load direction {3;2;1}.*

When using this data structure to provide the performing of the Eq. (1) after packing a new object into a container is necessary only put coordinates of new formed potential containers in the corresponding positions of the data structure without sorting of coordinates of all existing potential containers.

## 3.4 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. After deleting of an object it is necessary to reorganize all potential containers around this object.

The developed method of deleting a $D$-dimensional orthogonal object $i$ from a 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 containers $j$.
2. Put into the container $j'$ an object $i'$ with the dimensions $w_{i'}^d = w_i^d \quad \forall d \in \{1,\dots,D\}$ at a point with the coordinates equal to coordinates of the object $i$ placed into the container $j$: $x_{i'}^d = x_i^d \quad \forall d \in \{1,\dots,D\}$.
3. Create a list $\{L'\}$ containing potential containers the position and dimensions of which can be modified in container $j$ after deleting the object $i$. This list includes all potential containers $k'$ under the condition
$$x_{k'}^d \le x_i^d + w_i^d \quad \forall d \in \{1,\dots,D\}. \qquad (3)$$
4. Put into the container $j'$ at points $x_{k'}^d$ a set of objects with the dimensions
$$w_{k'}^d = p_{k'}^d \quad \forall d \in \{1,\dots,D\},\ k' \in \{L'\}. \qquad (4)$$
When placing objects into the container $j'$ allow them to 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 $x_{k''}^d$, $k'' \in \{L''\}$ a set of objects with the dimensions:
$$w_{k''}^d = p_{k''}^d \quad \forall d \in \{1,\dots,D\}. \qquad (5)$$
When placing objects into the container $j''$ allow them to 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 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. Replace in the container $j$ the list of its potential containers $\{L'\}$ to $\{L'''\}$.

In result is obtained a set of potential containers fully describes all freely orthogonal spaces of a container after deleting an object from it in a process of solving of any orthogonal packing problem of arbitrary dimension.

## 4. COMPUTATIONAL EXPERIMENTS

This paper contains full description of methods applied for management objects and the most of which have been presented 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 contained all details of the carried out computational experiments.

Consider results of solving standard three-dimensional orthogonal bin packing problems [9]. The test instances of orthogonal packing problems are grouped into 8 classes differing from each other by sets of objects to be packed. Containers are cubic in all considered test instances and have the dimensions 100 × 100 × 100. The aim is to pack all given objects into minimum count of the containers. For each test class were solved instances with a number of objects equal to 50, 100, 150 and 200. Given a test class and an instance size we generated 10 different problem instances based on different random sets of packed objects. Following the results published in scientific literature [9,25] we did not consider the test classes II–III because they have the same properties with the class I.

Results of solving of all test instances are summarized in Table 3. The best solutions in the table are bold. Were considered the following algorithms:

- LB – an exact lower bound of the problem [3];
- HA – the heuristic algorithm proposed by Lodi et al. for the node model [18];
- C-EPBFD – the heuristic algorithm for the placement with the Extreme Points rule [8];
- MPC – the model of potential containers using the described packing methods.

*Table 3: The Tests Results For The Standard Three-Dimensional Orthogonal Packing Problems.*

| Test class | LB | HA | C-EPBFD | MPC |
|---|---|---|---|---|
| Class I | 124.0 | 132.3 | 130.5 | **129.4** |
| Class IV | 292.2 | 294.3 | **294.0** | 300.3 |
| Class V | 66.4 | 73.6 | 72.6 | **70.6** |
| Class VI | 93.0 | 100.2 | 98.1 | **96.9** |
| Class VII | 55.4 | 63.7 | 62.8 | **62.7** |
| Class VIII | 77.8 | 87.1 | **85.4** | 88.2 |

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 solutions obtained by packing algorithms realized by other researchers.

The described multilevel linked data structure in practice more than twice increases access to the potential containers compared with the simple linear linked list [26] which needs in sorting [24]. This data structure is the most effective when it is using in 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 as it shown in paper [23] on a two-dimensional orthogonal packing problems as well in paper [27] where were considered standard three-dimensional orthogonal packing problems.

## 5. SOFTWARE FOR ORTHOGONAL PACKING PROBLEMS

All the described methods were used at developing a class library for solving the cutting and packing problems. The class library is designed using object-oriented programming techniques and is created with the programming language C++. The UML diagram of the class library for orthogonal packing and problems is given on figure 11.

Using the designed class library was developed an applied software intended for the optimization orthogonal packing and rectangular cutting problems [28]. The software allows to solve the following orthogonal cutting and packing problems:

- one-dimensional bin packing problem (1DBPP) [29];
- rectangular non-guillotine cutting problem [30];
- strip packing problem (SPP) [30];
- two-dimensional bin packing problem (2DBPP);
- three-dimensional bin packing problem (3DBPP).

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 including classes C1–C6 [31] and classes C7–C10 [32];
- 2DBPP instances [33];
- 3DBPP instances [9].

As an example we consider a three-dimensional orthogonal packing problem. On figure 12 is shown a window of the developed software containing geometrical parameters of containers and objects in the considered packing problem.

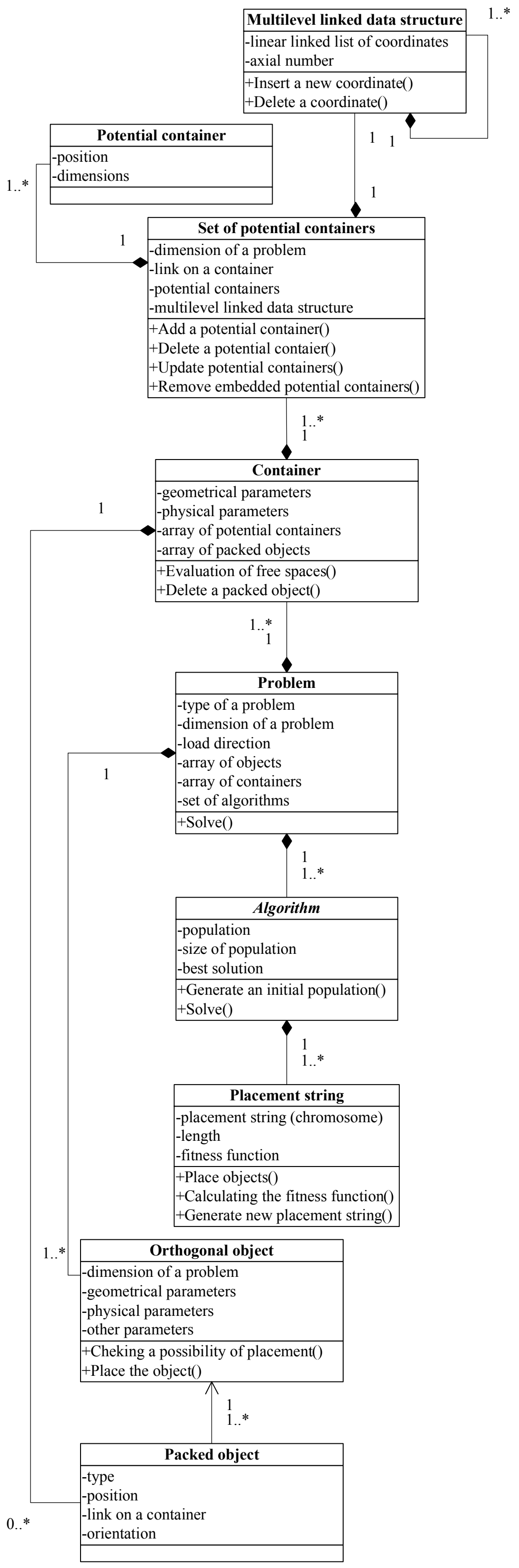
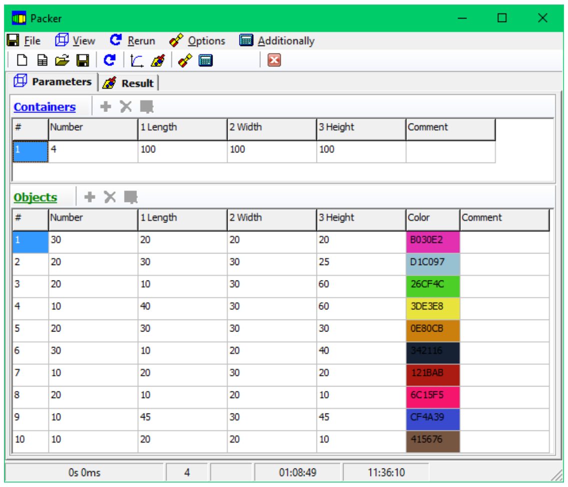


*Figure 11: UML Diagram Of The Developed Class Library For Orthogonal Packing Problems.*



| # | Number | 1 Length | 2 Width | 3 Height | Comment |
|---|---|---|---|---|---|
| 1 | 4 | 100 | 100 | 100 | |

| # | Number | 1 Length | 2 Width | 3 Height | Color | Comment |
|---|---|---|---|---|---|---|
| 1 | 30 | 20 | 20 | 20 | B030E2 | |
| 2 | 20 | 30 | 30 | 25 | D1C097 | |
| 3 | 20 | 10 | 30 | 60 | 26CF4C | |
| 4 | 10 | 40 | 30 | 60 | 3DE3E8 | |
| 5 | 20 | 30 | 30 | 30 | 0E80CB | |
| 6 | 30 | 10 | 20 | 40 | 342116 | |
| 7 | 10 | 20 | 30 | 20 | 121BAB | |
| 8 | 20 | 10 | 20 | 10 | 6C15F5 | |
| 9 | 10 | 45 | 30 | 45 | CF4A39 | |
| 10 | 10 | 20 | 20 | 10 | 415676 | |

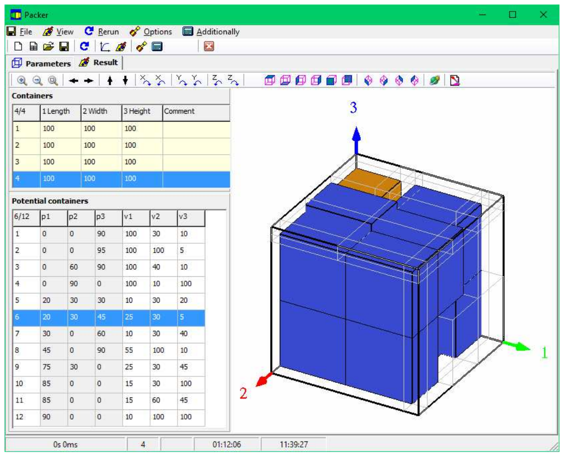
0s 0ms | 4 | | 01:08:49 | 11:36:10

*Figure 12: Parameters Of Considered Three-Dimensional Orthogonal Packing Problem.*

Solution of a packing problem is represented in the form of schemes of filling containers which are visualized on a tab "Result" of the software. For an example on figure 13 is given a packing scheme obtained for the forth container.

The developed software allows to show all existing potential containers that are contained in any selected container. On figure 14 are presented all the potential containers which are inside the container shown on figure 13.

Today the software we use during the computational experiments carried out over various algorithms and methods intended to solve various orthogonal packing and rectangular cutting problems.



| 4/4 | 1 Length | 2 Width | 3 Height | Comment |
|---|---|---|---|---|
| 1 | 100 | 100 | 100 | |
| 2 | 100 | 100 | 100 | |
| 3 | 100 | 100 | 100 | |
| 4 | 100 | 100 | 100 | |

| 6/12 | p1 | p2 | p3 | v1 | v2 | v3 |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 90 | 100 | 30 | 10 |
| 2 | 0 | 0 | 95 | 100 | 100 | 5 |
| 3 | 0 | 60 | 90 | 100 | 40 | 10 |
| 4 | 0 | 90 | 0 | 100 | 10 | 100 |
| 5 | 20 | 30 | 30 | 10 | 30 | 20 |
| 6 | 20 | 30 | 45 | 25 | 30 | 5 |
| 7 | 30 | 0 | 60 | 10 | 30 | 40 |
| 8 | 45 | 0 | 90 | 55 | 100 | 10 |
| 9 | 75 | 30 | 0 | 25 | 30 | 45 |
| 10 | 85 | 0 | 0 | 15 | 30 | 100 |
| 11 | 85 | 0 | 0 | 15 | 60 | 45 |
| 12 | 90 | 0 | 0 | 10 | 100 | 100 |

0s 0ms | 4 | | 01:12:06 | 11:39:27

*Figure 13: Visualization Of The Obtained Solution.*

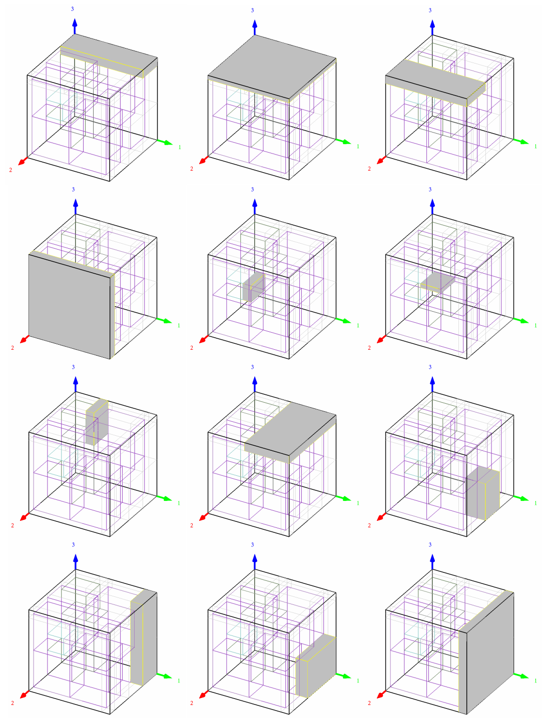


*Figure 14: Visualization Of All Potential Containers.*

## 6. CONCLUSION

In this first part of the paper we have presented methods of placing and deleting of orthogonal objects which are used in formation of orthogonal packing schemes. They have the following obvious advantages:

- possibility of solving orthogonal packing or rectangular cutting problems of any types;
- possibility of packing objects 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 packing due to usage the multilevel linked data structure.

All presented methods and algorithms are realized in a form of applied software for solving the orthogonal packing and rectangular cutting problems. In our future work we are planning to expand the possibilities of the developing software in the following ways:

- programming of known and new developed heuristic and metaheuristic algorithms for optimization of the orthogonal packing problems;
- including into the library of instances a new set of standard orthogonal packing and rectangular cutting problems;
- generation of detailed reports with the instructions of procedures of loading containers for consumers of the software in real industries which allows to use it not only for testing algorithms and methods in our scientific researches as now.

The presented method of deleting orthogonal objects of arbitrary dimension will be applied in algorithms of local improvement of resulting packing which are the subject of our subsequent research.

## 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. 2014/105, Project No. 1441).

## REFERENCES

[1] G. Wascher, H. Haubner, and H. Schumann, “An improved typology of cutting and packing problems”, *European Journal of Operational Research*, Vol. 183, No. 3, 2007, pp. 1109-1130.

[2] A. Bortfeldt, and G. Wascher, “Constraints in container loading – a state-of-the-art review”, *European Journal of Operational Research*, Vol. 229, No. 1, 2013, pp. 1-20.

[3] M. A. Boschetti, “New lower bounds for the finite three-dimensional bin packing problem”, *Discrete Applied Mathematics*, Vol. 140, 2004, pp. 241-258.

[4] Y. Q. Gao, H. B. Guan, Z. W. Qi, Y. Hou, and L. Liu, “A multi-objective ant colony system algorithm for virtual machine placement in cloud computing”, *Journal of Computer and System Sciences*, Vol. 79, No. 8, 2013, pp. 1230-1242.

[5] S. C. H. Leung, D. F. Zhang, C. L. Zhou, and T. Wu, “A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem”, *Computers & Operations Research*, Vol. 39, No. 1, 2012, pp. 64-73.

[6] A. Lodi, S. Martello, and M. Monaci, “Two-dimensional packing problems: A survey”, *European Journal of Operational Research*, Vol. 141, No. 2, 2002, pp. 241-252.

[7] V. A. Chekanin, and A. V. Chekanin, “Efficient Algorithms for Orthogonal Packing Problems”, *Computational Mathematics and Mathematical Physics*, Vol. 53, No. 10, 2013, pp. 1457-1465.

[8] T. G. Crainic, G. Perboli, and R. Tadei, “Extreme point-based heuristics for three-dimensional bin packing”, *Informs Journal on Computing*, Vol. 20, No. 3, 2008, pp. 368-384.

[9] S. Martello, D. Pisinger, and D. Vigo, “The three-dimensional bin packing problem”, *Operations Research*, Vol. 48, No. 2, 2000, pp. 256-267.

[10] S. P. Fekete, J. Schepers, and J. C. van der Veen, “An exact algorithm for higher-dimensional orthogonal packing”, *Operations Research*, Vol. 55, No. 3, 2007, pp. 569-587.

[11] L. Lins, S. Lins, and R. Morabito, “An n-tet graph approach for non-guillotine packings of n-dimensional boxes into an n-container”, *European Journal of Operational Research*, Vol. 141, No. 2, 2002, pp. 421-439.

[12] J. Westerlund, L. G. Papageorgiou, and T. Westerlund, “A MILP model for N-dimensional allocation”, *Computers & Chemical Engineering*, Vol. 31, No. 12, 2007, pp. 1702–1714.

[13] M. A. A. Martinez, F. Clautiaux, M. Dell'Amico, and M. Iori, “Exact algorithms for the bin packing problem with fragile objects”, *Discrete Optimization*, Vol. 10, No. 3, 2013, pp. 210-223.

[14] V. A. Chekanin, and A. V. Chekanin, “Development of the multimethod genetic algorithm for the strip packing problem”, *Applied Mechanics and Materials*, Vol. 598, 2014, pp. 377-381.

[15] D. Chen, J. Liu, Y. Fu, and M. Shang, “An efficient heuristic algorithm for arbitrary shaped rectilinear block packing problem”, *Computers & Operations Research*, Vol. 37, No. 6, 2010, pp. 1068-1074.

[16] J. F. Goncalves, and M. G. C. Resende, “A biased random key genetic algorithm for 2d and 3d bin packing problems”, *International Journal of Production Economics*, Vol. 145, No. 2, 2013, pp. 500-510.

[17] I. Kierkosz, and M. Luczak, “A hybrid evolutionary algorithm for the two-

dimensional packing problem", *Central European Journal of Operations Research*, Vol. 22, No. 4, 2014, pp. 729-753.

[18] A. Lodi, S. Martello, and D. Vigo, "Heuristic algorithms for the three-dimensional bin packing problem", *European Journal of Operational Research*, Vol. 141, No. 2, 2002, pp. 410-420.

[19] M. C. Riff, X. Bonnaire, and B. Neveu, "A revision of recent approaches for two-dimensional strip-packing problems", *Engineering Applications of Artificial Intelligence*, Vol. 22, No. 4-5, 2009, pp. 823-827.

[20] M. Garey, and D. Johnson, "*Computers Intractability: a Guide to the Theory of NP-completeness*", W.H.Freeman, San Francisco, 1979.

[21] V. A. Chekanin, and A. V. Chekanin, "An efficient model for the orthogonal packing problem", *Advances in Mechanical Engineering*, Vol. 22, 2015, pp. 33-38.

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

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

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

[25] O. Faroe, D. Pisinger, and M. Zachariasen, "Guided local search for the three-dimensional bin packing problem", *INFORMS Journal on Computing*, Vol. 15, No. 3, 2003, pp. 267-283.

[26] M. A. Weiss, "*Data Structures and Algorithm Analysis in C++*", Pearson Education, Boston, 2014.

[27] V. A. Chekanin, and A. V. Chekanin, "Improved data structure for the orthogonal packing problem", *Advanced Materials Research*, Vol. 945-949, 2014, pp. 3143-3146.

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

[29] D. Pisinger, and M. Sigurd, "Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem", *INFORMS Journal on Computing*, Vol. 19, No. 1, 2007, pp. 36-51.

[30] F. G. Ortmann, N. Ntene, and J. H van Vuuren, "New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems", *European Journal of Operational Research*, Vol. 203, No. 2, 2010, pp. 306-315.

[31] J. O. Berkey, and P. Y. Wang, "Two-dimensional finite bin-packing algorithms", *Journal of the Operational Research Society*, Vol. 38, No. 5, 1987, pp. 423-429.

[32] S. Martello, and D. Vigo, "Exact solution of the two-dimensional finite bin packing problem", *Management Science*, Vol. 44, No. 3, 1998, pp. 388-399.

[33] S. P. Fekete, and J. Schepers, "New classes of lower bounds for bin packing problems", *Lecture Notes in Computer Science*, Vol. 1412, 1998, pp. 257-270.