Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

A hybrid approach of simulation and metaheuristic for the polyhedra packing problem

Germán Fernando Pantoja-Benavides[a] and David Álvarez-Martínez[a*]

[a]*Universidad de los Andes, Colombia*

C H R O N I C L E

Article history:
Received March 8 2021
Received in Revised Format July 10 2021
Accepted July 19 2021
Available online July, 19 2021

Keywords:
Polyhedra packing problem
Probabilistic tabu search
Simulation
Unity®

A B S T R A C T

This document presents a simulation-based method for the polyhedra packing problem (PPP). This problem refers to packing a set of irregular polyhedra (convex and concave) into a cuboid with the objective of minimizing the cuboid's volume, considering non-overlapping and containment constraints. The PPP has applications in additive manufacturing and packing situations where volume is at a premium. The proposed approach uses Unity® as the simulation environment and considers nine intensification and two diversification movements. The intensification movements induce the items within the cuboid to form packing patterns allowing the cuboid to decrease its size with the help of gravity-like accelerations. On the other hand, the diversification movements are classic transition operators such as removal and filling of pieces and enlargement of the container, which allow searching on different solution neighborhoods. All simulated movements were hybridized with a probabilistic tabu search. The proposed methodology (with and without the hybridization) was compared by benchmarking with all previous works solving the PPP with irregular items. Results show that satisfactory solutions were reached in a short time; even a few published results were improved.

1. Introduction

Cutting and packing problems (C&PP) have been an object of great interest within the computational geometry and operational research communities (Ma et al., 2018). These problems aim to maximize space utilization or minimize space waste when embedding a set of items within larger containers (Dyckhoff, 1990), and have numerous applications in 1D, 2D, and 3D, which may explain research communities' interest in solving them. Furthermore, applications of C&PP in 3D with irregular items happen in scenarios where the volume is limited and some components have to be tightly packed; for example, in automobiles' engineering design, for example, avionics (Romanova et al., 2018) and furniture transportation (Egeblad et al., 2010). An outstanding application of C&PP in 3D with irregular pieces is found in *additive manufacturing* (AM), in which the printing time is reduced when several pieces are placed within the AM machine's production volume (Egeblad et al., 2009). AM is also known as 3D printing and layer technology (Araújo et al., 2020); it is a set of technologies developed in the late 1980s (Wong and Hernandez, 2012), which allows the production of pieces with specific desired shapes. This is done by adding material layer-by-layer (Gebhardt and Hotter, 2016) without requiring other physical devices such as molds (Hague et al., 2004). This is a very time-consuming process; producing a piece can take hours or even days. However, production time can be decreased by printing several objects in one run of the machine. Thus, in order to minimize the production time, the placement of the pieces within the machine's production volume becomes a 3D packing problem. There is a lack of research in C&PP concerning 3D and irregularity in the items to be packed. Wäscher et al. (2007) found that there are fewer publications of C&PP in 3D than in 1D and 2D. Furthermore, Dyckhoff (1990) and Wäscher et al. (2007) found fewer publications of C&PP involving irregular shapes than regular ones. These differences in the number of publications may happen due to the complications that 3D and irregularity imply. 3D generally increases the solution space when compared to 2D and 1D, which increases the solution time. On the other hand, irregular shapes are more complicated to model than regular ones, which complicates the assessment of the interaction between items and makes it more time-consuming (Bennell and

* Corresponding author Tel: +57 3117346940
E-mail: d.alvarezm@uniandes.edu.co (D. Álvarez-Martínez)

2022 Growing Science Ltd.
doi: 10.5267/j.ijiec.2021.7.003

Oliveira, 2008). This situation reveals the research opportunity of C&PP in 3D with irregular items, in which this study is framed. An important constraint within C&PP is the rotation of the pieces to be packed. The C&PP literature usually considers fixed orientations of the pieces, which are appropriated when considering regular shapes, e.g., boxes. However, these fixed orientations may not be very suitable when considering irregular shapes in real applications, such as AM. Another less common approach is considering the pieces' free rotation. This approach is more complex to handle due to the solution space increase; however, it may lead to better solutions. This study adopts the pieces' free rotation. This study's problem is packing a polyhedra set (convex and concave) into a minimal volume cuboid, adopting the pieces' free rotation constraint; this is known as the polyhedra packing problem (PPP) (Romanova et al., 2018), which is an NP-hard problem (Chazelle et al., 1989); therefore, the use of approaches alternative to mathematical programming may be appropriate. The solution approach in this work is a simulation of movements that can be framed within a metaheuristic. We used Unity® as the simulation environment and *tabu search* as the metaheuristic framework.

The article is organized as follows: the problem description and classification are presented in Section 2. In Section 3, a summary of existing related work is provided. The details of the simulation and *tabu search* are shown in Section 4. Section 5 presents the benchmarks used to validate the algorithm and the computational results, along with their corresponding analysis. Finally, the conclusions are shown in Section 6.

2. Problem Description

The problem addressed in this paper is the PPP in which a polyhedra set is packed within a cuboid and the objective is to minimize the cuboid's volume. This problem does not consider a fixed orientation of the items to be packed, i.e., free rotation of the items in all axes is allowed. The cuboid has no fixed dimensions, i.e., the cuboid's width, length, and height can change in order to reduce its volume. This problem considers two placement constraints, the non-overlapping constraint and the containment constraint (Romanova et al., 2018). The former refers to the fact that each pair of polyhedra cannot share any common interior points; however, the polyhedra may touch. The later constraint implies that each polyhedron has to be fully contained within the limits of the cuboid.

Researchers have developed some typologies to classify the C&PP; three of them are suitable for the PPP. The first is the one developed by Dyckhoff (1990), in which, using his notation, the PPP classifies as 3/V/O/F, 3/V/O/M, or 3/V/O/R. The four attributes are associated as follows:

- Dimensionality: "3" means that the problem is three-dimensional.
- Kind of assignment: "V" denotes that all the small items are placed into a set of large objects.
- The assortment of large objects: "O" is for only one large object.
- The assortment of small items: the fourth attribute depends on the instances used: "F" stands for a few small pieces of varying figure types. "M" implies that there are many small items of different figure types. "R" means that there are many small items of relatively few figure types.

The second typology considered is the one developed by Wäscher et al. (2007), which arises due to the insufficiency of Dyckhoff's typology with respect to the inclusion of some emerging development in the field of C&PP. This second typology is an improvement of the first one, particularly in the criterion concerning the assortment of large objects. In this way, the improved typology has five criteria in which the PPP is categorized, as follows:

- Dimensionality: the problem is three dimensional.
- Kind of assignment: in the PPP, all the small items are assigned to a set of large objects; therefore, the kind of assignment is input minimization.
- The assortment of small items: this criterion, once again, depends on the nature of the instances used; therefore, it may be a weakly or strongly heterogeneous assortment.
- The assortment of large items: in this problem, there is only one large object, the cuboid.
- The shape of small items: the objective of the PPP is to manipulate small irregular items; nonetheless, small regular ones can also be used.

Moreover, within the typology of Wäscher et al. (2007), the C&PP can be understood under the concepts of basic, intermediate, and refined problem types. The classification of basic problem types uses the combination of the criteria kind of assignment and assortment of small items; therefore, the PPP is an Open Dimension Problem (ODP). The intermediate problem types are mapped further by considering the criterion of the assortment of large objects, and the refined problem types are broken down further by considering the last two criteria (dimensionality and shape of small items). However, according to Wäscher et al. (2007), there are not many open dimension problems; therefore, the ODP categorization remains the same.

The third and last typology was developed by Araújo et al. (2020) as a refinement of the typology proposed by Wäscher et al. (2007) for AM-related problems such as the PPP. This novel typology uses four criteria, as the typology suggested by

Dyckhoff, but these include more AM-related information. The PPP under this new typology, in the notation of Araújo et al. (2020), would be 3/Si/Oo/A. This notation has the following meanings for each attribute:

- Dimensionality: "3" stands for a three-dimensional problem, which is the usual case in AM.
- Optimization criterion: "Si" is Single input minimization, which addresses problems with only one container with one or more variable dimensions, and the objective is to find a configuration of small items that minimizes the variable dimensions.
- Build volume type: "Oo" means one container with an open Z-height or variable-length dimension.
- Attributes of the assortment of small items: "A" describes the presence of multiple instances with different characteristics, particularly in terms of demand variation and complexity of the objects (small pieces).

3. State of the Art

The *strip packing problem* (SPP) is one of the most researched *open dimension problems* (ODP). In this problem, a set of 2D small items are placed within a large rectangular object, which has a fixed width and a variable length. Then, the objective is to minimize the length. The SPP has been studied with regular-shaped small items (SPP) by Martello et al. (2003), Bortfeldt (2006), and Alvarez-Valdes et al. (2008), and *irregular*-shaped small items (ISPP) by Gomes and Oliveira (2006), Leung et al. (2012), Alvarez-Valdes et al. (2013), and Cherri et al. (2016). These problems (SPP and ISPP) have their correspondence in a higher dimension (3D). In this way, the *three-dimensional strip packing problem* (3D-SPP) is the problem in which a given set of small items has to be packed into a cuboid, with two fixed dimensions (i.e., width and length) and a variable dimension (height); therefore, the objective is to minimize the height of the cuboid. This problem has been studied with regular-shaped small items (e.g., boxes) (3D-SPP) by Bortfeldt and Mack (2007), Allen et al. (2011), and Wei et al. (2012); and with *irregular*-shaped small items (3D-ISPP) by Stoyan et al. (2004), Egeblad et al. (2009), and Liu et al. (2015).

Within the ODPs, there are other problems in which all dimensions of the large object are variable (e.g., width, length, and height of a cuboid). Therefore, these are a generalization of the ODPs mentioned until now. Thus, in 2D, the problem of minimizing the size (i.e., area) of a container (e.g., rectangle) in which a set of small items are packed is known as the *minimal enclosure problem* (MEP) (Milenkovic and Daniels, 1999). The correspondence of the MEP in 3D is the PPP (Romanova et al., 2018). Relevant studies related to 3D-ODP (PPP included) are expanded below. Egeblad et al. (2009) developed a solution method for the multidimensional strip packing problem (3D-SPP included), as a dimensional generalization of the method presented in Egeblad et al. (2007). This approach uses a heuristic technique and considers fixed orientations of the small items. The algorithm is composed of a local search procedure, and the metaheuristic *guided local search* (GLS). The algorithm begins with the placement of the small items that may have overlap. The overlap is iteratively reduced with the local search procedure by translating one piece at a time to a minimum overlap position in an axis-aligned way. Liu et al. (2015) proposed the heuristic algorithm HAPE3D for the 3D-SPP based on the principle of minimum potential energy, which states that a body will be displaced to a position that minimizes its total potential energy. The HAPE3D is also based on the constructive algorithm HAPE (Liu and Ye, 2011), which is a solution approach used for the 2D irregular packing problem (nesting). A remarkable distinction of the HAPE3D is that it does not need to calculate the no-fit polygon nor to disassemble a polyhedron into simpler ones. This algorithm allows multiple orientations of the small items. Romanova et al. (2018) developed a solution algorithm called COMPOLY that compacts polyhedra and found a local optimum by solving a mathematical model based on quasi-phi-functions, a concept introduced in Stoyan et al. (2016). Therefore, they formulated the PPP as a nonlinear programming problem considering continuous rotations and translations of the small items. Ma et al. (2018) proposed an efficient method to obtain sub-optimal solutions for packing 3D small items into 3D containers, considering free rotation of the small items. The solution is optimization-based due to the fact it combines continuous local optimization and combinatorial optimization. The former iteratively adjusts the positions and orientations of the small items to obtain a tight packing. The latter considers operations such as swapping, replacement, and hole filling to escape from local optima. Egeblad et al. (2009) and Liu et al. (2015) proposed approaches for the 3D-SPP and adopted the item's fixed orientations constraint. This problem differs from the PPP in two aspects: the cuboid's dimensions that can change and the rotation constraint. First, in the 3D-SPP, the cuboid's volume changes through the variation of a cuboid's dimension (height), whereas, in the PPP, the cuboid's volume changes through the variation of a cuboid's dimensions (width, length, and height). Second, regarding the rotation constraint, Egeblad et al. (2009) and Liu et al. (2015) considered the small items to have fixed orientations, whereas this study considers the small items to rotate freely in all axes. Ma et al. (2018) focused on a problem that has significant differences when compared with PPP. First, the problem studied by Ma et al. (2018) considers arbitrary shapes of the container (including cuboid), besides the arbitrary shape of the items to pack. Second, the container does not change its dimensions, i.e., the container's shape and dimensions are fixed. Therefore, all small items may not be placed within the container. However, the algorithm developed by Ma et al. (2018) can be used to solve the PPP (and 3D-SPP) by repetitively changing the container's dimensions and verifying that all the small items are placed within the container. Nevertheless, this procedure would be very time-consuming. Using the description of PPP in the previous section (Problem description), this problem has only been studied by Romanova et al. (2018) to the best of the authors' knowledge. Their method is based on nonlinear programming, which offers outstanding solutions; however, it takes considerable time. In contrast, the methodology proposed in this paper gets satisfactory solutions in a short time, taking advantage of a simulation environment.

4. Solution Methodology

The proposed solution is a set of movements (translation and rotation) that are simulated in Unity®. These movements can be framed within a structure of a metaheuristic such as *tabu search* (TS). The proposed approach is explained in four subsections: Section 4.1 shows the most relevant Unity features, the simulation environment used in this study, for the PPP. Section 4.2 shows the base solution algorithm that includes all movement simulations without a TS structure hybridization. Finally, Section 4.3 presents the hybridization of the base model with the *probabilistic tabu search* (PTS).

4.1. Simulation environment

In the past, simulations were just a tool used to evaluate a solution's quality or a way to manage uncertainty (Talbi, 2009). However, the latest technological advances have made simulation an appealing first choice to face a problem (Lucas et al., 2015; Martínez et al., 2018a,b). These technological advances have resulted in robust simulation engines that offer several useful functionalities; this is the case with Unity. Unity is a suitable tool for 3D simulations. Unity is a multiplatform game/simulation engine developed by Unity Technologies that handles simulation projects in 2D and 3D. For 3D simulations, Unity uses the *physics processing unit* PhysXTM, which has been verified to represent physical phenomena accurately (Martínez & Álvarez, 2019). This validation is remarkable since PhysX uses a fixed time-step, i.e., all the physical phenomena calculations happen within a time frame (0.03 seconds in this study), which may approximate the calculations. Unity has several features that are useful to obtain solutions of the PPP. The Unity features used in this study were the following:

- *Controlled translations of bodies in the simulation*. Unity allows translating each body in the simulation in any direction, i.e., a product of the linear combination of the Cartesian axes' unit vectors ($\hat{e}_x$, $\hat{e}_y$, and $\hat{e}_z$). In this study, this feature was only applied to the cuboid and only in axis-aligned directions (see Fig. 1). The details of the translational movements are explained in Section 4.2.3.

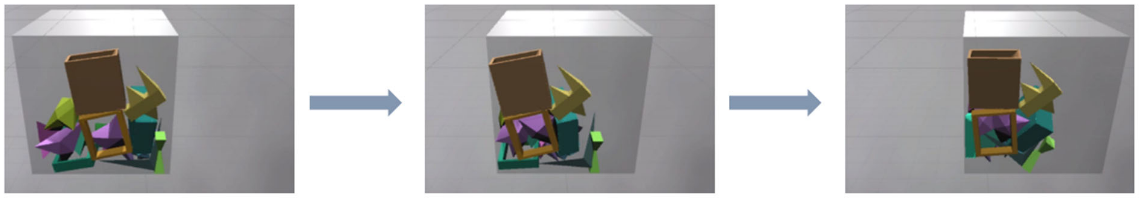

Fig. 1. Translation in the x-axis using the instance SGPS04nc20a

- *Controlled rotations of bodies in the simulation*. Each body can rotate at any angle to any Cartesian axis. The rotation calculations use quaternions, which do not suffer from gimbal lock and can easily be interpolated. In this study, this feature was only applied to the cuboid and in right angles (see Fig. 2). The details of the rotations are explained in Section 4.2.3.

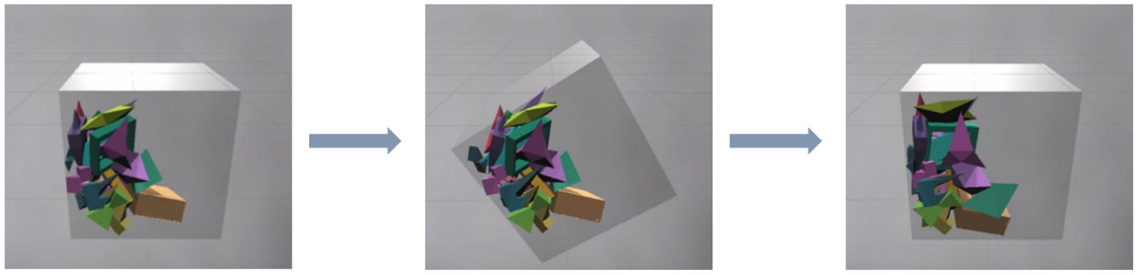

Fig. 2. Rotation in the $\hat{e}_x$ direction using the instance SGPS04nc20a

- *Compaction and expansion of bodies in the simulation*. The bodies in the simulation can be scaled up and scaled down. In this study, these features were only applied to the cuboid. The cuboid's volume changed when its dimensions (width, length, and height) varied. The details of the cuboid's compaction and expansion are explained in Sections 4.2.2. and 4.2.4.
- *Mass specification of bodies in the simulation*. The bodies in the simulation can have a mass specified by the user. In this study, the items' mass within the cuboid was assumed to be the same value as the items' volume. The volume of the items was estimated through the method proposed by Zhang and Chen (2001).
- *Gravity-like accelerations on bodies in the simulation*. Unity allows adding multiple accelerations to some bodies in the simulation that cause them to move accordingly. The user can change both the magnitude and direction of the accelerations, even on runtime. In this study, three accelerations were constantly applied to all the items within the cuboid. The magnitude of all three accelerations was the same and equal to 9.81 m/s2, and each one was applied to

the negative direction of each axis, i.e., $-\hat{e}_x$, $-\hat{e}_y$, and $-\hat{e}_z$. Therefore, the items within the cuboid tended to cluster in one cuboid's vertex.

- *Free translations and rotations of the bodies in the simulation*. PhysX calculates the positions and rotations of the bodies in the simulation according to the sum of forces applied to each body; therefore, this feature is one of the main reasons for using Unity. In this study, the software automatically calculates the positions and orientations of the items within the cuboid when the cuboid translates and rotates.
- *Collision detection between the bodies in the simulation*. PhysX resolves contacts through a two-phase collision detection that prevents two bodies from overlapping (Unity Technologies, 2019). Therefore, this feature is used to verify a solution's feasibility, i.e., the non-overlapping and containment constraints. The two-phase collision detection is composed of a broad and a narrow phase. In the former, fast algorithms are used to discard pairs of bodies far away from each other to pass to the narrow phase. The pairs of bodies that move to the later phase (narrow) are meticulously checked for a collision using more sophisticated algorithms. For further information about two-phase collision detection, refer to LaValle (2006).
- *Elimination and insertion of bodies in the simulation*. In this study, the insertion of bodies is used to generate the initial solution from which the simulation can begin. Moreover, the elimination and insertion of the items within the cuboid are used to change the solution neighborhood. The details of the elimination and insertion of bodies applied to PPP are found in Section 4.2.4.

4.2. Base model

The base model without a TS structure hybridization is shown in Algorithm 1.

Algorithm 1. *Base model*
```
1:  set an initial solution $S_0$
2:  compact the cuboid to get $S_0^c$ and $Vol(S_0^c)$
3:  set $S \leftarrow S_0^c$, $S^* \leftarrow S_0^c$, $Vol(S_0^c)$, $IterUnimproved \leftarrow 0$
4:  while time limit $maxTime$ not reached do
5:      select the next movement $M$
6:      compact the cuboid to get $S_M^c$
7:      set $S \leftarrow S_M^c$ and get $Vol(S)$
8:      if $(Vol(S) < Vol^*)$ then
9:          set $Vol^* \leftarrow Vol(S)$, $S^* \leftarrow S$, $IterUnimproved \leftarrow 0$
10:     else
11:         $IterUnimproved = IterUnimproved + 1$
12:     end if
13:     if $(IterUnimproved = maxUnimprovedIterations)$ then
14:         Apply diversification over $S$ randomly choose either $Shaking$ or $Insertion$
15:         $IterUnimproved \leftarrow 0$
16:     end if
17: end while
```

The notation used in Algorithm 1 is the following: S_0 is the initial solution, S_0^c is the solution obtained after submitting the S_0 to the compaction procedure, $Vol(S_0^c)$ is the volume of the cuboid (objective function) that is obtained with S_0^c, S is the current solution, S^* is the incumbent solution, $Vol(S^*)$ is the value of the objective function of S^*, $IterUnimproved$ is the number of iterations without improvement of the incumbent, $maxTime$ is the maximum time allowed for the algorithm, M is the movement that is executed, S_M^c is the solution obtained after submitting the solution found with M to the compaction procedure, $Vol(S)$ is the value of the objective function of S, and $maxUnimprovedIterations$ is the maximum number of iterations without improvement of the incumbent allowed.

The general algorithm flow is described as follows: first, in line 1, a feasible solution is generated by one of the two considered methods to create an initial solution; these methods are described in Section 4.2.1. Next, in line 2, the cuboid undergoes the compaction procedure to reduce its size; the compaction procedure is detailed in Section 4.2.2. Then, in line 3, the current and the incumbent solutions are updated with the solution obtained after the compaction procedure, the incumbent cuboid's volume is updated with the after-compaction solution's volume, and $IterUnimproved$ is set to zero. Later, in line 4, the loop with a time-stopping criterion begins. Then, in line 5, the following movement's selection and execution happen; information related to the movements is found in Section 4.2.3. Then, in line 6, the compaction procedure takes place (see Section 4.2.2). Later, in line 7, the current solution and its volume are updated. Then, in line 8, it is verified if the current solution's volume is better (i.e., smaller) than the incumbent volume. If this condition is satisfied, then the incumbent solution and volume are updated, and $IterUnimproved$ is set to zero (line 9). Otherwise, $IterUnimproved$ increases by one (line 11). Finally, in line 13, it is verified if $IterUnimproved$ has not exceeded $maxUnimprovedIterations$; if this is the case, then one of the two diversification movements ($Shaking$ or $Insertion$) happens (line 14), and $IterUnimproved$ is set to zero (line 15). The diversification movements are described in Section 4.2.4.

4.2.1. Initialization methods

In this study, two methods to get an initial solution were considered: the box-related method and the sphere-related method. These methods are explained below

Box-related method. This method replaces each piece to be packed with a box, which in this case corresponds to the *axis-aligned bounding box* (AABB) (the procedure for obtaining the AABB is explained below). After each piece is replaced with a box, an algorithm is used to pack the boxes within the cuboid. The chosen algorithm in this study was the one proposed by Martínez et al. (2015) due to the flexibility and efficiency of its solutions.

The algorithm proposed by Martínez et al. (2015) had a constructive phase and an improvement phase. Throughout the first (constructive) phase, the algorithm returned at least one packing pattern, mainly composed of the boxes' positions and some feasibility indicators. These packing patterns tended to bring pieces of similar shapes together.

It is worth noting that the algorithm proposed by Martínez et al. (2015) was developed for the *container loading problem*, in which the container's dimensions are known, i.e., the container's dimensions are parameters of the algorithm. Therefore, an initial container size was supposed, and it was iteratively increased until the feasibility indicators stated that all boxes were successfully packed within the container (cuboid). The supposed container was a cube with an edge equal to the average of the widths, lengths, and heights of all AABBs, multiplied by a factor. This factor initially corresponded to applying the ceiling function to the cubic root of the number of boxes (pieces). The container's size increase happened when the factor mentioned above increased by one in each iteration.

Through the AABB procedure, each piece is enclosed within the smallest *axis-aligned bounding box* (AABB). This box is generated with the highest and lowest values of the pieces' points at each Cartesian axis. Therefore, each AABB is formed by six points and has six faces parallel to a Cartesian axis.

Fig. 3 shows the process of creating AABB of two polyhedra used by the container loading algorithm. Fig. 4 shows the process of replacing the AABBs with their respective small items in the positions obtained with the container loading algorithm.

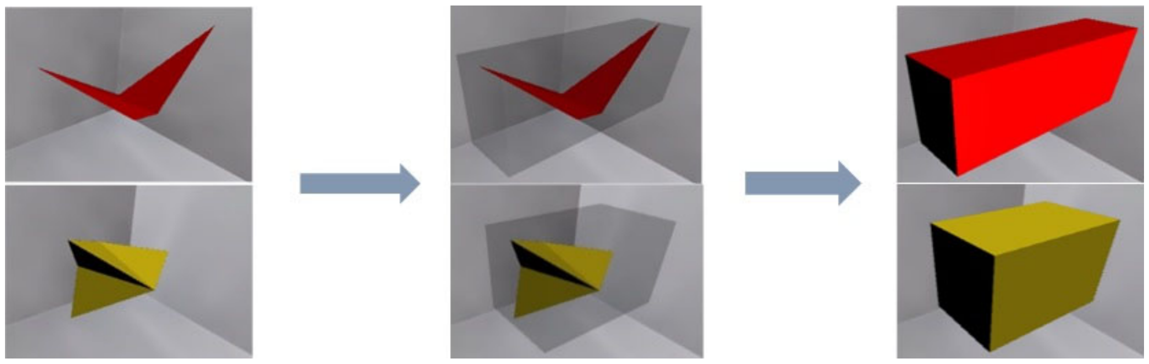

Fig. 3. Construction of AABB of two polyhedra

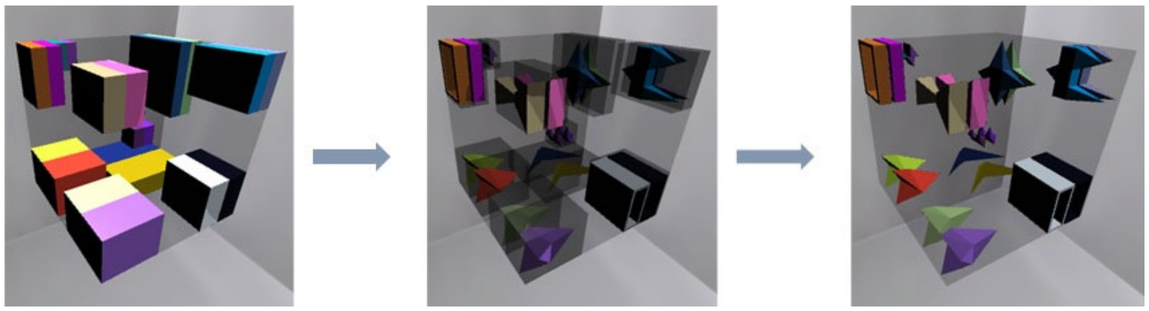

Fig. 4. Replacement of the packed AABB, packed by the algorithm presented in Alvarez Martínez et al. (2015), by the corresponding small items. The instance used in the example is SGPS04nc20a

Sphere-related method. This method replaces each piece with the smallest enclosing ball computed with the fast algorithm proposed by Fischer et al. (2003). The largest ball replaced all the small items to facilitate the initial items' placement within the cuboid in this study. This placement was a product of locating the balls next to each other, forming an equal-sided cuboid, which happened when the number of small items was equal to the third power of an integer, e.g., 8, 27, or 64. However, in the case when the number of small items was not equal to an integer's third power, the cuboid could be equal-sided or could have two equal sides and a different one. In this situation, the cuboid's size began by considering the smallest equal-sided cuboid. Then it was analyzed if all the items are contained within the cuboid with one or two shortened sides. The length of a side was measured with the distance of the number of large balls covering the side, e.g., a side can be 2, 3, or 4 large balls long. Therefore, the shortening of a side was the decrease of the number of large balls by one, e.g., a side that was three large balls long was the shortened side of a four large balls long side.

Fig. 5 shows the substitution of the small items by the largest enclosing ball, and Fig. 6 shows the replacement of the spheres with the small items.

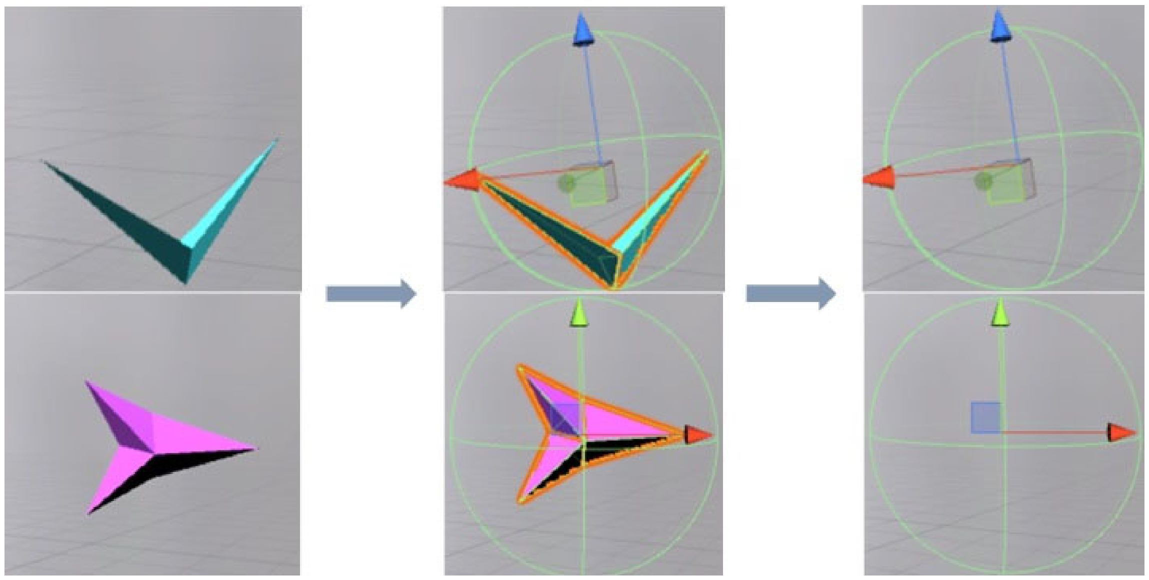
Fig. 5. Enclosing of the small items into the largest enclosing ball

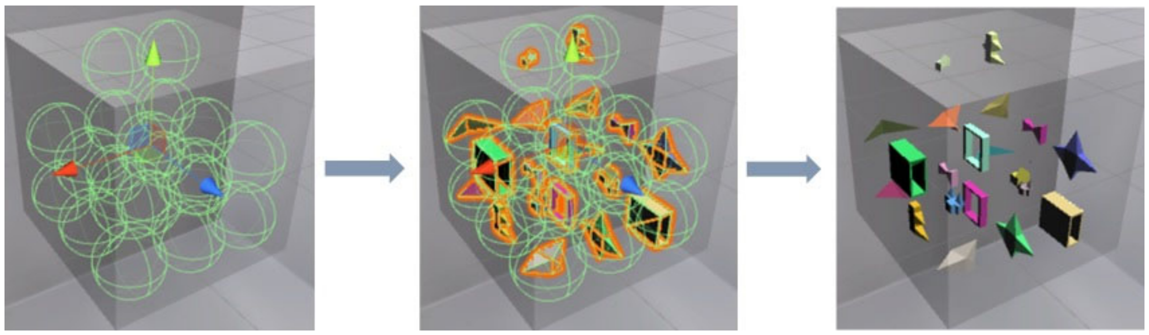
Fig. 6. Assignation of small items to the spheres and then the elimination of the spheres

4.2.2. Compaction procedure

In this study, the compaction procedure is the most crucial section for the PPP since it reduces the cuboid's size. The size reduction happened by moving the cuboid's faces towards the center of the cuboid. Three of the faces (one per each axis) are momentarily assigned a mass, and each face is affected by one gravity acceleration only, the one with the same direction as the face. Therefore, in each axis, one face pressed the small items against a fixed face not affected by gravity accelerations. The compaction procedure finished 3 seconds after all gravity-affected faces detected that they had touched a small item; therefore, the time is dependent on the configuration of the pieces at that moment, but it was unlikely to last more than 6 seconds in total.

The mass assigned to the three faces was half the sum of the items' mass.

4.2.3. Movements and selection of movements

This study considered nine cuboid movements: three axis-aligned translational movements, and six axis-aligned rotational movements. There was one translation per each axis and two rotations per each axis (negative and positive directions). Each translational movement was composed of three serial translations, which left the cuboid in the same original position. In the first translation, the cuboid displaced half the cuboid's dimension (width, length, and height) parallel to the axis in which the translations happened. This first translation was in the axis' positive direction. The second translation happened in the axis' negative direction, and the displacement was equal to the cuboid's dimension parallel to the axis. Finally, the third translation occurred in the axis' positive direction, and the displacement was half the cuboid's dimension parallel to the axis in which the translation happened. Each translational movement composed of three serial translation lasted 11.25 seconds. In each serial translation, the cuboid displaced 0.8% of the total displacement in each time window, which in this study was specified by the fixed time step and was 0.03 seconds long. Therefore, each of the serial translations lasted 3.75 seconds, and it was independent of the instance. Each rotational movement was composed of two serial movements: one rotation and one translation, which left the cuboids in the same position as before the rotation. The first serial movement was a 90 degree rotation from an axis, after which a fast translation happened to take the cuboid to the same position as before the beginning of the rotation. It is worth noting that the fast translation is unnecessary when the center of the cuboid meets the Cartesian origin when rotating; otherwise, it is necessary to maintain a single reference point. Each rotational movement composed of the two serial movements lasted 4.125 seconds. In the rotation, the cuboid rotated 0.8% of the total rotation angle in each time window. Therefore, the rotation lasted 3.75 seconds, and it was independent of the instance. The fast translation in this study was ten times faster than a regular translation; therefore, the fast translation lasted 0.375 seconds. In the base model, the movements were cyclically selected in the following order: translation in the x-axis (T_x), translation in the y-axis (T_y), translation in the z-axis (T_z), rotation in the negative x-axis (R_x^-), rotation in the positive x-axis (R_x^+), rotation in the negative y-axis (R_y^-), rotation in the positive y-axis (R_y^+), rotation in the negative z-axis (R_z^-), and rotation in the positive z-axis (R_z^+).

4.2.4. Diversification movements

In this study, two diversification movements were implemented: *Shaking* and *Insertion*. The diversification movements aim to abruptly change the small items' positions and orientation to change the solution neighborhood. *Shaking* had three phases: an expansion, an abrupt rotation, and a fast translation. The expansion caused the cuboid's size to increase (worsening the objective function value) by moving the cuboid's faces away from the cuboid's center. The cuboid's faces moved away from the cuboid's center such that the cuboid became a cube with a side value of 1.2 times the maximum cuboid's dimension (with, length, and height). The abrupt rotation was the simultaneous execution of R_z^- and R_x^-. Since these rotations were simultaneous, the time they took was the same as the time that one single rotation would have taken (3.75 seconds). The expansion happened in the same time frame, in which the abrupt rotation began; therefore, the expansion did not take additional time. Finally, the last phase was the fast translation that was the same as the one explained in Section 4.2.3.

On the other hand, *Insertion* had two phases: removal and reinsertion of items. In the former, the small items whose ratio between the number of contact points with the cuboid's faces divided by the number of contacting faces was greater than 0.5 and lesser than 1.5 were removed. Thus, the items with several contacting points with the cuboid's faces or those with no contact with the cuboid's faces were not removed. In the second phase, all the removed items were reinserted at the center of the cuboids. These two phases happened in a single time frame (0.03 seconds); however, two additional time frames were needed for PhysX to solve possible overlapping. Therefore, *Insertion* lasted 0.09 seconds to be executed, and it was independent of the instance.

4.3. Hybridization of simulations with a TS structure

The hybridization of simulations with metaheuristics is a flexible and powerful tool for optimization problems that allows obtaining good quality (near-optimal) solutions in reasonable computing time (Chica et al., 2017). The metaheuristic chosen to hybridize the simulations is *tabu search* (TS), specifically *probabilistic tabu search* (PTS) because it has been used to find efficient solutions to large combinatorial problems (Gendreau and Potvin, 2010). Furthermore, TS is organic to the base model since the TS's searching moves can be easily associated with the base model's nine movements described in Section 4.2.3.

PTS gets its name due to its probabilistic procedure in the selection of the movements. This procedure is explained in Section 4.3.3. The hybridization of the base model with the PTS is shown in Algorithm 2. The notation used in Algorithm 2 is the same as the one used in Algorithm 1 with the introduction of the following terms: $TabuList$ is the tabu moves list, $tabuTenure$ is the maximum number of elements in $TabuList$, i.e., the maximum number of iterations that a move has the tabu status; $\widetilde{N}(S)$ is the set of possible movements M' that can be applied to the current solution, and $\hat{f}$ is a function that estimates a usage probability to each M'.

Algorithm 2 has some steps equal to Algorithm 1; therefore, the steps that are different are described as follows: in line 5, it is verified if the length of $TabuList$ is equal to $tabuTenure$. The tabu status declaration is explained in Section 4.3.1. If the condition in line 5 is satisfied, then a move with tabu status is aspirated from $TabuList$ through the procedure described in

Section 4.3.2. Next, in line 8, a movement is randomly selected using the roulette method based on the probabilities assigned to each movement. These probabilities are obtained through the estimation procedure presented in Section 4.3.3. Finally, in line 11, $TabuList$ is updated, i.e., a move gets the tabu status according to the last executed movement, and the tabu move that stayed the longest in $TabuList$ is declared as no tabu.

Algorithm 2. *Hybridization of the base model with PTS*

```
1:  set an initial solution $S_0$
2:  compact the cuboid to get $S_0^c$ and $Vol(S_0^c)$
3:  set $S \leftarrow S_0^c$, $S^* \leftarrow S_0^c$, $Vol(S_0^c)$, $IterUnimproved \leftarrow 0$
4:  while time limit $maxTime$ not reached do
5:     if $(|TabuList| = tabuTenure)$ then
6:        Aspirate a Tabu move from the $TabuList$
7:     end if
8:     select the next movement $M$ using the roulette method with probability $\hat{f}(M')|M' \in \widetilde{N}(S)$
9:     compact the cuboid to get $S_M^c$
10:    set $S \leftarrow S_M^c$ and get $Vol(S)$
11:    update $TabuList$
12:    if $(Vol(S) < Vol^*)$ then
13:       set $Vol^* \leftarrow Vol(S)$, $S^* \leftarrow S$, $IterUnimproved \leftarrow 0$
14:    else
15:       $IterUnimproved = IterUnimproved + 1$
16:    end if
17:    if $(IterUnimproved = maxUnimprovedIterations)$ then
18:       Apply diversification over $S$ randomly choose either $Shaking$ or $Insertion$
19:       $IterUnimproved \leftarrow 0$
20:    end if
21: end while
```

4.3.1. Tabu status

A tabu move or a move with tabu status is a move that does not occur for some iterations (Gendreau and Potvin, 2010). The declaration of moves as tabu may avoid reversing the effect of recent moves. Therefore, the moves that are declared as tabu are those whose effect may reverse previous ones. For example, the R_x^+ movement may reverse the effect of the R_x^- movement if these movements are executed one after the other. This situation can be avoided if the R_x^+ movement gets a tabu status after the R_x^- movement execution. In this study, the tabu declaration depended on the type (rotational and translational) of the last executed movement. If the last executed movement was translational, the movement itself got the tabu status because it was the more likely movement to reverse its effect. On the other hand, if the last executed movement was rotational, the rotation movement in the same axis and the opposite directions got the tabu status. However, if the reverse rotational movement was already in the $TabuList$, the last executed movement itself got the tabu status to avoid a biased repetition of movements. Table 1 shows which movement got the tabu status according to the last executed movement.

Table 1
Declaration of tabu moves

Executed movement	***Tabu movement***
T_x	T_x
T_y	T_y
T_z	T_z
R_x^-	R_x^+
R_x^+	R_x^-
R_y^-	R_y^+
R_y^+	R_y^-
R_z^-	R_z^+
R_z^+	R_z^-

4.3.2. Aspiration procedure

The aspiration procedure is used to cancel the tabu status of a move that could be attractive (Gendreau & Potvin, 2010). In this study, this procedure happened in all iterations in which the condition in line 5 of Algorithm 2 was satisfied. In this procedure, each move with tabu status was assigned a probability value through the estimation procedure described in Section 4.3.3. Then, one of these moves was randomly chosen to be aspirated (the move was removed of its tabu status) through the roulette method. The aspirated move was then considered to be executed along with the moves without the tabu status. The roulette method is used to randomly select an element considering the probability associated with each element. In this study, the elements were the movements, and their probabilities were obtained from the estimation procedure (Section 4.3.3.). This method was used because it is recommended to apply randomness to the aspiration procedure (Gendreau and Potvin, 2010).

4.3.3. Estimation procedure

The estimation procedure was used to assign a probability to each movement. These probabilities depended on the number of considered movements and their estimated effect on the objective function value. These probabilities were used by the roulette method described in Section 4.3.2. to select a movement. In cases where the effect of a move over the objective function value is hard to determine, the effect can be estimated using a surrogate objective function. The PPP's objective function is the cuboid's volume minimization, and the effect that a movement would have on the cuboid's volume was hard to determine without performing it. Therefore, in this study, a surrogate function was used to estimate the effect of the movements. The used surrogate function was based on the concept of free space, which refers to the available space that the items have to move within the cuboid in a specific direction. This function was related to the original objective function assuming that if the free space was large in a specific direction, then the items had a large space to move and find a better pattern that would decrease the cuboid's volume. The value of the free space specified the probability of each movement. As the free space of a movement increased, the probability of choosing that movement through the roulette method also increased. The free space related to each cuboid movement (effect) was established with the free space values of two axis-aligned orientations ($\hat{e}_x$, $-\hat{e}_x$, $\hat{e}_y$, $-\hat{e}_y$, $\hat{e}_z$, and $-\hat{e}_z$). For translational movements, the effect was estimated as the mean of the free space values associated with the translation axis. Moreover, for rotational movements, their effect was estimated as the mean of the free space values associated with perpendicular orientations to the axis in which the rotation happened, maintaining the orientation's sign. Table 2 shows the established relationship between the free space associated with each movement with the free space of each direction.

Table 2
Assignation of the six values of free space (fs) to the nine movements.

Movement	***Estimation function***
T_x	$AVG(\bar{X}_{fs}-.\bar{X}_{fs}+)$
T_y	$AVG(\bar{Y}_{fs}-.\bar{Y}_{fs}+)$
T_z	$AVG(\bar{Z}_{fs}-.\bar{Z}_{fs}+)$
R_x^-	$AVG(\bar{Y}_{fs}-.\bar{Z}_{fs}-)$
R_x^+	$AVG(Y_{fs}+.\bar{Z}_{fs}+)$
R_y^-	$AVG(\bar{X}_{fs}-.\bar{Z}_{fs}-)$
R_y^+	$AVG(\bar{X}_{fs}+.\bar{Z}_{fs}+)$
R_z^-	$AVG(\bar{X}_{fs}-.\bar{Y}_{fs}-)$
R_z^+	$AVG(\bar{X}_{fs}+.Y_{fs}+)$

In this study, two methods to determine the free space were considered: one based on the φ-functions and the other one based on AABBs. In the former, φ-functions were used to determine the distance between items considering the items' actual shape. In the latter, the distance was determined between the AABBs which enclosed the items. The φ-functions return a value that corresponds to the interaction between a pair of phi-objects (particular geometric objects); the value is positive for non-overlapping objects, zero for touching objects, and negative for overlapping objects (Chernov et al., 2010). The φ-functions were constructed using the procedure presented in Stoyan et al. (2002) with the algorithm's assistance based on Barber et al. (1996). Unfortunately, the φ-functions take considerable time to be computed due to their mathematical complexity. The computation can even take several minutes, which was not suitable for a fast estimation using a surrogate function. Therefore, these functions were discarded from this study, and the only considered estimation method was the AABBs-based, which is much less sophisticated than the method based on φ-functions. There were two cases in the AABBs-based estimation method considering overlapping between a pair of AABB: overlap and no overlap. The second case happened when there was overlapping at most on one Cartesian plane in which the boxes were orthogonally projected; otherwise, the first case happened. The AABBs-based method estimated the free space using one-axis distances, which depended on the boxes' overlapping case. There were six one-axis distances for each AABB, one per each face. When there was no overlap in the face's direction, the one-axis distance was calculated as the distance with the nearest box (other AABB or cuboid) in the face direction. Therefore, the free space increased as the one-axis distance increased. Otherwise, when there was overlap in the face's direction, then the one-axis distance was calculated as the inverse distance between the face and the overlapping face. Therefore, the free space decreased as the overlapping between the AABBs increased because that would mean that the pieces are close.

5. Computational experiments and results analysis

To demonstrate the efficiency of the proposed algorithms, we have run and compared it with other methodologies. All simulations were run on an Intel® CoreTM i7-7700HQ CPU @ 2.80 GHz computer with a 16 GB RAM, an NVIDIA GeForce GTX 1050 Ti GPU, and a Windows 10 operative system. The programming language was C#.

5.1. Instances

The polyhedra sets were named using the authors' last name and the publication year of the paper in which the polyhedra information was found. Therefore, the polyhedra sets' names were formed with the initial of each author's last name, followed by the last two digits of the publication year. Each instance was coded using the name of the polyhedra set followed by two features: the convexity of the pieces and the number of pieces. If all pieces of the instance were convex, then a "c" appeared after the polyhedra set's name. Otherwise, if at least one piece of the instance was non-convex, then an "nc" appeared after the polyhedra set's name. Then, the name was followed by the number of pieces that formed the instance. Also, if two or more instances had the same name, they were differentiated by adding a letter at the end of the name. Table 3 shows all the instances found in the literature, whose items are composed of convex parts to the authors' best knowledge. These instances were classified according to the assortment of the pieces. The classification criterion was based on the ratio between the number of types of pieces and the number of pieces that composed the instance. If this ratio was less than 0.5, then the instance was considered a weak assortment of the pieces; Otherwise, if the ratio was greater than or equal to 0.5, then the instance was considered a strong assortment of the pieces.

Table 3
Details of the instances.

Instance	*Assortment*	*Composition*
SGPS04c80	*Weak*	*20 pieces of types 1, 2, and 4, and 10 pieces of types 3 and 5*
SGPS04nc20a	*Strong*	*2 pieces of each type*
SGPS04nc20b	*Weak*	*10 pieces of types 9 and 10*
SGPS04nc20c	*Weak*	*20 pieces of type 9*
SGPS04nc30	*Weak*	*3 pieces of each type*
SGPS04nc40	*Weak*	*4 pieces of each type*
SGPS04nc45	*Weak*	*5 pieces of types 1-5, and 4 pieces of types 6-10*
SGPS04nc50	*Weak*	*5 pieces of each type*
SGSPM05c7	*Strong*	*1 piece of each type*
SGSPM05c12	*Strong*	*1 piece of types 1-4, 3 pieces of types 5 and 7, and 2 pieces of type 6*
SGSPM05c25	*Weak*	*2 pieces of types 1 and 7, 3 pieces of type 6, 4 pieces of types 2, 4, and 5, and 6 pieces of type 3*
SGSPM05c98	*Weak*	*14 pieces of each type*
GP08nc14	*Strong*	*3 pieces of type 4, 2 pieces of types 1, 7, and 8, and 1 piece of types 2, 3, 5, 6, and 9*
LLCY15nc36	*Weak*	*8 pieces of types 1, 2, 4, and 5, and 4 pieces of type 3*
AÖAB19nc4	*Strong*	*1 piece of types 1-4*
AÖAB19nc7	*Strong*	*1 piece of types 5-11*

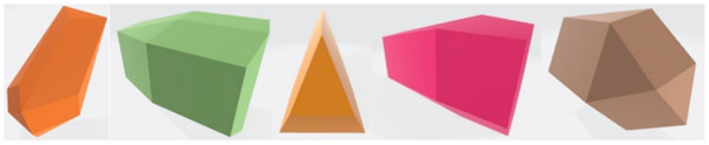
Fig. 7. Convex Pieces (1 to 5 from left to right) of the data set SGPS04 from Stoyan et al. (2004)

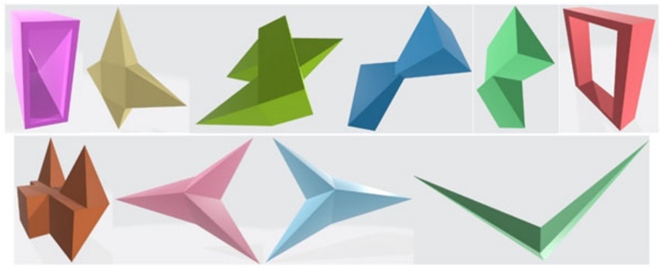
Fig. 8. Non-Convex Pieces (1 to 10 from left to right, from top to bottom) of the data set SGSPM04 from Stoyan et al. (2004)

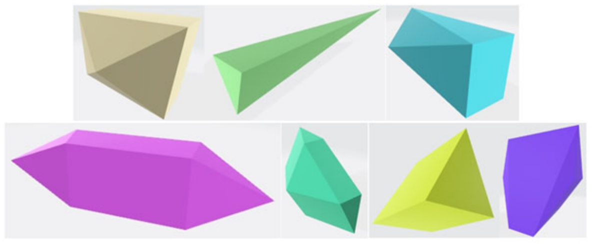
Fig. 9. Convex Pieces (1 to 7 from left to right) of the data set SGSPM05 from Stoyan et al. (2005)

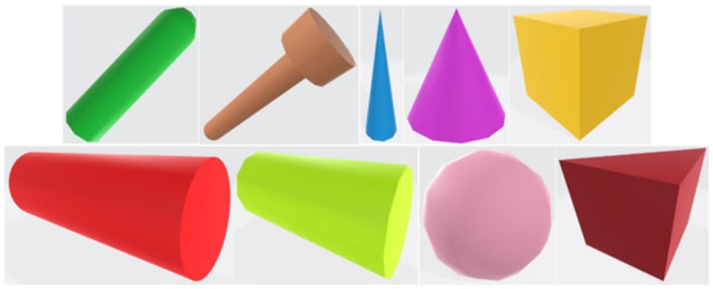

Fig. 10. Pieces (1 to 9 from left to right, from top to bottom) of the data set GP08 from Gogate and Pande (2008)

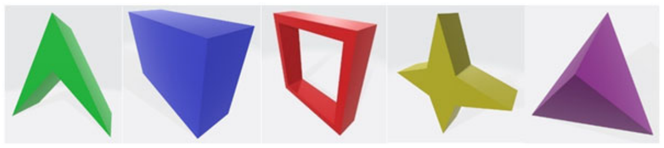

Fig. 11. Non-Convex Pieces (1 to 5 from left to right) of the data set LLCY15 from Liu et al. (2015)

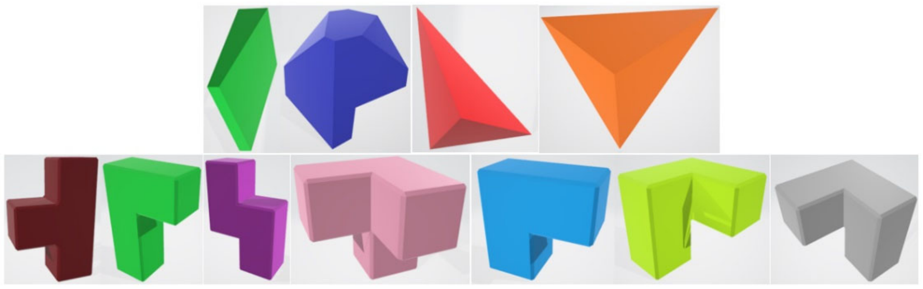

Fig. 12. Non-Convex Pieces (1 to 11 from left to right, from top to bottom) of the data set AÖAB19 from Araújo et al. (2020)

5.2. Combinations of parameters

Each instance was run ten times with each combination of factors. There were five factors: *maxTime*, *tabuTenure*, *maxUnimprovedIterations*, diversification, and initialization. The first factor (*maxTime*) was set for each simulation and used as a stopping criterion. This time can be arbitrarily set at the discretion of the users. However, to compare each combination of factors, the same *maxTime* was set to each simulation. This time was set considering the behavior of Algorithm 2 throughout time with four instances: AÖAB19*nc7*, SGPS04*nc50*, SGPS04*c80*, and SGSPM05*c7*. Simulations were run for 2000 seconds, and for each instance, the cuboid's volume was recorded each time it changed.

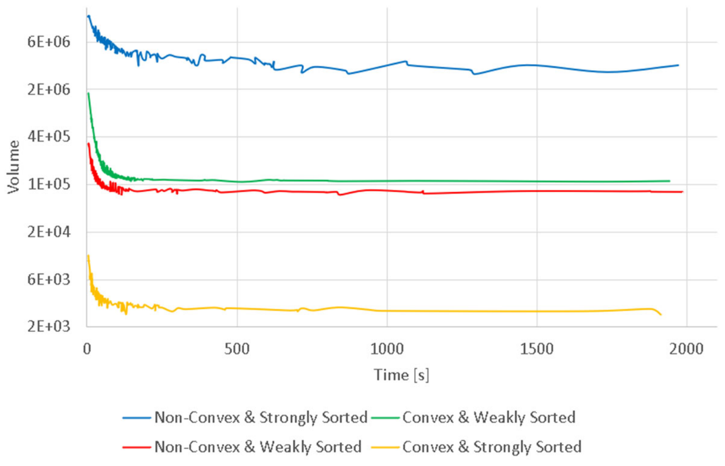

Fig. 13. Behavior of algorithm 2 throughout time on instances AÖAB19nc7 (non-convex strongly sorted), SGPS04nc50 (non-convex weakly sorted), SGPS04c80 (convex weakly sorted), and SGSPM05c7 (convex strongly sorted)

Fig. 13 shows how the cuboid's volume of four instances changed over time; at the beginning of the simulations, the cuboid's volume rapidly decreases, but the algorithm's capacity to find better solutions decreases as time passes. Furthermore, the most notorious changes in the cuboid's volume happened in the first 200 seconds. Therefore, *maxTime* was set to 200 seconds for each simulation.

The parameter *tabuTenure* was set to 2, 4, and 6, and *maxUnimprovedIterations* to 5 to 10.

The diversification was done using one of the two movements explained in Section 4.2.4 in two ways. The first one was named *Shak*, and it only used the *shaking* movement. The second one was named *Rand*, and it randomly chose either the *shaking* movement or the *insertion* movement with a 50% chance each. The initialization was done through one of the two methods explained in Section 4.2.1. The Box related method is hereinafter referred to as B, and the Sphere-related method is hereinafter referred to as S. These factors led to a complete factorial design of experiments with 24 different combinations for Algorithm 2 and 8 different combinations for Algorithm 1. Algorithm 1 had fewer combinations than Algorithm 2 because the parameter *tabuTenure* is not present in Algorithm 1.

5.3. Statistical analyses

An ANOVA was performed over the simulations' results of each combination with each instance using Minitab 19; the Pareto chart of the standardized effects is shown in Fig. 15. It is worth noting that the instances were treated as a block of the design experiment, and the interactions between the instances and the other factors were analyzed. Fig. 15 shows that evidently, there are significant interactions between the instances and other factors, which means that each instance may have a preferred combination of parameters. This situation could affect the determination of a single combination of parameters that have a good performance with any instance. When pairwise comparisons were performed using the Fisher LSD method with 95% confidence, significant differences were found between the results associated with the 32 treatments for each instance.

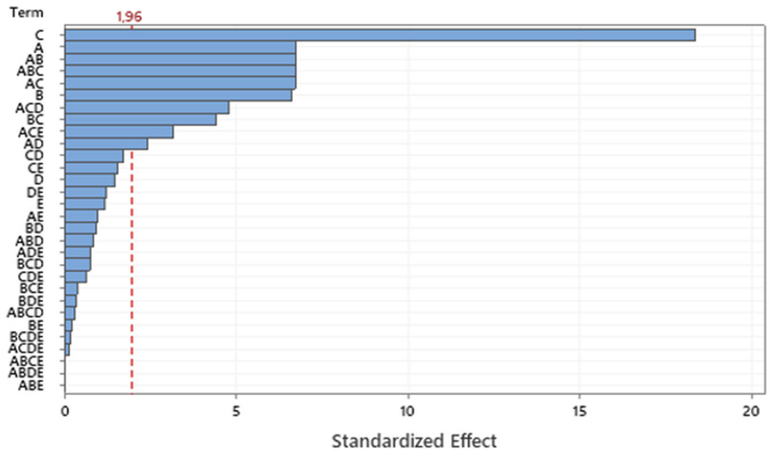

Fig. 14. Pareto chart of the standardized effects (Confidence = 95%). A = instance, B = tabuTenure, C = initialization, D = maxUnimprovedIterations, and E = diversification

Fig. 14 shows that *tabuTenure* and the initialization method directly affect the value of the objective function in contrast of *maxUnimprovedIterations* and the diversification method. However, these two last factors form significant 2-way and 3-way interactions.The parameter that differentiated Algorithm 1 and Algorithm 2 was *tabuTenure* because it was related to the hybridization with the PTS. Therefore, this factor's significance indicated the possibility of finding significant differences between Algorithm 1 and Algorithm 2 in some instances. In this sense, when reviewing the pairwise comparisons for each instance, results showed that in 11 out of the 16 instances, there was no significant difference between using Algorithm 1 and Algorithm 2, i.e., at least one combination of each algorithm was not significantly different in the best Fisher LSD group. The remaining five instances showed that the results obtained with Algorithm 2 were significantly better than the results obtained with Algorithm 1; these instances were: SGPS04*c20*, SGPS04*nc20a*, SGPS04*nc20b*, SGSPM05*c7*, and SGSPM05*c12*. The simulation approaches' initialization method significantly affected the final volume of the cuboid due to the huge difference in the initial volume of the cuboid. The *sphere-related method* always led to a larger initial cuboid's volume than the *box-related method*. Theoretically, the difference in the initial cuboid's volume should not significantly affect the final cuboid's volume since the volume rapidly decreased in the initial iterations. However, simulations could be negatively affected by this initial difference because when the cuboid's initial volume was large, all objects could reach higher speeds than the ones PhysX could handle. These higher speeds were reached because all objects within the cuboid were under the influence of gravity-like accelerations and had more space to move. When objects move fast, the simulations may become unstable since the collision calculations may not be appropriate. Therefore, PhysX could take more time to perform these calculations or even the simulations had to return to the last stable point and recalculate each object's physics within the simulation.

maxUnimprovedIterations and the diversification method were not a significant factor in the value of the objective function, which can be explained by the low number of iterations that occurred in each simulation. The expected time for a movement was 6.5 seconds, and the compression lasted at least 3 seconds; therefore, in a 200 seconds simulation, there would be near 20 iterations. Therefore, a diversification method would have happened 2 (with *maxUnimprovedIterations* = 10) or 4 times (with *maxUnimprovedIterations* = 5). The *maxTime* value and the movement's duration with the compression limited the possibility to determine the effect of *maxUnimprovedIterations* and the diversification method.

5.4. Analysis procedures

Two procedures were used to facilitate the solutions' analysis: one to determine the solutions' feasibility according to the non-overlapping constraint, and the other one to indicate how much intertwining existed when instances had pieces with holes. The first procedure is detailed in Section 5.4.1, and the second one is described in Section 5.4.2. It is worth noting that there is no metric for the considered intertwining to the authors' best knowledge. Furthermore, in this study, the metric GAP was used to compare the approach's results with published results. This metric is explained in Section 5.4.3.

5.4.1. Solutions' feasibility

This procedure estimates how much overlapping was in the solutions. Overlapping in the solutions was possible since PhysX may estimate the physics calculations due to the fixed time step feature. A four-step procedure estimated this overlapping:

1. A 3D grid was superposed on each solution's cuboid. The 3D grid was created by dividing each cuboid's dimension (width, length, and height) into 250 lines. Therefore, there were 15,625,000 points to verity for each solution.
2. For each point of the grid, the number of pieces containing it was determined.
3. The point was classified according to the number of pieces containing it. If any piece did not contain the point, it was classified as *blank*. If a single piece contained the point, it was classified as *right*. If two or more pieces contained the point, it was classified as *wrong*.
4. The overlapping was estimated as the percentage that wrong points represented of all no blank points.

5.4.2. Intertwining metric

This procedure was used to determine how much intertwining exists in a solution when the instance has pieces with holes.

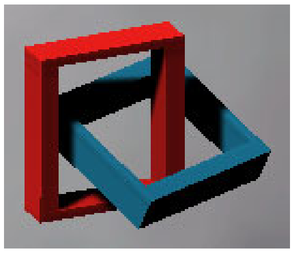
Fig. 15. Intertwining of two ring-like pieces

In this study, intertwining is considered to have occurred when ring-like pieces are interlaced (see Fig. **15**). In this study, the ring-like pieces are pieces with a convex hole formed by the union of the piece's parts. It is worth noting that all pieces with holes from the instances were ring-like pieces.

The intertwining metric is the percentage of intertwining pieces (IP), indicating how much of the total ring-like pieces intertwine. The procedure of obtaining IP is composed of the following three steps:

1. Determine the hole of each piece as a convex part. The points in which the inner faces of the ring-like piece intersect each other constitute the hole. The convex hull of these points constitutes the hole as a convex part.
2. Determine if a pair of ring-like pieces are intertwining. The pair of pieces are intertwining only if their holes simultaneously intersect with at least one convex part of the other piece. Otherwise, the pair of pieces are not intertwining.
3. Divide the number of intertwining pieces between the number of ring-like pieces in the instance.

5.4.3. Gap

This metric is used to determine how close the obtained results (r) are to published results (pr). This metric is calculated through Eq. (1).

$$Gap = \frac{r - pr}{pr} * 100 \tag{1}$$

A positive Gap implies that the obtained result was not better than the literature's result. In contrast, a negative Gap implies that the obtained result was better than the literature's result.

5.5. Results of convex instances

All simulations were run all convex instances with the following combination of parameters: Algorithm 2, *tabuTenure* = 4, *box-related method* as initialization method, *maxUnimprovedIterations* = 10, and the *Rand* diversification method. This combination was chosen because it statistically had the best results in four of the five convex instances

Tables 4 - 6 show the results of the 5 considered convex instances. Fig. 16 shows the layout of the best solution to the instance SGSPM05*c7,* Fig. 17 shows the layout of the best solution to the instance SGSPM05*c12*, Fig. 18 shows the layout of the best solution to the instance SGSPM05*c25,* and all of these results are shown in Table 4. Fig. 19 shows the layout of the best solution to the instance SGPS04*c80* and its results are shown in Table 5. Fig. 20 shows the layout of the best solution to the instance SGSPM05*c98* and its results are shown in Table 6.

Table 4
Results of the instances: SGSPM05*c7*, SGSPM05*c12*, and SGSPM05*c25*

Instance	*Stoyan et al. (2005)*		*Egeblad et al. (2009)*		*Proposed approach*			
	Volume	*Time [s]*	*Volume*	*Time [s]*	*Mean*	*Sd*	*Gap [%]*	*Overlap [%]*
SGSPM05c7	*3240.0*	-	*2317.2*	*600*	*2490.8*	*237.2*	*7.49*	*0*
SGSPM05c12	*6492.6*	-	*4164.3*	*600*	*4396.0*	*207.0*	*5.56*	*0*
SGSPM05c25	*11006.4*	-	*7156.8*	*600*	*8021.7*	*177.8*	*12.08*	*0*

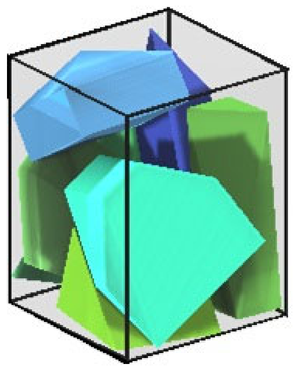
Fig. 16. Best solution obtained of the instance SGSPM05*c7*

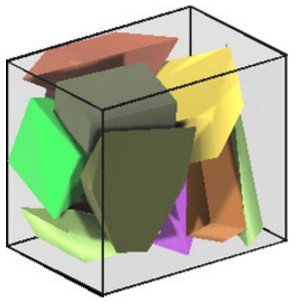
Fig. 17. Best solution obtained of the instance SGSPM05*c12*

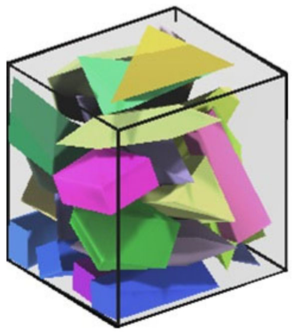
Fig. 18. Best solution obtained of the instance SGSPM05*c25*

Table 5
Results of the instance SGPS04*c80*

Instance	*Stoyan et al. (2004)*		*Proposed approach*			
	Volume	*Time [s]*	*Mean*	*Sd*	*Gap [%]*	*Overlap [%]*
SGPS04c80	*113582.3*	*3388.0*	*110288.7*	*1834.3*	*-2.90*	*0*

Table 6
Results of the instance SGSPM05*c98*.

Instance	*Stoyan et al. (2005)*		*Proposed approach*			
	Volume	*Time [s]*	*Mean*	*Sd*	*Gap [%]*	*Overlap [%]*
SGSPM05c98	*23113.06*	*147967.3*	*31265.1*	*708.4*	*35.27*	*0*

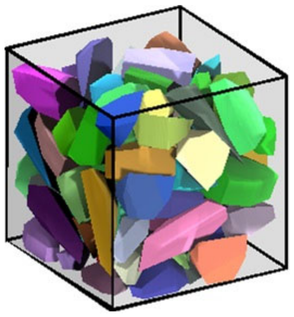

Fig. 19. Best solution obtained of the instance SGPS04*c80*

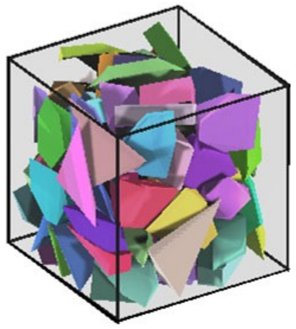

Fig. 20. Best solution obtained of the instance SGSPM05*c98*

When applied on convex instances, the results of the proposed methodology are competitive with results obtained with other methodologies.

5.6. Results of non-convex instances

All simulations were run all non-convex instances with the following combination of parameters: Algorithm 2, *tabuTenure* = 2, *box-related method* as initialization method, *maxUnimprovedIterations* = 5, and the *Rand* diversification method. This combination was chosen because it statistically had the best results in ten of the eleven non-convex instances.

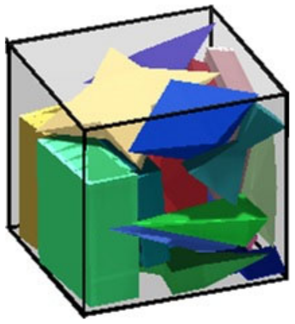

Fig. 21. Best solution obtained of the instance SGPS04*nc20*a

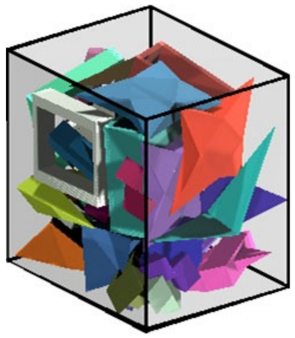

Fig. 22. Best solution obtained of the instance SGPS04*nc30*

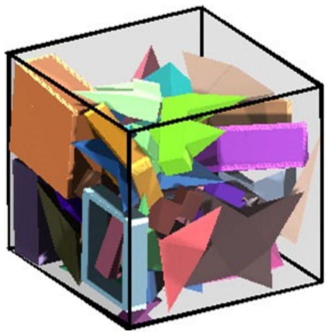

Fig. 23. Best solution obtained of the instance SGPS04*nc40*

Tables 7 - 10 show the results of the 11 considered non-convex instances. Fig. 21 shows the layout of the best solution to the instance SGPS04*nc20*a, Fig. 22 shows the layout of the best solution to the instance SGPS04*nc30,* Fig. 23 shows the layout of the best solution to the instance SGPS04*nc40*, Fig. 24 shows the layout of the best solution to the instance SGPS04*nc50*, Fig. 25 shows the layout of the best solution to the instance LLCY15*nc36*, and all of these results are shown in Table 7. Fig. 26 shows the layout of the best solution to the instance SGPS04*nc20*b, Fig. 27 shows the layout of the best solution to the instance SGPS04*nc20*c, Fig. 28 shows the layout of the best solution to the instance SGPS04*nc45,* and all of these results are shown in Table 8. Fig. 29 shows the layout of the best solution to the instance GP08*nc14* and its results are shown in Table 9. Fig. 30 shows the layout of the best solution to the instance AÖAB19*nc4*, Fig. 31 shows the layout of the best solution to the instance AÖAB19*nc7,* and these results are shown in Table 10. It is worth noting that Tables 7 and 8 include the IP metric because the instances displayed in the table are composed of at least two ring-like pieces.

Table 7
Results of the instances: SGPS04*nc20*a, SGPS04*nc30*, SGPS04*nc40*, SGPS04*nc50*, and LLCY15*nc36*

Instance	*Liu et al. (2015)*		*Romanova et al. (2018)*		*Proposed approach*				
	Volume	*Time [s]*	*Volume*	*Time [s]*	*Mean*	*Sd*	*Gap [%]*	*Overlap [%]*	*IP [%]*
SGPS04nc20a	*32550.0*	*26202.1*	*27432.6*	*34313.3*	*39080.5*	*2305.5*	*42.46*	*0.01*	*0*
SGPS04nc30	*48300.0*	*53741.5*	*40277.2*	*33008.9*	*54337.4*	*2837.7*	*34.91*	*0.08*	*0*
SGPS04nc40	*61950.0*	*99952.0*	*53158.9*	*201501.5*	*71233.6*	*3790.2*	*34.00*	*0.17*	*0*
SGPS04nc50	*77280.0*	*125210.6*	*61630.7*	*270654.8*	*89419.7*	*2470.1*	*45.09*	*0.10*	*40*
LLCY15nc36	*12480.0*	*9637.5*	*10461.7*	*23023.1*	*13751.2*	*351.6*	*31.44*	*0.12*	*0*

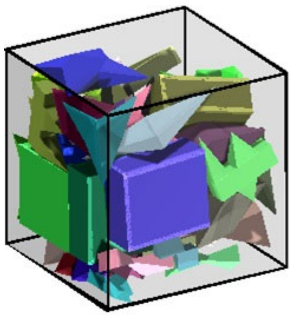
Fig. 24. Best solution obtained of the instance SGPS04*nc50*

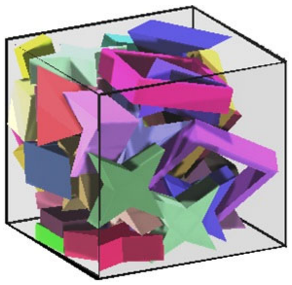
Fig. 25. Best solution obtained of the instance LLCY15*nc36*

Table 8
Results of the instances: SGPS04*nc20*b, SGPS04*nc20*c, and SGPS04*nc45*

Instance	*Romanova et al. (2018)*		*Proposed approach*				
	Volume	*Time [s]*	*Mean*	*Sd*	*Gap [%]*	*Overlap [%]*	*IP [%]*
SGPS04nc20b	*15275.5*	*8729.4*	*27625.2*	*1604.6*	*80.85*	*0.03*	*-*
SGPS04nc20c	*12805.7*	*59497.9*	*35416.0*	*1218.43*	*176.56*	*0.01*	*-*
SGPS04nc45	*61860.8*	*159884*	*83307.1*	*2920.7*	*34.67*	*0.07*	*50*

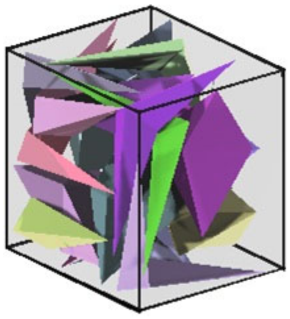
Fig. 26. Best solution obtained of the instance SGPS04*nc20*b

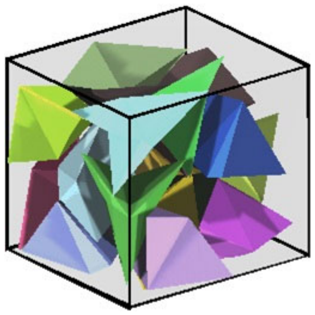
Fig. 27. Best solution obtained of the instance SGPS04*nc20*c

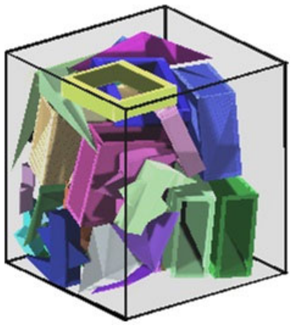
Fig. 28. Best solution obtained of the instance SGPS04*nc45*

Table 9
Results of the instances GP08*nc14*

Instance	*Gogate and Pande (2008)*		*Proposed approach*			
	Volume	*Time [s]*	*Mean*	*Sd*	*Gap [%]*	*Overlap [%]*
GP08nc14	*514.1*	*< 1800*	*440.6*	*13.3*	*-14.30*	*0*

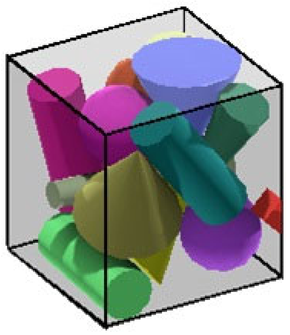
Fig. 29. Best solution obtained of the instance GP08*nc14*

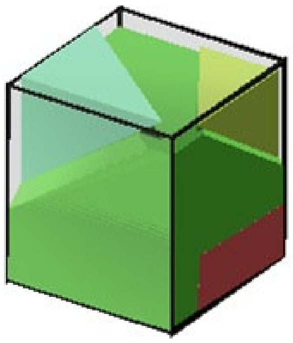
Fig. 30. Best solution obtained of the instance AÖAB19*nc4*

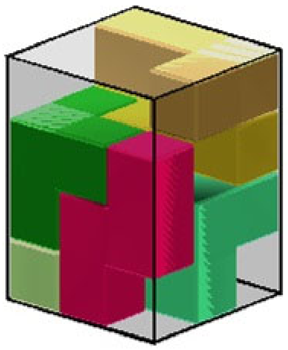
Fig. 31. Best solution obtained of the instance AÖAB19*nc7*

Table 10
Results of the instances: AÖAB19*nc4* and AÖAB19*nc7*

Instance	*Araújo et al. (2020)*		*Proposed approach*			
	Volume	*Time [s]*	*Mean*	*Sd*	*Gap [%]*	*Overlap [%]*
AÖAB19nc4	*1000000*	*< 140*	*1747869*	*290548.6*	*74.79*	*0.02*
AÖAB19nc7	*≈1430000*	*> 2000*	*3758747*	*309836*	*162.85*	*< 0.01*

When applied on non-convex instances, the proposed methodology results have a considerable gap compared with other methodologies. This situation particularly arises in two scenarios: when the assortment of the pieces is very weak, such as in the instances *SGPS04*nc20*b* and *SGPS04nc20c*; and when the pieces have a high fitting degree, such as in the instances AÖAB19*nc4* and AÖAB19*nc7,* which form a Soma cube. This methodology's performance seems to be affected due to the roughness of the movements that led to fast collisions and caused the pieces to separate. Furthermore, these instances required a high-level fit which is hard to meet with rough movements.

6. Conclusions and future work

Two simulation-based algorithms were presented to tackle the *polyhedra packing problem*. The first algorithm considered all simulated movements, and the second algorithm was the hybridization of the first algorithm with the *probabilistic tabu search*. These algorithms included basic simulation movements, which can be used as a starting point for developing more sophisticated algorithms that better exploit the structure of C&PP.

Results showed that the hybridization led to significantly better results in 5 of the 16 instances than the algorithm without hybridization; in the other 11 instances, the two algorithms were not significantly different. This situation could indicate that hybridization with metaheuristics improves the performance of simulation-based algorithms.

The simulation-based algorithms were used to obtain satisfactory results in a short time. These algorithms were fast, and the results were competitive when compared with other methodologies. In some instances, the gap was substantial; however, it should be noted that the maximum running time for the algorithms was short. The speed of this algorithm makes it suitable to hybridize with other more sophisticated approaches.

Results showed that simulation is a suitable tool to verify containment and non-overlapping constraints due to the low overlap values. Overlap values for all instances were under 0.2%, demonstrating the capacity of PhysX to efficiently calculate the physics of the objects within the simulation.

The intertwining metric was proposed in cases where pieces had a hole. This metric shows the percentage of pieces which are interlacing. This metric can be used as a starting point to create constraints that can be considered in future studies to obtain more realistic packing patterns.

As future work, we propose considering different shaped containers such as cylinders, spheres, or even more complicated shapes. This would lead to more applications in the industry. Also, we propose to hybridize simulations with other metaheuristics or with mathematical programming to get better solutions.

Acknowledgments

The authors thank the support of Ministerio de Ciencia, Tecnología e Innovación (MinCiencias) and the Department of Industrial Engineering of the Universidad de los Andes.

References

Allen, S. D., Burke, E. K., & Kendall, G. (2011). A hybrid placement strategy for the three-dimensional strip packing problem. *European Journal of Operational Research*, *209*(3), 219-227.

Alvarez-Valdes, R., Martinez, A., & Tamarit, J. M. (2013). A branch & bound algorithm for cutting and packing irregularly shaped pieces. *International Journal of Production Economics*, *145*(2), 463-477.

Alvarez-Valdés, R., Parreño, F., & Tamarit, J. M. (2008). Reactive GRASP for the strip-packing problem. *Computers & Operations Research*, *35*(4), 1065-1083.

Araújo, L. J., Özcan, E., Atkin, J. A., & Baumers, M. (2019). Analysis of irregular three-dimensional packing problems in additive manufacturing: a new taxonomy and dataset. *International Journal of Production Research*, *57*(18), 5920-5934.

Araújo, L. J., Panesar, A., Özcan, E., Atkin, J., Baumers, M., & Ashcroft, I. (2020). An experimental analysis of deepest bottom-left-fill packing methods for additive manufacturing. *International Journal of Production Research*, *58*(22), 6917-6933.

Barber, C. B., Dobkin, D. P., & Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, *22*(4), 469-483.

Bennell, J. A., & Oliveira, J. F. (2008). The geometry of nesting problems: A tutorial. *European Journal of Operational Research*, *184*(2), 397-415.

Bortfeldt, A. (2006). A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research*, *172*(3), 814-837.

Bortfeldt, A., & Mack, D. (2007). A heuristic for the three-dimensional strip packing problem. *European Journal of Operational Research*, *183*(3), 1267-1279.

Chazelle, B., Edelsbrunner, H., & Guibas, L. J. (1989). The complexity of cutting complexes. *Discrete & Computational Geometry*, *4*(2), 139-181.

Chernov, N., Stoyan, Y., & Romanova, T. (2010). Mathematical model and efficient algorithms for object packing problem. *Computational Geometry*, *43*(5), 535-553.

Cherri, L. H., Mundim, L. R., Andretta, M., Toledo, F. M., Oliveira, J. F., & Carravilla, M. A. (2016). Robust mixed-integer linear programming models for the irregular strip packing problem. *European Journal of Operational Research*, *253*(3), 570-583.

Chica, M., Juan Pérez, A. A., Cordon, O., & Kelton, D. (2017). Why simheuristics? Benefits, limitations, and best practices when combining metaheuristics with simulation. *Benefits, Limitations, and Best Practices When Combining Metaheuristics with Simulation (January 1, 2017)*. doi: 10.2139/ssrn.2919208.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, *44*(2), 145-159.

Egeblad, J., Garavelli, C., Lisi, S., & Pisinger, D. (2010). Heuristics for container loading of furniture. *European Journal of Operational Research*, *200*(3), 881-892.

Egeblad, J., Nielsen, B. K., & Brazil, M. (2009). Translational packing of arbitrary polytopes. *Computational Geometry*, *42*(4), 269-288.

Egeblad, J., Nielsen, B. K., & Odgaard, A. (2007). Fast neighborhood search for two-and three-dimensional nesting problems. *European Journal of Operational Research*, *183*(3), 1249-1266.

Fischer, K., Gärtner, B., & Kutz, M. (2003, September). Fast smallest-enclosing-ball computation in high dimensions. In *European Symposium on Algorithms* (pp. 630-641). Springer, Berlin, Heidelberg.

Gebhardt, A., & Hötter, J. S. (2016). *Additive manufacturing: 3D printing for prototyping and manufacturing*. Carl Hanser Verlag GmbH Co KG.

Gendreau, M., & Potvin, J. Y. (Eds.). (2010). *Handbook of metaheuristics* (Vol. 2, p. 9). New York: Springer.

Gogate, A. S., & Pande, S. S. (2008). Intelligent layout planning for rapid prototyping. *International Journal of Production Research*, *46*(20), 5607-5631.

Gomes, A. M., & Oliveira, J. F. (2006). Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research*, *171*(3), 811-829.

Hague*, R., Mansour, S., & Saleh, N. (2004). Material and design considerations for rapid manufacturing. *International Journal of Production Research*, *42*(22), 4691-4708.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

Leung, S. C., Lin, Y., & Zhang, D. (2012). Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem. *Computers & Operations Research*, *39*(3), 678-686.

Liu, X., Liu, J. M., & Cao, A. X. (2015). HAPE3D—a new constructive algorithm for the 3D irregular packing problem. *Frontiers of Information Technology & Electronic Engineering*, *16*(5), 380-390.

Liu, X., & Ye, J. W. (2011). Heuristic algorithm based on the principle of minimum total potential energy (HAPE): a new algorithm for nesting problems. *Journal of Zhejiang University-SCIENCE A*, *12*(11), 860-872.

Lucas, T. W., Kelton, W. D., Sanchez, P. J., Sanchez, S. M., & Anderson, B. L. (2015). Changing the paradigm: Simulation, now a method of first resort. *Naval Research Logistics (NRL)*, *62*(4), 293-303.

Ma, Y., Chen, Z., Hu, W., & Wang, W. (2018, August). Packing irregular objects in 3D space via hybrid optimization. In *Computer Graphics Forum* (Vol. 37, No. 5, pp. 49-59).

Martello, S., Monaci, M., & Vigo, D. (2003). An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, *15*(3), 310-319.

Martínez, D. A., Alvarez-Valdes, R., & Parreño, F. (2015). A grasp algorithm for the container loading problem with multi-drop constraints. *Pesquisa Operacional*, *35*, 1-24.

Martínez-Franco, J. C., & Álvarez-Martínez, D. (2018, December). Physx as a middleware for dynamic simulations in the container loading problem. In *2018 Winter Simulation Conference (WSC)* (pp. 2933-2940). IEEE.

Martínez, J. C., Cuellar, D., & Álvarez-Martínez, D. (2018). Review of Dynamic Stability Metrics and a Mechanical Model Integrated with Open Source Tools for the Container Loading Problem. *Electronic Notes in Discrete Mathematics*, *69*, 325-332.

Martínez, J. C., Cuellar, D., Céspedes, E., & Martínez, D. (2018). Packagecargo-open source tool based on optimization and simulation for the container loading problem with dynamic stability. In *2018 International Conference on Industrial Engineering and Operations Management* (pp. 2363-2370).

Milenkovic, V. J., & Daniels, K. (1999). Translational polygon containment and minimal enclosure using mathematical programming. *International Transactions in Operational Research*, *6*(5), 525-554.

Romanova, T., Bennell, J., Stoyan, Y., & Pankratov, A. (2018). Packing of concave polyhedra with continuous rotations using nonlinear optimisation. *European Journal of Operational Research*, *268*(1), 37-53.

Stoyan, Y., Pankratov, A., & Romanova, T. (2016). Quasi-phi-functions and optimal packing of ellipses. *Journal of Global Optimization*, *65*(2), 283-307.

Stoyan, Y., Gil, M., Romanova, T., Ternoy, J., & Scheithauer, G. (2002). Construction of a-function for two convex polytopes. *Applicationes Mathematicae*, *29*(2), 199-218.

Stoyan, Y. G., Gil, M., Pankratov, A., & Scheithauer, G. (2004). Packing non-convex polytopes into a parallelepiped. *Preprint MATH-NM-06-2004: Technische Universität of Dresden*.

Stoyan, Y. G., Gil, N. I., Scheithauer, G., Pankratov, A., & Magdalina, I. (2005). Packing of convex polytopes into a parallelepiped. *Optimization*, *54*(2), 215-235.

Talbi, E. G. (2009). *Metaheuristics: from design to implementation* (Vol. 74). John Wiley & Sons.

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European journal of operational research*, *183*(3), 1109-1130.

Wei, L., Oon, W. C., Zhu, W., & Lim, A. (2012). A reference length approach for the 3D strip packing problem. *European Journal of Operational Research*, *220*(1), 37-47.

Wong, K.V., & Hernandez, A. (2012). A review of additive manufacturing. *ISRN Mechanical Engineering, 1*.

Zhang, C., & Chen, T. (2001). Efficient feature extraction for 2d/3d objects in mesh representation. *In Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205),* Vol. 3, IEEE, pp. 935–938.