

# Algorithms for Working with Orthogonal Polyhedrons in Solving Cutting and Packing Problems

Vladislav Chekanin<sup>1</sup> and Alexander Chekanin<sup>1</sup>

<sup>1</sup> *Moscow State University of Technology «STANKIN», 3a Vadkovsky lane, Moscow, 127055, Russia*

## Abstract

In this paper problems of cutting and packing objects of complex geometric shapes are considered. To solve these NP-hard problems, it is proposed to use an approach based on geometric transformation of polygonal objects to composite objects (orthogonal polyhedrons) made up of rectangles or parallelepipeds of a given dimension. To describe the free space inside a voxelized container, a model of potential containers is used as the basic model that provides the ability of packing orthogonal polyhedrons. A number of specialized algorithms are developed to work with orthogonal polyhedrons including: algorithms for placing and removing composite objects, an algorithm for forming a packing with a given distance between objects to be placed. Two algorithms for the placement of orthogonal polyhedrons are developed and their efficiency is investigated. An algorithm for obtaining a container of complex shape presented as an orthogonal polyhedron based on a polygonal model is given. The article contains examples of placement schemes obtained by the developed algorithms for solving problems of packing two-dimensional and three-dimensional non-rectangular composite objects.

## Keywords

Orthogonal polyhedron, voxelization, packing problem, irregular cutting

## 1. Introduction

The problem of finding the best way to place objects of complex geometric shapes is characterized by a wide range of its practical applications. Among the most common areas of such practical applications are [1–7]:

- cutting industrial materials (metal sheets, cardboard, plywood and etc.);
- rational usage of free spaces (for example, spaces of aircraft, tankers, spaceships and etc.);
- geometric surface coverage with specified shapes;
- modeling the microstructure of composite materials;
- generation of active electronically scanned arrays.

All packing problems are NP-hard optimization problems for which there are no algorithms of polynomial complexity to solve them [1, 8], which makes it urgent to develop algorithms that provide suboptimal solutions in an acceptable time. Modern methods for solving problems of packing objects of complex geometric shape require usage of the hodograph vector function of dense placement, which requires the subsequent application of nonlinear programming methods characterized by high computational complexity [9–15]. The paper proposes an approach that allows you to reduce the complexity of the problem. It consists in voxelization of objects to be packed [16–18], subsequent transforming the obtained voxelized objects to orthogonal polyhedrons after their decomposition [19] and finally application of algorithms especially developed for placement of orthogonal polyhedrons.

---

*GraphiCon 2021: 31st International Conference on Computer Graphics and Vision, September 27-30, 2021, Nizhny Novgorod, Russia*

EMAIL: vladchekanin@rambler.ru (V. Chekanin); av.chekanin@stankin.ru (A. Chekanin)

ORCID: 0000-0001-9448-0583 (V. Chekanin); 0000-0003-0788-469X (A. Chekanin)



© 2021 Copyright for this paper by its authors.

Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

We will consider the problem of placement orthogonal polyhedrons in the general  $D$ -dimensional case. A container is specified as a  $D$ -dimensional parallelepiped with the dimensions  $\{W^1; W^2; \dots; W^D\}$  (the superscript in formulas means the number of the coordinate axis). Objects to be packed are specified by a set of  $n$  compound objects  $O_i$  as orthogonal polyhedrons ( $i \in [1; n]$ ). Every compound object consists from a set of  $m_i$  simple objects (rectangles or parallelepipeds) at the same time, each simple object  $o_{ik}, k \in [1; m_i]$  has the overall dimensions  $\{w_{ik}^1; w_{ik}^2; \dots; w_{ik}^D\}$  as well is located at point  $\{z_{ik}^1; z_{ik}^2; \dots; z_{ik}^D\}$  of the considered orthogonal polyhedron. When all compound objects consist of only one simple object, then the problem is reduced to the classical problem of orthogonal packing or rectangular cutting [2, 20].

## 2. Developed algorithms for working with orthogonal polyhedrons

### 2.1. Description of container free spaces

To describe a packing entire is used the developed model of potential containers [20, 21]. A potential container is an imaginary orthogonal region ( $D$ -dimensional parallelepiped) for which there is free space inside a packing. The model of potential containers used forms the set of all possible potential containers of minimum capacity. The designation  $\{p_h^1; p_h^2; \dots; p_h^D\}$  is used to indicate the overall dimensions of a potential container  $P_h$  and its position inside the considered packing is determinate by a set of values  $\{x_h^1; x_h^2; \dots; x_h^D\}$ . When a new orthogonal object has to be put into a container it is necessary to check the condition that it does not intersect with objects already placed in the placement scheme. When using the model of potential containers, it is possible to guarantee the correctness of the resulting packing if each placed object can be completely placed inside at least one potential container. Obviously, this check is performed quickly, and the speed of orthogonal packing formation is high.

To update a set of potential containers after placing an orthogonal polyhedron  $O_i$  at the point  $\{X_i^1, X_i^2, \dots, X_i^D\}$  of a  $D$ -dimensional container, the entire of which is fully described by a set of potential containers  $\Omega_0$ , the following algorithm is performed.

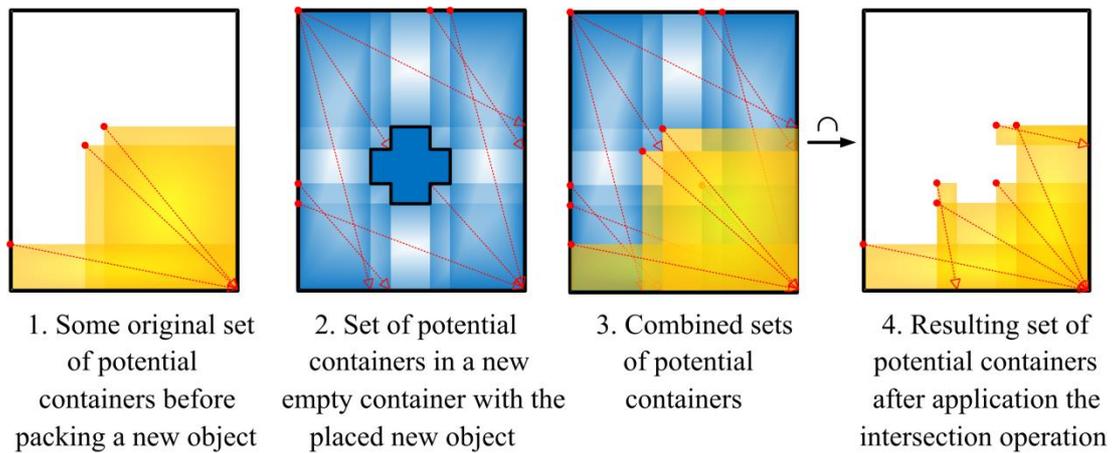
Step 1. Create a set  $\Omega'_0 \subset \Omega_0$  of potential containers  $h: \exists d \in \{1, \dots, D\}: x_h^d \leq X_i^d + S_i^d$ , where  $\{S_i^1, S_i^2, \dots, S_i^D\}$  denotes the overall dimensions of a  $D$ -dimensional parallelepiped bounding the orthogonal polyhedron  $O_i: S_i^d = \max(z_{ik}^d + w_{ik}^d), \forall d \in [1; D], k \in [1; m_i]$ .

Step 2. Place an orthogonal polyhedron  $O_i$  in the specified position  $\{X_i^1, X_i^2, \dots, X_i^D\}$  of a new identical empty container, as a result of which a set of potential containers  $\Omega$  will be formed in it. Placement of the orthogonal polyhedron is performed by sequentially placing of all its orthogonal objects  $o_{ik}, k \in [1; m_i]$ .

Step 3. Apply the intersection operation [19] to sets of potential containers  $\Omega'_0$  and  $\Omega$  to get a set of potential containers  $\Omega''_0 = \Omega'_0 \cap \Omega$  that describes all free spaces of the original container in the area of the compound object  $O_i$ . The operation of intersection of sets of potential containers is implemented in the same way as the operation of intersection of two orthogonal polyhedrons, each of which contains the same parameters (position, overall dimensions) of orthogonal objects as the corresponding parameters of potential containers.

Step 4. Replace in the set  $\Omega_0$  all potential containers that are also in the set  $\Omega'_0$  with potential containers from the set  $\Omega''_0$ .

This algorithm for updating potential containers is presented in Figure 1.

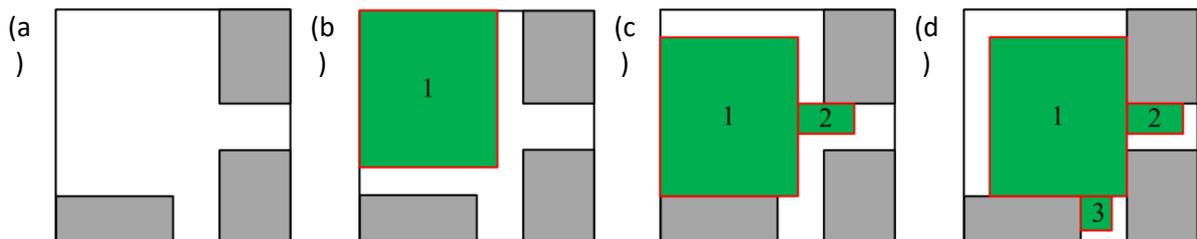


**Figure 1:** Updating of potential containers when placing an orthogonal polyhedron

## 2.2. Packing an orthogonal polyhedron

Two algorithms have been developed for packing an orthogonal polyhedron into a container, which are implemented with a generalization in dimension.

Algorithm 1 consists in the sequential formation of an orthogonal polyhedron and performing a series of offsets during its placement in the container [22]. Figure 2 presents an example of the sequential formation of a composite object when it is placed inside a container with geometrical restrictions (shown in gray in the figure).



**Figure 2:** Orthogonal polyhedron in the packing process (Algorithm 1): (a) original container with restrictions; (b) placement of the first object; (c) subsequent placement of the second object; (d) final result of placement the entire orthogonal polyhedron

The second algorithm (denoted by Algorithm 2) for placing an orthogonal polyhedron is based on determining the common area of acceptable placement for intermediate orthogonal polyhedrons composed of separate objects of the original composite object [19].

Table 1 contains the results of computational experiment which was conducted for comparing these two algorithms.

In the course of the computational experiment, the problem of placing two identical orthogonal polyhedrons into a container was solved, the height ( $W^2$ ) of which coincides with the height of the rectangle bounded around the placed orthogonal polyhedron. When solving the problem, a personal computer was used (processor – Intel Core i5-8400, 2.8 GHz; RAM – 8 GB).

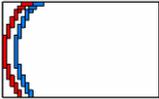
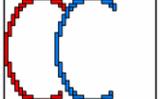
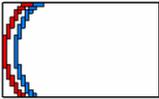
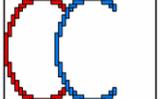
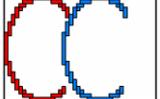
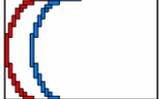
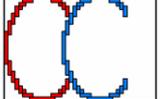
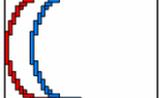
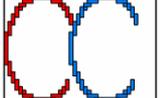
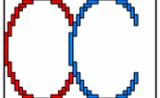
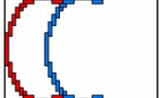
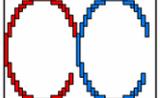
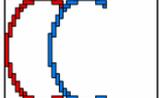
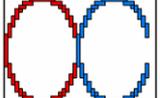
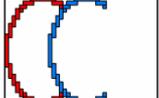
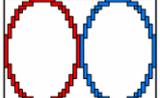
Table 1 uses the following notation:

- $m$  – the number of simple objects in the orthogonal polyhedron;
- $T_1$  – the time spent on the placement of orthogonal polyhedrons by Algorithm 1;
- $T_2$  – the time spent on the placement of orthogonal polyhedrons by Algorithm 2.

The computational experiment was performed for various intermediate orthogonal polyhedrons consisting of orthogonal objects. When placing an orthogonal polyhedron consisting of objects whose

number is less than 11, both algorithms ensure the placement of composite objects in the same time (0.03-0.04 s), so Table 1 shows the results obtained when  $m \in [11;28]$ .

**Table 1**  
Comparison of algorithms for placing an orthogonal polyhedron

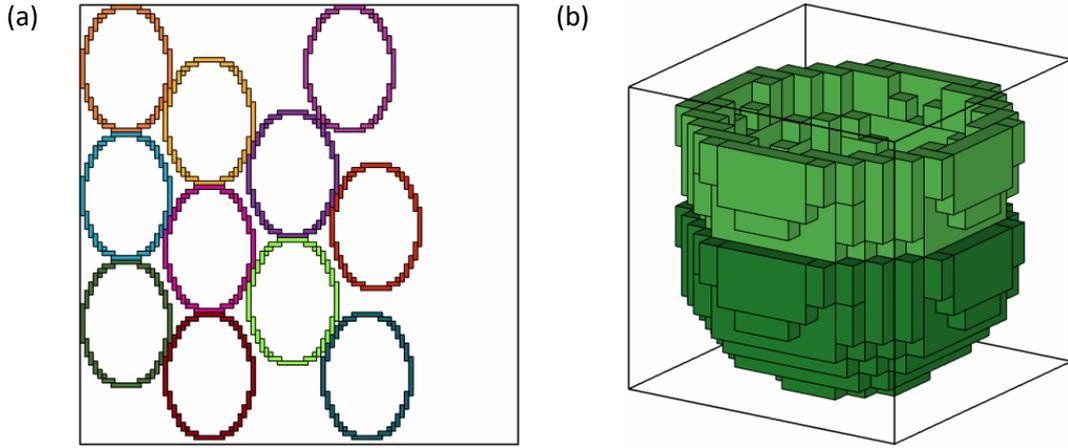
$m$	$T_1, s$	$T_2, s$	$W^2$	Visualization	$m$	$T_1, s$	$T_2, s$	$W^2$	Visualization
11	0.08	0.04	26		20	1.30	0.04	29	
12	0.07	0.04	26		21	0.22	0.04	29	
13	0.16	0.04	27		22	0.55	0.04	29	
14	2.89	0.04	28		23	0.28	0.05	29	
15	7.58	0.04	29		24	2.71	0.05	29	
16	118.13	0.04	29		25	1.60	0.05	29	
17	0.64	0.04	29		26	14.18	0.07	29	
18	1.20	0.04	29		27	8.38	0.07	29	
19	0.22	0.04	29		28	9.08	0.07	29	

The results obtained showed that for all problems, Algorithm 2 provides placement in a time that is not less than the time spent by Algorithm 1, while for all values  $m \in [11;28]$ , Algorithm 2 provides placement of objects at a speed that is on average two orders of magnitude higher.

Since Algorithm 2 places orthogonal polyhedrons without shifting, their optimal placement is obtained without using the algorithm for determining the optimal position described in article [22].

In Figure 3, a is presented an example of packing 11 ellipses in a rectangular container. It was obtained by Algorithm 2 in 2.8 s while Algorithm 1 did not provide a solution in a time limited to 300 s.

Algorithm 2 provides a significant increase in the speed of placement of both two-dimensional and three-dimensional orthogonal polyhedrons. As an example, consider the problem of placing two identical orthogonal bowl-shaped polyhedrons consisting of 110 orthogonal objects in a three-dimensional container (Figure 3, b). Algorithm 1 provides placement of these orthogonal polyhedrons in 13.32 seconds, while Algorithm 2 spends only 0.40 seconds.



**Figure 3:** Test packing problems: (a) placing of 11 ellipses in a rectangular container; (b) placing of 2 bowl-shaped objects in a parallelepiped

### 2.3. Removing a compound object from a packing

To increase the density of a placement of orthogonal polyhedrons, their local redistribution within the formed packing can be performed. In this case, after removing some orthogonal polyhedron of a given dimension, it is necessary to perform the procedure for updating the set of potential containers in the area of its initial placement.

The algorithm for deleting one object which described in article [23] can be used to implement the algorithm for deleting a compound object represented by a set of simple objects.

We assume that the orthogonal polyhedron  $O_i$  being removed is located in the container  $j$  at point  $\{X_{ij}^1, X_{ij}^2, \dots, X_{ij}^D\}$ .

The developed algorithm for updating the set of potential containers after removing an orthogonal polyhedron of arbitrary dimension  $D$  contains 10 steps.

Step 1. Create an empty copy of container  $j$  and denote it as a container  $j'$ .

Step 2. Set the number  $k := 1$  for the currently deleted simple object  $o_{ik}$ .

Step 3. Place the simple object  $o_{ik}$  at the point  $\{X_{ij}^1 + z_{ik}^1, X_{ij}^2 + z_{ik}^2, \dots, X_{ij}^D + z_{ik}^D\}$  of the container  $j'$ .

Step 4. Increase the number of the current simple object ( $k := k + 1$ ). If  $k \leq m$  then go to step 3, otherwise go to step 5.

Step 5. Select in the container  $j$  potential containers, the overall dimensions of which can be changed after deleting the composite object. This set  $\Omega_1$  includes potential containers for which the inequality  $x_{k'}^d \leq X_{ij}^d + S_i^d$  is true (the vector  $\{S_i^1, S_i^2, \dots, S_i^D\}$  stores overall dimensions of the AABB parallelepiped bounding the orthogonal polyhedron  $O_i$ ).

Step 6. Fill the container  $j'$  with objects that completely fill the space described by a set  $\Omega_1$  of potential containers. Denote the set of remained potential containers in the container  $j'$  by  $\Omega_2$ .

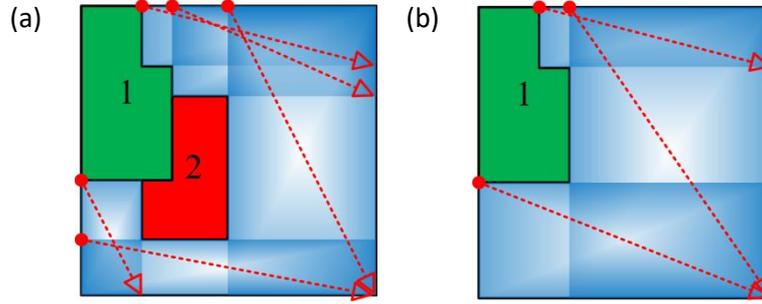
Step 7. Create an empty copy of container  $j$  and denote it as a container  $j''$ .

Step 8. Fill the container  $j''$  with objects that completely fill the space described by a set  $\Omega_1$  of potential containers. Denote the set of remained potential containers in the container  $j''$  by  $\Omega_3$ .

Step 9. Remove the compound object  $O_i$  from the container  $j$ .

Step 10. Remove all potential containers from set  $\Omega_1$  from the container  $j$  and add into it all potential containers from set  $\Omega_3$ .

Figure 4, a presents an example of a container containing two placed orthogonal polyhedrons and its free spaces. Figure 4, b shows the contents of the container after removing the red orthogonal polyhedron (No. 2).



**Figure 4:** Removing a two-dimensional orthogonal polyhedron: (a) objects and potential containers before removing OM No. 2; (b) entire of the container after removing the orthogonal polyhedron

## 2.4. Algorithm for forming a complex-shaped container

To transform a container represented in the form of a  $D$ -dimensional parallelepiped into an orthogonal polyhedron, it is proposed to place a set of virtual orthogonal objects into it (they are denoted as constraint objects) before forming the placement scheme. The set of constraint objects will be represented finally by a single orthogonal polyhedron  $\hat{O}$  of geometric constraints of a container.

We will consider a container in the form of a  $D$ -dimensional parallelepiped with overall dimensions  $\{W^1, W^2, \dots, W^D\}$ , as well as an original set  $A$  of orthogonal constraint objects. The constraint object with a number  $k$  in the form of a  $D$ -dimensional parallelepiped will be denoted by  $\hat{o}_k$ , its position in the coordinate system of the container will determine the vector  $\{z_k^1, z_k^2, \dots, z_k^D\}$ , and its overall dimensions will determine the vector  $\{w_k^1, w_k^2, \dots, w_k^D\}$ . For each constraint object  $\hat{o}_k$ , the type of operation applied to it (addition or subtraction of an orthogonal object [19]), which is set when it is created, is known in advance.

All constraint objects from the set  $A$  are transformed according to the overall dimensions of the container, so that no constraint object extends beyond the container boundaries. To do this, the following algorithm is performed (steps 1-5).

Step 1. Create an empty set  $\hat{A}$  of orthogonal constraint objects ( $\hat{A} = \emptyset$ ).

Step 2. Set the number of the current constraint object  $k := 1$ .

Step 3. For the constraint object  $\hat{o}_k$ , perform a check for its placement entirely outside the container boundaries:  $z_k^d \geq W^d$  or  $z_k^d + w_k^d \leq 0$  for everyone  $d \in \{1, \dots, D\}$ . If the constraint object is located outside the container, then go to step 5, otherwise go to step 4.

Step 4. Based on the constraint object  $\hat{o}_k$ , create a new constraint object  $\hat{o}$  whose position vector  $\{\hat{z}^1, \hat{z}^2, \dots, \hat{z}^D\}$  and overall dimensions  $\{\hat{w}^1, \hat{w}^2, \dots, \hat{w}^D\}$  are defined as follows:

- $\hat{z}^d = 0$  and  $\hat{w}^d = w_k^d + z_k^d$  if  $z_k^d < 0 \quad \forall d \in \{1, \dots, D\}$ ;
- $\hat{z}^d = z_k^d$  and  $\hat{w}^d = W^d - z_k^d$  if  $z_k^d > W^d \quad \forall d \in \{1, \dots, D\}$ .

The same operation (addition or subtraction) is set for the constraint object  $\hat{o}$  as for the original constraint object  $\hat{o}_k$ .

Place the constraint object  $\hat{o}$  in the set  $\hat{A}$ .

Step 5. Go to the next constraint object  $k := k + 1$ . If  $k \leq |A|$  then go to step 3, otherwise stop the algorithm.

When creating an orthogonal polyhedron of geometric constraints, the following sets are formed based on the resulting set  $\hat{A}$ :

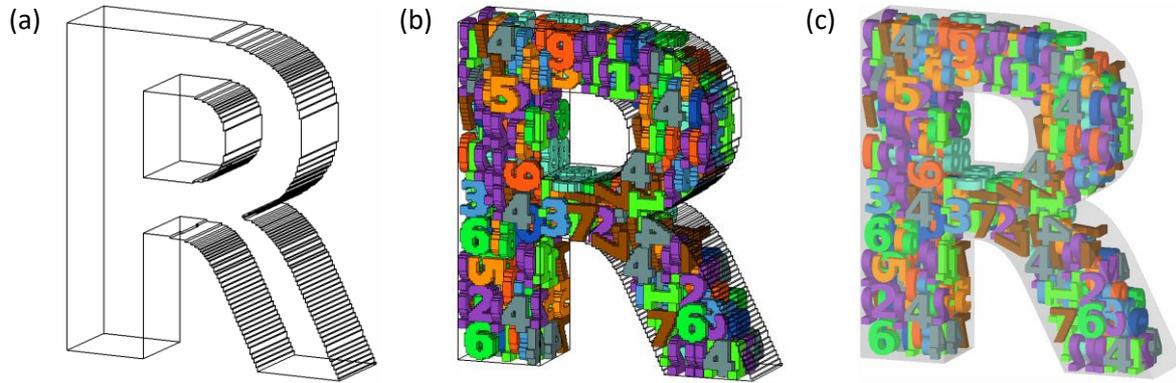
- a set of constraint objects  $\hat{A}^+$  to which the set-theoretic addition operation is applied;
- a set of constraint objects  $\hat{A}^-$  to which the set-theoretic subtraction operation is applied.

To form an orthogonal polyhedron  $\hat{O}$  of geometric constraints with the sets  $\hat{A}^+$  and  $\hat{A}^-$ , the algorithms described in the article [19] are used. This orthogonal polyhedron of geometric constraints of the container may include constraint objects that will not have common points with each other.

The process of creating a container based on a polygonal model includes the following steps.

1. Voxelization of the model contour and filling the voids of the resulting orthogonal polyhedron.
2. Decomposition of the resulting orthogonal polyhedron of geometric constraints into large objects to increase the speed of packing formation.
3. Apply a subtraction operation with an orthogonal polyhedron of geometric constraints to a AABB parallelepiped bounding the given polygonal model.

Figure 5 presents an example of a container based on a three-dimensional model of the Latin letter R, as well as the result of packing a set of orthogonal polyhedrons into it.



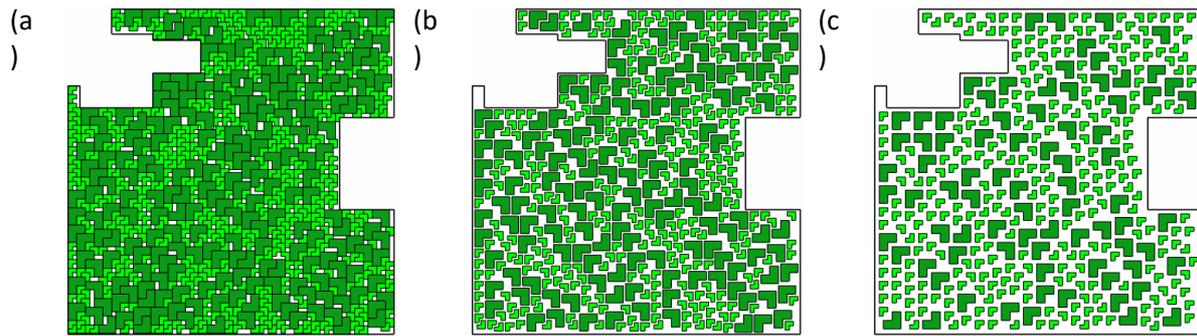
**Figure 5:** A complex-shaped container: (a) container obtained after application the proposed algorithm; (b) result of packing a set of orthogonal objects; (c) visualization of the resulting packing in a polygonal form

## 2.5. Forming an equidistant packing

To obtain a packing with a given fixed gap  $\rho$  between orthogonal polyhedrons (an equidistant packing), the position and overall dimensions of all orthogonal objects that are part of these orthogonal polyhedrons are corrected during their placement.

When placing a  $D$ -dimensional orthogonal polyhedron  $O$  at a point  $\{X^1, X^2, \dots, X^D\}$  of a container, each orthogonal object  $o_k \in O$  will be placed at a point with the gap  $\rho$  in the direction of the origin:  $\{X^1 + z_k^1 - \rho, X^2 + z_k^2 - \rho, \dots, X^D + z_k^D - \rho\}$ , and also will have an increased overall size by an amount  $2\rho$ :  $\{w_k^1 + 2\rho, w_k^2 + 2\rho, \dots, w_k^D + 2\rho\}$ . After obtaining a set of potential containers describing free entire of the container (which formed as a result of placing this orthogonal polyhedron), the original values of the vectors describing the position and overall dimensions of its orthogonal objects are restored.

Figure 6 shows examples of the formation of an equidistant packing of a large number of objects when setting different gaps  $\rho$  (the overall dimensions of a rectangle bounding the container are  $300 \times 300$ ).



**Figure 6:** Forming equidistant placement schemes: (a) packing of 826 objects with zero gap; (b) equidistant packing of 467 objects with gap  $\rho = 2$ ; (c) equidistant packing of 370 objects with gap  $\rho = 4$

### 3. Conclusion

The article presents a number of algorithms designed for packing objects of complex shape, represented as orthogonal polyhedrons. These algorithms are the basis for a new method for solving problems of cutting and packing irregular objects, which consists in reducing the solved problems to the problems of placing orthogonal polyhedrons of a given dimension, which can be solved in significantly less time.

An algorithm for placing orthogonal polyhedrons of arbitrary dimension was proposed, based on the sequential formation of an orthogonal polyhedron and performing a series of displacements during its placement in the container (Algorithm 1). This algorithm provides fast production of dense placement schemes of orthogonal polyhedrons made up of a small number of orthogonal objects (usually consisting of no more than 10 objects). To increase the speed of placement of orthogonal polyhedrons with a larger number of orthogonal objects, an improved algorithm (Algorithm 2) is developed, based on the creation of orthogonal polyhedrons that determine the areas of acceptable placement for each object from this compound object. Algorithm 2 significantly increases the speed of packing formation (on average by two orders of magnitude) without losing the quality of placement, which makes it advisable to use it in solving any types of problems of packing orthogonal polyhedrons.

The developed algorithm for removing an orthogonal polyhedron and the use of the model of potential containers that describes all existing free areas inside the containers makes it possible to apply methods for local reallocation of compound objects placed in the container.

A method for forming an equidistant packing where all objects are located at a given minimum distance from each other has been developed. This method is used, in particular, when forming the arrangement of complex-shaped parts on a 3D printer platform.

An important distinguishing feature of the presented algorithms is their implementation for the case of an arbitrary dimension of the problem. All the described algorithms are programmatically implemented and tested in the author's software «Packer» [20, 24, 25].

### 4. References

- [1] G. Wäscher, H. Haußner, H. Schumann, An improved typology of cutting and packing problems, *European Journal of Operational Research* 183(3) (2007) 1109–1130. doi:10.1016/j.ejor.2005.12.047.
- [2] A. Bortfeldt, G. Wäscher, Constraints in container loading - A state-of-the-art review, *European Journal of Operational Research* 229(1) (2013) 1–20. doi:10.1016/j.ejor.2012.12.006.
- [3] T. G. Crainic, G. Perboli, R. Tadei, Recent advances in multi-dimensional packing problems, in: C. Volosencu (Eds.), *New Technologies—Trends, Innovations and Research*, InTech, 2012, pp. 91–110. doi:10.5772/33302.

- [4] R. J. Mailloux, S. G. Santarelli, T. M. Roberts, D. Luu, Irregular polyomino-shaped subarrays for space-based active arrays, *International Journal of Antennas and Propagation* (2009). doi:10.1155/2009/956524.
- [5] S. Plankovskyy, Y. Tsegelnyk, O. Shypul, A. Pankratov, T. Romanova, Cutting irregular objects from the rectangular metal sheet, in: M. Nechyporuk, V. Pavlikov, D. Kritskiy (Eds), *Integrated Computer Technologies in Mechanical Engineering*, volume 1113, Springer, Cham, 2020, pp. 150-157. doi:10.1007/978-3-030-37618-5\_14.
- [6] C. Zhao, L. Jiang, K. L. Teo, A hybrid chaos firefly algorithm for three-dimensional irregular packing problem, *Journal of Industrial & Management Optimization* 16(1) (2020) 409–429. doi:10.3934/jimo.2018160.
- [7] M. O. Arbuzov, A. Y. Nekrasov, A. N. Sobolev, A progressive method of mounting machine parts on the shaft, *IOP Conference Series: Materials Science and Engineering* 709(2) (2020) 022066. doi:10.1088/1757-899X/709/2/022066.
- [8] D. S. Johnson, A brief history of NP-completeness, 1954–2012, *Documenta Mathematica Extra Volume ISMP* (2012) 359–376. URL: <https://elibm.org/article/10011465>.
- [9] S. C. Leung, Y. Lin, D. Zhang, Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem, *Computers & Operations Research* 39(3) (2012) 678–686. doi:10.1016/j.cor.2011.05.025.
- [10] Y. Stoyan, T. Romanova, A. Pankratov, A. Chugay, Optimized object packings using quasi-phi-functions, in: G. Fasano, J. D. Pintér (Eds.), *Springer Optimization and Its Applications*, volume 105, Springer, Cham, 2015, pp. 265–293. doi:10.1007/978-3-319-18899-7\_13.
- [11] N. Chernov, Y. Stoyan, T. Romanova, Mathematical model and efficient algorithms for object packing problem, *Computational Geometry* 43(5) (2010) 535–553. doi:10.1016/j.comgeo.2009.12.003.
- [12] J. Bennell, G. Scheithauer, Y. Stoyan, T. Romanova, Tools of mathematical modeling of arbitrary object packing problems, *Annals of Operations Research* 179(1) (2010) 343–368. doi:10.1007/s10479-008-0456-5.
- [13] E. G. Birgin, L. H. Bustamante, H. F. Callisaya, J. M. Martínez, Packing circles within ellipses, *International Transactions in Operational Research* 20(3) (2013) 365–389. doi:10.1111/itor.12006.
- [14] E. G. Birgin, R. D. Lobato, J. M. Martínez, Packing ellipsoids by nonlinear optimization, *Journal of Global Optimization* 65(4) (2016) 709–743. doi:10.1007/s10898-015-0395-z.
- [15] M. Verkhoturov, A. Petunin, G. Verkhoturova, K. Danilov, D. Kurennov, The 3D object packing problem into a parallelepiped container based on discrete-logical representation, *IFAC-PapersOnLine* 49(12) (2016) 1–5. doi:10.1016/j.ifacol.2016.07.540.
- [16] A. V. Tolok, N. B. Tolok, Mathematical programming problems solving by functional voxel method, *Automation and Remote Control* 79(9) (2018) 1703–1712. doi:10.1134/S0005117918090138.
- [17] A. C. J. De Korte, H. J. H. Brouwers, Random packing of digitized particles, *Powder technology*, 233 (2013) 319–324. doi:10.1016/j.powtec.2012.09.015.
- [18] T. Byholm, M. Toivakka, J. Westerholm, Effective packing of 3-dimensional voxel-based arbitrarily shaped particles, *Powder Technology* 196(2) (2009) 139–146. doi:10.1016/j.powtec.2009.07.013.
- [19] V. Chekanin, Solving the problem of packing objects of complex geometric shape into a container of arbitrary dimension, *CEUR Workshop Proceedings*, 2744 (2020). doi:10.51130/graphicon-2020-2-3-50.
- [20] V. A. Chekanin, A. V. Chekanin, Algorithms for management objects in orthogonal packing problems, *ARPN Journal of Engineering and Applied Sciences* 11(13) (2016) 8436–8446. URL: [http://www.arpnjournals.org/jeas/research\\_papers/rp\\_2016/jeas\\_0716\\_4620.pdf](http://www.arpnjournals.org/jeas/research_papers/rp_2016/jeas_0716_4620.pdf).
- [21] V. A. Chekanin, A. V. Chekanin, New effective data structure for multidimensional optimization orthogonal packing problems, in: A. Evgrafov (Eds.), *Advances in Mechanical Engineering, Lecture Notes in Mechanical Engineering*. Springer, Cham, 2016, pp. 87–92. doi:10.1007/978-3-319-29579-4\_9.
- [22] V. A. Chekanin, A. V. Chekanin, Algorithm for the placement of orthogonal polyhedrons for the cutting and packing problems, in: A. Evgrafov (Eds.), *Advances in Mechanical Engineering*,

- Lecture Notes in Mechanical Engineering, Springer, Cham, 2020, pp. 41–48. doi:10.1007/978-3-030-39500-1\_5.
- [23] V. A. Chekanin, A. V. Chekanin, Deleting objects algorithm for the optimization of orthogonal packing problems, in: A. Evgrafov (Eds.), *Advances in Mechanical Engineering, Lecture Notes in Mechanical Engineering*, Springer, Cham, 2017, pp. 27–35. doi:10.1007/978-3-319-53363-6\_4.
- [24] V. A. Chekanin, A. V. Chekanin, Design of library of metaheuristic algorithms for solving the problems of discrete optimization, in: A. Evgrafov (Eds.), *Advances in Mechanical Engineering, Lecture Notes in Mechanical Engineering*, Springer, Cham, 2018, pp. 25–32. doi:10.1007/978-3-319-72929-9\_4.
- [25] V. A. Chekanin, A. V. Chekanin, Development of algorithms for the correct visualization of two-dimensional and three-dimensional orthogonal polyhedrons, in: A. Radionov, A. Karandaev (Eds.), *Advances in Automation, RusAutoCon 2019, Lecture Notes in Electrical Engineering*, volume 641, Springer, Cham, 2020, pp. 891–900. doi: 10.1007/978-3-030-39225-3\_96.