

Width Beam and Hill-Climbing Strategies for the Three-Dimensional Sphere Packing Problem

Mhand Hifi* and Labib Yousef

EPRAOD EA 4669, Université de Picardie Jules Verne

7 rue du Moulin Neuf, 80000 Amiens, France.

Emails: {mhand.hifi, labib.yousef}@u-picardie.fr

*Corresponding author.

Abstract—In this paper we propose to enhance a width-beam search in order to solve the three-dimensional sphere packing problem. The goal of the problem is to determine the minimum length of the container having fixed width and height, that packs n predefined unequal spheres. The width-beam search uses a greedy selection phase which determines a subset of eligible positions for packing the predefined items in the target object and selects a subset of nodes for exploring some promising paths. We propose to handle lower bounds in the tree and apply a hill-climbing strategy in order to diversify the search process. The performance of the proposed method is evaluated on benchmark instances taken from the literature. The obtained results are compared to those reached by some recent methods available in the literature. Encouraging results have been obtained.

Index Terms—Beam; Heuristic; Hill-Climbing; Optimization; Packing.

I. INTRODUCTION

THIS paper deals with the *Three-Dimensional Sphere Packing Problem* (noted 3DSPP), where an instance of such a problem is defined by a set I of n unequal items and an object \mathcal{P} having fixed width W and height H and, unlimited length (another representation for similar problems can be found in Wascher *et al.* [20]). In this case, each item $i \in I = \{1, \dots, n\}$ is characterized by its radius r_i and the goal of the problem is to minimize the length, denoted L , of the object \mathcal{P} such that all items of I are packed in the target object, without overlapping. The 3DSPP may be formulated as follows:

$$\min L \quad (1)$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \geq (r_i + r_j)^2 \quad (2)$$

$$\forall (i, j) \in I \times I, i < j$$

$$r_i \leq x_i \leq L - r_i, \forall i \in I \quad (3)$$

$$r_i \leq y_i \leq H - r_i, \forall i \in I \quad (4)$$

$$r_i \leq z_i \leq W - r_i, \forall i \in I \quad (5)$$

$$\underline{L} \leq L \leq \bar{L} \quad (6)$$

$$(x_i, y_i, z_i) \in \mathbb{R}_+^3, \forall i \in I \quad (7)$$

where the objective function (1) minimizes the length of the object \mathcal{P} containing all the items of I , Eq. (2) ensures the non-overlap constraint of any pair of distinct items (i, j) of $I \times I$; that is, the distance between the centers of both items which must be greater than or equal to the sum of their radii and Eqs (3)-(5) ensure that all items of I belong to the

target object \mathcal{P} of dimensions (L, W, H) and Eq. (7) ensures that all items are placed in the object \mathcal{P} . Also, it is easy to start any approach by a trivial value representing the sum of the spheres' area affected to \underline{L} (Eq. (6)) and a feasible solution value, that can be obtained by applying a simple greedy procedure, affected to \bar{L} .

In this paper, we propose new strategies in order to enhance a width-beam search based algorithm already proposed in Hifi and Yousef [8]. First, we introduce a new greedy strategy in order to generate some eligible nodes. Second, we propose to curtail the search process by estimating a global lower bound. Third and last, a hill-climbing strategy is used in order to correct the global lower bound and to select some nodes realizing highest potentials able to reach better solutions.

The remainder of the paper is organized as follows. Section II gives a literature review for the 3DSPP and some of its variants. The problem representation is discussed in Section III-A. The greedy selection phase, which serves to determining a subset of eligible positions for predefined items to pack, is detailed in Section III-B. A modified version of the algorithm is described in Section III-D, where a lower bound is used for exploring better paths that diversify the search. Moreover, because of the huge number of feasible positions that a predefined item may generate, both beam width and hill-climbing strategies cooperate for selecting the best promising nodes at the same level of the developed tree. Section IV evaluates the performance of the proposed algorithm and compares its produced results to those reached by the original width-beam search and recent methods available in the literature. Finally, Section V concludes by summarizing the contribution of this paper.

II. RELATED WORKS

The 3DSPP belongs to the well-known family of Cutting and Packing (CP) that represents a natural combinatorial optimization problems. Problems of CP family admit numerous real-world applications in the domain of industrial engineering, logistics, manufacturing, production process, automated planning, etc. One of the more recent paper addressing an optimization with a packing problem is due to Sutou and Dai [18], where the unequal sphere problem has been used for tackling an application of the automated radio-surgical treatment planning. Wang [19] has also considered sphere packing problems as an optimization tool for the radio-surgical

treatment planning. Other problems of the CP family have been described and redefined in Wascher *et al.* [20] where an instance of these problems can be defined by a set of predetermined items to be packed in one or many larger containers (objects) so as to minimize the unused area / space or in some cases to maximize a utility function. Furthermore, the items are bounded by their dimensions (rectangular, circular, or irregular) and the objects can be bounded (rectangular, circular, ...) or unbounded (strips / parallelepipeds, ...).

Among existing papers addressing sphere packing problems, Lochmann *et al.* [12] proposed a statistical analysis for packing random spheres with variable radius distribution. Li and Ji [11] discussed a dynamics-based collective method for random sphere packing and they tried to apply it to the problem of packing sphere into a cylinder container. In this paper, the stability of the method and its convergence were tackled. Farr [3] studied the problem of random close packing fractions of log-normal distributions of hard spheres. The author tailored a one-directional approach in order to predict a close packing of spheres of log-normal distributions of sphere sizes.

Packing spheres into a container has been addressed by Sutou and Dai [18] who proposed a global optimization approach. Stoyan *et al.* [17] designed a mathematical model in order to pack spheres into an open container, where both height and widths are fixed whereas the length is unfixed. They use a neighboring search based upon extremum points in order to construct and improve a series of solutions. M'Hallah *et al.* [13] proposed a heuristic based on combining VNS with nonlinear programming solver. The method iterates some moves of the current configuration and complete the partial configuration with a solver dedicated for nonlinear programmes. M'Hallah and Alkandari [14] considered the principle used in [13] to solve the problem of packing identical spheres into the smallest containing sphere. Soontrapa and Chen [16] tackled the problem of packing identical spheres into a smallest containing sphere by using a random search according to Monte Carlo's method. Birgin and Sobral [1] proposed twice-differentiable non-linear programming models for the problem of packing both circles and spheres into different containers where the containers may be circular, rectangular, etc. In order to find a global solution for their proposed models, ALGENCAN solver was used for generating a multiple starts solutions. Finally, Hifi and Yousef [8] investigated the use of a dichotomous search for solving the three-dimensional sphere packing problem (an extensive efficient models and methods for packing both circular and sphere problems were reviewed in Hifi and M'Hallah [4]).

In this paper, we propose to enhance the algorithm proposed in Hifi and Yousef [8] by considering three modifications: (i) considering a modified greedy strategy in order to generate eligible nodes, (ii) an estimation of the global lower bound for curtailing the search process and, (iii) the hill-climbing used for correcting the global lower bound and selecting only some nodes with highest potentials.

III. A WIDTH-BEAM SEARCH FOR 3DSPP

In this part, the problem representation and strategies used are first described in Section III-A. Second, Section III-B describes the greedy procedure in order to build feasible packings containing all items of I . Third, Section III-C discusses the principle of the proposed algorithm and its main steps. Fourth and last, Section III-D presents the modifications used for enhancing the algorithm.

A. Representation of the problem

The local strategy is based on the simple Greedy Principle (called GP) where the minimum distance position is favored for packing a series of predefined items. GP is then used as the first evaluation operator for finding a subset of possible positions of the next item to pack. Such a procedure uses the following notations:

- The bottom-left-depth corner of \mathcal{P} is positioned at $(0, 0, 0)$ and \mathcal{P} is characterized by a set formed with six labels (namely faces): $\mathbb{F} = \{\text{left, top, right, bottom, depth, front}\}$. Then, \mathcal{P} is represented in the Euclidean space, as illustrated in Figure 1.
- The center of the i -th item belonging to I is positioned at (x_i, y_i, z_i) .
- The distance $\delta_{i,j}$ between two items i and j is computed as follows: $\forall (i, j) \in I^2$,

$$\delta_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} - (r_i + r_j).$$

Note that assigning an item $i \in I$ to a possible position of \mathcal{P} while respecting non-overlapping between this item and the left-face of \mathcal{P} requires to satisfy the following distance: $\delta_{i,\text{left}} = x_i - r_i$. For more general cases, Table I reports the distance to be satisfied whenever an item i is assigned to a selected position of \mathcal{P} .

TABLE I
THE DISTANCE BETWEEN AN ITEM i AND A FACE f

f	$\delta_{i,f} \mid i \in I, f \in \mathbb{F}$
left	$x_i - r_i$
bottom	$y_i - r_i$
depth	$z_i - r_i$
right	$L - x_i - r_i$
top	$H - y_i - r_i$
front	$W - z_i - r_i$

B. Defining eligible positions

It is well-known that tailored heuristics are mainly based on the strategies which are able to guide well the search process. These strategies may be use some selection criteria in order to provide either partial or final solutions for the problem to solve. Herein, we consider a simple greedy principle (GP) which is based on searching the position realizing the minimum distance position between items and faces. In fact, GP is used as a selection criterion for defining a set of eligible positions to assign to the predefined item i (not

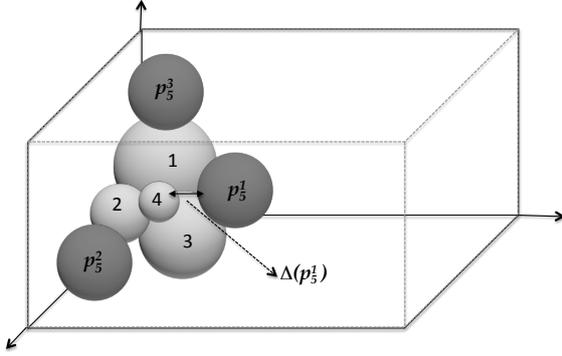


Fig. 1. Illustration of the mechanism used for computing eligible positions considered by GP.

already positioned) among all eligible positions representing the set P_{I_i} .

In what follows, we assume that the (center of the) first item $i = 1$ of I is positioned at the position (r_1, r_1, r_1) and $\forall i \in I, i \geq 2$, the following notations are considered:

- I_i denotes the set of items of I already positioned in the current object \mathcal{P} .
- \bar{I}_i contains the items of I which are not yet assigned to \mathcal{P} .
- P_{I_i} denotes the set of distinct eligible positions for the next item i to pack given the set of packed items I_i .
- An eligible position $p_{i+1} \in P_{I_i}$ (for the item i) is determined by using three elements e_1, e_2 and e_3 where an element is either an item of I already positioned (representing the set I_i) or one of the six faces belonging to \mathbb{F} .
- $\mathcal{T}_{p_{i+1}}$ represents the set composed of the three elements e_1, e_2 and e_3 .

Figure 1 illustrates GP's mechanism on a small example. For this example, assume that the first four items are already positioned in the object \mathcal{P} ; then, there are three eligible positions that emerge for the next item 5 to pack. Following the above notations, $I_4 = \{1, 2, 3, 4\}$ and $P_{I_4} = \{p_5^j, j = 1, \dots, 3\}$. First, the position p_5^1 touches both items 1 and 3 and, the "bottom" face of \mathcal{P} . Second, the position p_5^2 is obtained by using the item 2 and both faces "left" and "bottom" of \mathcal{P} . Third and last, the position p_5^3 is computed by using the item 1 and both faces "left" and "depth" of \mathcal{P} . Finally, it follows that $\mathcal{T}_{p_5^1} = \{1, 2, \text{bottom}\}$, $\mathcal{T}_{p_5^2} = \{2, \text{left}, \text{bottom}\}$ and $\mathcal{T}_{p_5^3} = \{1, \text{left}, \text{depth}\}$. Further, item 4 is positioned around the three items 1, 2 and 3 that means $\mathcal{T}_4 = \{1, 2, 3\}$. For the next item 5 to pack, its coordinates are computed by using both items 1 and 3 and one of the faces (the "bottom" in this case) of \mathcal{P} that gives $\mathcal{T}_5 = \{1, 3, \text{bottom}\}$. We recall that the objective of the problem is to minimize the length of the target object \mathcal{P} . It means that the right face of \mathcal{P} can be omitted and

only the five faces can be considered when optimizing the length of the target \mathcal{P} . Hence, all eligible positions may be obtained by using the fifth faces, the positioned items and the current item to pack.

Then, the value corresponding to the $(i + 1)$ -th item to pack, when positioned at the eligible position $p_{i+1}^k \in P_{I_i}$ by GP, is computed as follows

$$\Delta(p_{i+1}^k) = \min_{j \in I_i \cup \mathbb{F}''} \delta_{(i+1,j)} \quad (1)$$

where $\mathbb{F}' = \mathbb{F} \setminus \{\text{right}\}$ and $\mathbb{F}'' = \mathbb{F}' \setminus \mathcal{T}_{p_{i+1}^k}$.

Finally, when GP is used, it starts by positioning the first item $i = 1$ at the bottom-left-depth position, i.e., at the position (r_1, r_1, r_1) , while the remaining $n - 1$ items are successively positioned according to the minimum distance rule (cf., Eq. (1)). As illustrated in Figure 1, the item 5 will be placed at position p_5^1 because its corresponding distance realizes the minimum value.

C. A width-beam search heuristic for the 3DSPP

Beam Search (BS) has been first proposed in [15] for tackling the scheduling problem and it has since been successfully applied to many other combinatorial optimization problems (some adaptations can be found in Della Croce *et al.* [2], Hifi *et al.* [5], [6], [7] and, Yavuz [21]). Such an approach can be viewed as a truncated tree search procedure where its objective is to avoid exhaustive search by performing a partial enumeration of the solution space.

1) *Packing all items on the target object:* At each level of the developed tree, only a subset of nodes (called the *set of elite nodes*) are selected for further branching and the other nodes are discarded, where no backtracking is performed. For each level, the cardinality of the elite nodes to be investigated is fixed to ω that is called the *beam width*. Generally, these selected ω nodes represent those having a high potential to lead the best solutions for the treated problem. Furthermore, each node is assessed via an *evaluation function* whose role is to provide a promising separation mechanism of the nodes of each level of the developed tree.

As observed in Hifi and Yousef [8], applying BS to 3DSPP required to define the nodes of the tree and the branching mechanism out of the nodes of B . Herein, a node η_i is represented by the pair of subsets:

- 1) The first subset I_i containing all items assigned to the target object \mathcal{P} and,
- 2) The complementary subset \bar{I}_i containing the unassigned items.

Moreover, branching out of a node η_i is equivalent to create at most $|P_{I_i}|$ branches emanating out of the current node (related to the eligible positions as described in Section III-B). Each resulting node corresponds to packing the subset of items I_i and assigning to the current item i a favorite eligible position. Moreover, each of these created nodes will be represented by a pair of two complementary subsets of items of I . Further, in order to explore a reasonable number of nodes, a width-beam search almost of the standard beam search has been

used in Hifi and Yousef [8]. Therefore, all nodes emanating from the same level are simultaneously evaluated following an estimator operator and only the best ones are selected for the rest of the search.

Such a process is described by the main steps of Algorithm 1, where it works according to a given node, namely η^ℓ . This node is the one taken at the level ℓ of the developed tree. Thereafter, the initialization step is applied for starting the set B containing the best provided nodes regarding the starting node η^ℓ (lines 1 to 3), the initialization of the variable `feasible` to false (line 4) and, fixing the runtime limit t_{\max} (line 5) for which the algorithm stops when that time is performed (in this case, the control parameter t_{iter} , for the limit t_{\max} , is initialized to zero). Note that the variable `feasible` is used for controlling the (un)feasibility of the series of the solutions build. The main loop (line 7) starts by choosing the best eligible positions for each node belonging to B . These positions are computed by using GP's selection (cf., Section III-B). Second, all created nodes are stored in a provisional set B_ω where the potential of each of these nodes are evaluated according to the final solution provided by iteratively applying GP as a heuristic (cf. as discussed in the last paragraph of Section III-B). Thereafter, for each final solution (either feasible or unfeasible for the target object \mathcal{P}), the potential of a node $\eta \in B_\omega$ is represented by the *density of the positioned items* in \mathcal{P} . Whenever one of these constructed solutions provides a feasible solution (line 11), i.e., all items are positioned in the target object \mathcal{P} , then the algorithm stops with a feasible solution (i.e., setting the variable `feasible` to true). Otherwise, the set B of the best nodes is updated (line 12) by the ω nodes which realizing the highest densities and the current level of the developed tree is incremented. The internal runtime t_{iter} is then incremented and the process is iterated until either B is reduced to an empty set or when

Algorithm 1 . Beam Search for the 3DSPP: BS

Input. A node η^ℓ .

Output. `feasible` // setting equal to `true` whenever a feasible packing is reached, `false` otherwise

1: **Initialization Step.**

2: Let ω be a predefined beam width.

3: Set $B = \{\eta^\ell\}$, where η^ℓ denotes the input node associated to the ℓ -th level.

4: Set the variable `feasible` to `false` /* no feasible solution at hand */

5: Let t_{\max} be a maximum fixed runtime and t_{iter} (initialized to 0) be a counter which serves to control the time spent for exploration the space search.

6: **Iterative Step.**

7: **while** $((B \neq \emptyset)$ and $(t_{\text{iter}} < t_{\max}))$ **do**

8: Branch from the current level ℓ by selecting the ω eligible positions for each node $\eta_{\ell_i} \in B$;

9: Insert all obtained nodes into B_ω ;

10: Evaluate the potential of each node belonging to B_ω using GP for completing the path.

11: If a feasible solution is given by GP, then set `feasible` to `true` and **exit**;

12: Replace B by the best ω nodes of B_ω realizing highest densities and, increment the level ℓ .

13: Update the current runtime t_{iter} .

14: **end while**

t_{\max} , the limited runtime, is performed, i.e., $t_{\text{iter}} \geq t_{\max}$.

Algorithm 2 . A Dichotomous Search Based Heuristic: DSBH

Input. An instance of 3DSPP.

Output. An object \mathcal{P} of dimensions (L_{best}, W, H) and the coordinates of all items of I .

1: **Initialization step**

2: Call an iterative GP on the open strip (∞, W, H) and let \bar{L} be the starting length reached.

3: Set $\underline{L} \leftarrow \frac{4\pi}{3 \times W \times H} \sum_{i \in N} (r_i^3)$ and $L^* = \bar{L}$.

4: Set ω to a predefined minimum value.

5: **Iterative step**

6: **while** (the runtime limit is not performed) **do**

7: **repeat**

8: $L^* = (\bar{L} + \underline{L})/2$

9: Generate the starting node η^1 with its three sets I_i, \bar{I}_i and P_I^1 .

10: Set `feasible` \leftarrow BS(η^1), where BS is called with (L^*, W, H)

11: **If** (`feasible=true`) **then** set $\bar{L} = L^*$; $\underline{L} = L^*$ otherwise

12: **until** $(\bar{L} - \underline{L} \geq \alpha)$

13: Set $\underline{L} \leftarrow \frac{4\pi}{3 \times W \times H} \sum_{i \in N} (r_i^3)$ and increment ω .

14: **end while**

2) *Using a dichotomous search:* Because Algorithm 1 is applied on the target container \mathcal{P} , then one can repeat the same principle on a series of target containers $\mathcal{P}_1, \dots, \mathcal{P}_r$, where $r \geq 1$. Indeed, one can starts the search with the initial interval $[\underline{L}, \bar{L}]$, where \underline{L} denotes a lower bound for the 3DSPP and \bar{L} its upper bound (in the case of a feasible solution exists, its objective value is assigned to \bar{L}). Then, for each fixed interval, Algorithm 1 tries to construct the best feasible solution by packing all the items into the current target object; that is, (L^*, W, H) , where $L^* \in [\underline{L}, \bar{L}]$.

The main steps of the dichotomous search are summarized in Algorithm 2. First, it starts by defining the initial interval $[\underline{L}, \bar{L}]$ where the upper bound \bar{L} is obtained by applying GP as a heuristic on the open object, i.e., (∞, W, H) . The main loop “repeat ... until” (cf., lines 7 - 12) of the dichotomous procedure serves to explore a series of neighborhoods depending on the values of ω . At line 8, a new target upper bound is computed, namely $L^* = (\bar{L} + \underline{L})/2$. Line 9 generates the initial node positioned at the bottom-left-depth corner (in the position (r_1, r_1, r_1)) and creates its corresponding sets I_i, \bar{I}_i and P_I^1 (as discussed in Section III-B). At line 10, BS is called with the target value of the object (L^*, W, H) and the created sets reached at the next step. Line 11 serves to update the interval search where its upper bound is updated whenever a feasible solution is obtained, the lower bound is updated otherwise. Thereafter, the process is iterated until the gap between both lower and upper bounds becomes closest to a certain tolerance, namely α . Finally, the aforementioned process is iterated a certain number of times following the values of ω (line 13) and according to the runtime limit fixed.

D. A modified version of the width-beam search for 3DSPP

We first describe the modified GP that tries to generate some interesting eligible positions. Second and last, we introduce the lower bound in order to curtail the search; this upper bound cooperates with both hill-climbing strategy and beam width strategies in order to select future nodes for branching.

1) *A modified GP*: The Modified GP (MGP) provides other eligible positions able to homogeneously concentrate a subset of items on the target object. Return now to Figure 1 and observe the candidate positions of item p_5 : any eligible position induces a packing concentrated on the bottom-left-depth position. By applying this principle to the instance SYS1 (one of the instances tested in Section IV), one can observe that all packed items are focused on the starting position that leaves the other parts of the object sufficiently empty.

Herein, we first propose to modify such a placement by adding three corner positions whenever all eligible positions for a selected item to pack; that are $\mathcal{T}_{p_2^4} = \{\text{top, depth, left}\}$, $\mathcal{T}_{p_2^5} = \{\text{top, left, front}\}$ and $\mathcal{T}_{p_2^6} = \{\text{bottom, left, front}\}$.

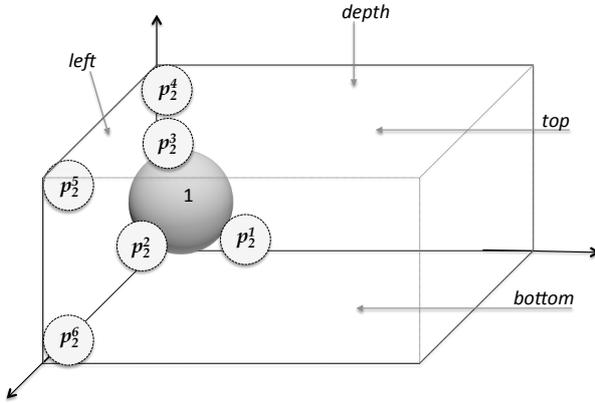


Fig. 2. Illustration of MGP's eligible positions.

Figure 2, where we assume that the first item is already positioned in the object \mathcal{P} , shows six eligible positions that emerge for the next item 2 to pack. According to the representation described above, $I_1 = \{1\}$ and $P_{I_2} = \{p_2^j, j = 1, \dots, 6\}$. First, the position p_2^1 touches the item 1 and both faces “depth” and “bottom” of \mathcal{P} . Second, the position p_2^2 is obtained by using the item 1 and both faces “left” and “bottom” of \mathcal{P} . Third, the position p_2^3 is computed by using the item 1 and both faces “left” and “depth” of \mathcal{P} . Finally, all other positions touch three faces of \mathcal{P} . It follows that $\mathcal{T}_{p_2^1} = \{1, \text{depth, bottom}\}$, $\mathcal{T}_{p_2^2} = \{1, \text{left, bottom}\}$, $\mathcal{T}_{p_2^3} = \{1, \text{left, depth}\}$, $\mathcal{T}_{p_2^4} = \{\text{top, left, depth}\}$, $\mathcal{T}_{p_2^5} = \{\text{top, left, front}\}$ and $\mathcal{T}_{p_2^6} = \{\text{bottom, left, front}\}$.

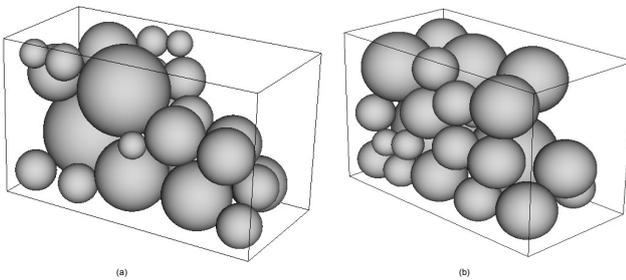


Fig. 3. Illustration of (a) GP's behavior and (b) its modified version MGP's on SYS1 instance.

On the one hand, as illustrated in Figure 2, the item 2 can

be packed at one of the six positions p_2^j , $j = 1, \dots, 6$, and if the smallest distance to the left side of \mathcal{P} is favored, then five positions remains favorable (cf., Figure 3.(a)). In this case GP provides an object of length equal to 11.51. On the other hand, If such a principle is applied, from Figure 3.(b) and for the same instance, one can observe the improvement made on the packing which at the same time produces a better length (10.49 in this case) for the target object.

E. Using the hill-climbing strategy

Hill-Climbing (HC) can be used as a curtailing strategy for avoiding exhaustive search. In this case, the search process may perform a partial enumeration of the solution space, where in term of tree, only a subset of paths are taken for further branchings and the other nodes are discarded. Further, each selected node is assessed via its evaluation function whose role is to provide a promising separation mechanism of the nodes. In our study, we introduce the HC strategy that is used for avoiding exhaustive search that is equivalent to an augmented beam search (Hifi *et al.* [7] and Yavuz [21])), where a subset of paths are taken for further branchings and the other nodes are discarded. At each step of the search procedure, a node η is selected and after evaluating all its successors, only the best ω nodes are chosen for further branchings. Each selected node is assessed via its evaluation function whose role is to provide a promising separation mechanism of the nodes.

Algorithm 3 describes a modified BS (noted MBS) where MBS replaces BS in the dichotomous search DSBH (cf., Algorithm 2, which also noted MDSBH for the modified DSBH). We recall that a node corresponds to a partial solution at level $\ell \leq n - 1$ and the set B of current nodes contains initially the starting nodes of the root node B_0 , whereas B_ω containing the offspring nodes is initialized to the empty set.

Algorithm 3 . A Modified BS (MBS)

Input. A set of items I and a predefined length l_{best} .
Output. *feasible*..

- 1: **Initialization Step.**
- 2: Let ω and ϵ be two predefined values.
- 3: Set $B = B_0$, where B_0 denotes the starting eligible nodes according to the first packed item $i = 1$.
- 4: Set the level $\ell = 1$ and $B_\ell = \emptyset$.
- 5: Set the variable *feasible* to *false* /* no feasible solution at hand */
- 6: **Iterative Step.**
- 7: **while** $(B \neq \emptyset)$ and (the runtime limit is not performed) and $(\ell < n)$ **do**
- 8: **for each** $\eta \in B$ **do**
- 9: Let $B_\eta = \{\gamma_1, \dots, \gamma_{|P_{I_\eta}|}\}$ be the successors of η .
- 10: Evaluate the potential of each node γ belonging to B_η by computing $g(\gamma)$ and $h'(\gamma)$.
- 11: For each $\gamma \in B_\eta$ apply ISBH(γ, L^*) and update L^* if necessary with the incumbent solution.
- 12: Set $B_\ell = B_\ell \cup B_\eta$;
- 13: **end for**
- 14: Filter B_ℓ by keeping the ω best nodes realizing the smallest values of $L^*/(g(\gamma) + h'(\gamma))$.
- 15: Replace all the nodes of B by those of B_ℓ , increment ℓ and set $B_\ell = \emptyset$.
- 16: **end while**

On the one hand, a selected node η taken from B (step 7), whose evaluation is z_η , creates a subset of nodes $B_\eta =$

$\{\gamma_1, \dots, \gamma_{|P_{I_\eta}|}\}$, where each resulting node is evaluated according to its *cost operator*; that is,

$$z_\eta = g(\eta) + h(\eta).$$

On the other hand, because $|P_{I_\eta}|$ is large, only some nodes are chosen for further branchings. Indeed (line 9), if a node γ of B_η packs at most $n-1$ items, then it remains in B_η when $z'(\gamma) < z^*$, such that

$$z'(\gamma) = g(\eta) + h'(\eta) \quad (2)$$

where $h'(\eta) = (1 + \epsilon)h(\eta)$ and ϵ is considered as a small predefined value that is used for making a correction on the complementary lower bound $h(\eta)$. Whenever Eq. (2) is not satisfied, then γ is removed from B_η .

Further, since we try to diversify the search that allows for exploring new solutions, we apply BGP on all selected nodes (line 10). Then, L^* is updated whenever BGP produces a better length; in this case, its corresponding incumbent solution is also updated. The rest of the nodes belonging to B_η (line 11) are reordered in nondecreasing order of their estimated lower bound $z'(\gamma)$ and only the best ω nodes are selected and becomes the new nodes of B for further branchings. This process is iterated until no further branching is possible, i.e., until $B = \emptyset$, or the last level is equal to n , or when the fixed runtime limit is performed. Note also that, at lines 9 and 10, if a node γ of B_η is a leaf (i.e, no further branching is possible out of γ), then its objective function value z_γ is computed and compared to z^* . If $z_\gamma < z^*$, then the incumbent solution is set to a leaf node; z^* is then updated: $z^* = z_\gamma$; and γ is removed from B_η .

IV. COMPUTATIONAL RESULTS

In this section we investigate the effectiveness of the modified width beam search-based heuristic (noted MDSBH) on two sets of benchmark instances: Set1 and Set2. The proposed algorithm was coded in C++ and tested on an Intel Core 2 Duo (2.53 Ghz and with 4 Gb of RAM) and the runtime limit was fixed to one hour.

The first set ‘‘Set1’’ contains six instances (SYS1, . . . , SYS6) extracted from Stoyan *et al.* [17], where the number of the predefined items varies from 25 to 60. These instances have been already tested using Stoyan *et al.*’s [17], Birgin and Sobral’s [1] and Kubach *et al.*’s [10] approaches.

The second set ‘‘Set2’’ contains six instances (KBTG1, KBTG2, KBTG3, KBTG7, KBTG8, and KBTG9) taken from Kubach *et al.* [10]. For each instance, both dimensions W and H of the object are fixed to 10 whereas the number of the predefined items is fixed to 30 (resp. 50) for the first (resp. last) three instances. Moreover, these six instances have been already tested in Kubach *et al.* [10] where they represent the six instances with unequal spheres.

A. Performance of MDSBH vs five heuristics: Set1

Generally, when using approximate algorithms to solve optimization problems, it is well-known that different parameter settings for the approach lead to results of variable

quality. As discussed in Section III-D, MDSBH considers three parameters: the beam width ω , the value of ϵ used for correcting the value of the global lower bound and the maximum runtime limit to fix. Our computational study was conducted by varying ω in the discrete interval $\{5, 6, 7, \dots\}$, the maximum runtime limit was fixed to 3600 seconds (which can be considered as a standard runtime limit considered by algorithms of the literature) and ϵ which takes one of the following values: 0.1, 0.2 and 0.3. Of course, the upper value of ω depends on the limited runtime and the size of the instance.

In order to show the effect of these parameters, we first discuss the quality of the solutions obtained by MDSBH when varying the value of ϵ . Table II shows MDSBH’s objective values when varying ϵ from 0.1 to 0.3. From Table II, we observe that MDSBH with $\epsilon = 0.2$ provides better average results since it realizes a value of 9.939 compared to both values 9.957 and 9.961, which corresponds to $\epsilon = 0.1$ and $\epsilon = 0.3$, respectively.

Label	MDSBSs’ solutions when varying ϵ		
	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$
SYS1	9.1946	9.1796	10.9001
SYS2	8.910122	8.8922	8.8922
SYS3	8.6862	8.6702	8.6862
SYS4	10.2154	10.2012	10.2300
SYS5	10.9237	10.8954	10.9222
SYS6	11.8105	11.7943	11.8105
Av.	9.957	9.939	9.961

TABLE II
BEHAVIOR OF MDSBH, WHEN VARYING ϵ , ON THE INSTANCES OF SET1.

Figure 4 illustrates the configurations realized by MDSBH for instance SYS1. Hence, for the rest of the paper, $\epsilon = 0.2$ is chosen for evaluating the performance of MDSBH on all benchmark instances of the literature.

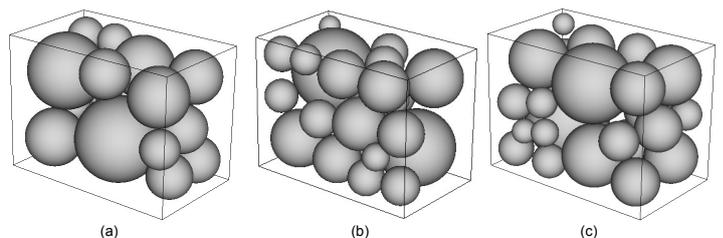


Fig. 4. SYS1 solutions (with MDSBH) when varying ϵ : (a) $\epsilon = 0.1$ with length $L^* = 9.1946$, (b) $\epsilon = 0.2$ with length $L^* = 9.1796$, and, (c) $\epsilon = 0.3$ with length $L^* = 10.9001$.

Second, for the instances of Set1, Table III compares the results of MDSBH to those reached by five algorithms: SYS (Stoyan *et al.* [17]), BSA (Birgin and Sobral [1]), $KBTG_s$ (Kubach *et al.* [10]), its parallel version noted $KBTG_p$ (proposed in Kubach *et al.* [9]), where the known solutions of the literature are taken from Kubach *et al.* [9], [10] and HY (Hifi and Yousef [8]).

Column 1 of Table III shows the instance label, Column 2 displays the objective value L_{SYS}^* realized by STS whereas column 3 displays BSA’s objective value (noted L_{BSA}^*).

#Inst. Label	SYS		BSA		HY		MDSBH	
	L_{SYS}^*	L_{BSA}^*	$L_{KBTG_s}^*$	$L_{KBTG_p}^*$	L_{HY}^*	L_{MDSBH}^*	ω^*	
SYS1	9.912	9.7942	9.2874	9.2656	9.2431	9.1796 \diamond	26	
SYS2	9.623	-	9.1280	8.9301	8.9164	8.8922 \diamond	29	
SYS3	9.473	9.3090	8.9850	8.7178	8.7055	8.6702 \diamond	31	
SYS4	11.086	11.0962	10.8760	10.4042	10.2357	10.2012 \diamond	36	
SYS5	11.646	11.6211	11.3494	10.9865	10.9359	10.8954 \diamond	34	
SYS6	12.842	12.7215	12.3745	11.8399	11.8178	11.7943 \diamond	16	
Av.	10.764	10.908	10.333	10.024	9.976	9.939		

TABLE III

PERFORMANCE OF MDSBH VERSUS THE FIVE HEURISTICS OF THE LITERATURE ON INSTANCES OF SET1. THE SYMBOLE “-” (RESP. “ \diamond ”) MEANS THAT THE VALUE FOR THIS INSTANCE IS NOT AVAILABLE (RESP. CORRESPONDS TO THE BEST SOLUTION).

Columns 4 and 5 report the solutions (noted $L_{KBTG_s}^*$) provided by the sequential KBTG_s algorithm, column 5 reports the best solutions reached by the parallel version of KBTG (noted $L_{KBTG_p}^*$) without fixing the runtime limit and column 6 displays the results reached by HY. Column 7 displays the solution realized by MDSBH (noted L_{MDSBH}^*). Finally, column 8 reports the best value of ω for which its best solution is performed. All results of Table III are summarized in Table IV, where it tallies the percentage improvement (when it happens) yielded by MDSBH when compared to the results reached by the five other algorithms (noted %SYS, %BSA, %KBTG_s, %KBTG_p and %HY according to the heuristics SYS, BSA, KBTG_s, KBTG_p and HY, respectively).

#Inst. Label	MDSBH vs all heuristics (% Improvement)				
	%SYS	%BSA	%KBTG _s	%KBTG _p	%HY
SYS1	7.39	6.28	1.16	0.93	0.69
SYS2	7.59	-	2.58	0.42	0.27
SYS3	8.47	6.86	3.50	0.55	0.41
SYS4	7.98	8.07	6.20	1.95	0.34
SYS5	6.45	6.24	4.00	0.83	0.37
SYS6	8.16	7.29	4.69	0.39	0.20
Av.	7.67	6.95	3.69	0.84	0.38

TABLE IV

PERCENTAGE IMPROVEMENTS BETWEEN ALL TESTED HEURISTICS: MDSBH, HY, SYS, BSA AND BOTH KBTG_s AND KBTG_p ON INSTANCES OF SET1.

The analysis of the results of both Tables III and IV follows:

- 1) First, MDSBH outperforms the five algorithms SYS, BSA, KBTG_s, KBTG_p and HY. Indeed, it is able to reach the best solutions for all instances of Set1.
- 2) Second, when comparing MDSBHs' results to those reached by SYS, one can observe that the percentage of the improvement varies from 6.45% (instance SYS5) to 8.47% (instance SYS3). This percentage improvement remains interesting when comparing MDSBHs' results to those reached by BSA: in this case, such improvement varies from 6.24% (instance SYS5) to 8.07% (instance SYS4).
- 3) Third, the improvement remains positive when comparing MDSBH's results to those provided by the sequential (resp. parallel) KBTG algorithm. Indeed, the improvement when compared to the sequential version varies from 1.16% (instance SYS1) to 6.20% (SYS4) whereas

it varies from 0.39 (instance SYS6) to 1.95% (instance SYS4) when compared to the parallel version.

- 4) Fourth and last, MDSBH realizes better results than those reached by HY; in this case, the percentage improvement varies from 0.20% (instance SYS6) to 0.69% (instance SYS1).

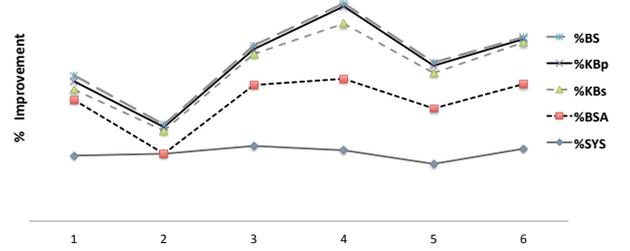


Fig. 5. Variation of the percentage improvement realized by MDSBH when compared to the results of the five heuristics (SYS, BSA, KBTG_s, KBTG_p and HY) on the instances of Set1.

Figure 5 shows the behavior of MDSBH on the instances of Set1 where each curve represents the improvement variation realized according to the algorithm SYS, BSA and both KBTG_s, KBTG_p and HY, respectively.

B. Performance of MDSBH versus KBTG and HY heuristics: Set2

In this section, we compare the results reached by MDSBH to those reached by KBTG_s and HY (note that, for this type of instances, both KBTG_s and HY realize the best objective values of the literature). This comparison is performed on the instances of the second group Set2 taken from Kubach *et al.* [10]. Herein, instead of determining the minimum length L^* of the target container \mathcal{P} , Kubach *et al.* [10] computed the density of all packed items in the final object \mathcal{P} . Therefore, we also report the best length L^* of the final object \mathcal{P} as in Hifi and Yousef [8], since it corresponds to the dual problem that maximizes the density of the occupied area (or equivalently to minimizing the unused area).

#Inst. Label	$d_{KBTG_s}^*$	HY		MDSBH		
		L^*	d^*	L^*	%HY	
KBTG1	54.096	10.9031	56.0092	10.8076	23	0.884
KBTG2	30.071	1.9900	30.071	1.9900	23	0.000
KBTG3	51.387	18.2415	53.6243	18.1936	24	0.263
KBTG7	55.372	13.0997	57.5662	12.9653	14	1.037
KBTG8	45.060	2.5825	47.004	2.5820	13	0.019
KBTG9	52.732	27.8033	55.3203	27.7152	26	0.318
Av.	48.120		49.932			0.420

TABLE V

PERFORMANCE OF BSBH VERSUS KBTG_s ON INSTANCES OF SET2.

The results realized by the three tested methods (MDSBH, KBTG_s and HY, respectively) are reported in Table V. Column 1 displays the instance label, column 2 reports the solution value (expressed in term of density) realized by KBTG_s's algorithm (extracted from Kubach *et al.* [9], [10]), columns 3 and 4 display both HY's length and its density whereas columns 5 and 6 report the best length realized by

MDSBH and the value of ω for which the best solution is reached. Finally, the last column displays the percentage of improvement realized by MDSBH according to the solution values reached by both KBTG_s and HY, respectively. The analysis of the results of Table V follows.

- 1) HY outperforms KBTGs since it provides an average density of 49.932% whereas KBTGs realizes a percentage value of 48.120%.
- 2) MDSBH remains competitive since it improves most solutions reached by both KBTGs and HY. Indeed, it is able to improve five out of six best solutions while it matches the other solution (instance KBTG2) when compared to the results reached by HY.
- 3) For the improved solutions (except for the instance KBTG2 where all algorithms reach the optimal solution), MDSBH realizes an improvement varying from 0.019% (instance KBTG8) to 1.037% (instance KBTG7).
- 4) Globally, the average improvement over all instances is equal to 0.420%, as displayed by the last line of Table V.

V. CONCLUSION

In this paper the three-dimensional sphere packing problem is solved by using a modified dichotomous search-based heuristic. The proposed method is based upon three complementary phases: (i) a modified greedy selection phase which tries to select more eligible positions to iteratively pack all predefined items into the target object, (ii) a width beam search combined with hill-climbing strategies for exploring promising paths and (iii) a dichotomous search for providing a best target object, that is able to pack all items without overlapping. The performance of the modified algorithm was evaluated on benchmark instances available in the literature. The provided results were compared to those reached by the original version of the algorithm, as well as to the results given by some recently proposed heuristics. The new version of the method remains competitive and succeeded in yielding new solutions for many instances.

REFERENCES

- [1] E. G. Birgin and F.N.C. Sobral. Minimizing the object dimensions in circle and sphere packing problems. *Computers & Operations Research*, 35, 2357–2375, 2008 (DOI: 10.1016/j.cor.2006.11.002).
- [2] F. Della Croce, M. Ghirardi and R. Tadei. Recovering beam search approach for combinatorial optimization problems. *Journal of Heuristics*, 10, 89–104, 2004 (DOI: 10.1023/B:HEUR.0000019987.10818.e0).
- [3] R. S. Farr. Random close packing fractions of log-normal distributions of hard spheres. *Powder Technology*, 245, 28–34, 2013 (DOI: 10.1016/j.powtec.2013.04.009).
- [4] M. Hifi and R. M'Hallah. A literature review on circle and sphere packing problems: models and methodologies. *Advances in Operations Research*, Article ID 150624, 22 p, 2009 (doi.org/10.1155/2009/150624).
- [5] M. Hifi and R. M'Hallah. Beam search and non-linear programming tools for the circular packing problem, *International Journal of Mathematics in Operational Research*, 1, 476–503, 2009 (DOI: 10.1504/IJ-MOR.2009.026278).
- [6] M. Hifi, R. M'Hallah and T. Saadi. Algorithms for the constrained two-staged two-dimensional cutting problem. *INFORMS, Journal on Computing*, 20 212–221, 2008.
- [7] M. Hifi and T. Saadi. A cooperative algorithm for constrained two-staged two-dimensional cutting problems. *International Journal of Mathematics in Operational Research*, 9, 104–124, 2010 (DOI: 10.1504/IJOR.2010.034363).
- [8] M. Hifi and L. Yousef. A dichotomous search-based heuristic for the three-dimensional sphere packing problem. Working paper, Exposed in the Seminary of ROAD Team, Laboratory EPROAD, Université de Picardie Jules Verne, october 2013.
- [9] T. Kubach, A. Bortfeldt, T. Tilli, and H. Gehring. Parallel greedy algorithms for packing unequal spheres into a cuboidal strip or a cuboid. Working Paper, Department of Management Science, University of Magdeburg, (Diskussionsbeitrag der Fakultät für Wirtschaftswissenschaft der FernUniversität in Hagen). No 440, Hagen 2009.
- [10] T. Kubach, A. Bortfeldt, T. Tilli, and H. Gehring. Greedy algorithms for packing unequal sphere into a cuboidal strip or a cuboid. *Asia-Pacific Journal of Operational Research*, 28(06), 739–753, 2011 (DOI: 10.1142/S0217595911003326).
- [11] Y. Li and W. Ji. Stability and convergence analysis of a dynamics-based collective method for random sphere packing. *Journal of Computational Physics*, 250, 373–387, 2013 (DOI: 10.1016/j.jcp.2013.05.023).
- [12] K. Lochmann, L. Oger, and D. Stoyan. Statistical analysis of random sphere packings with variable radius distribution. *Solid State Sciences*. 8(12), 1397–1413, 2006 (DOI: 10.1016/j.solidstatesciences.2006.07.01).
- [13] R. M'Hallah, A. Alkandari, and N. Mladenović. Packing unit spheres into the smallest sphere using VNS and NLP. *Computers & Operations Research*, 40(2), 603–615, 2013 (DOI: 10.1016/j.cor.2012.08.019).
- [14] R. M'Hallah and A. Alkandari. Packing unit spheres into a cube using VNS. *Electronic Notes in Discrete Mathematics*, 39(1), 201–208, 2012.
- [15] P. S. Ow and T.E. Morton. Filtered beam search in scheduling, *International Journal of Production Research*, 26, 297–307, 1988 (DOI:10.1080/00207548808947840).
- [16] K. Soontrapa and Y. Chen. Mono-sized sphere packing algorithm development using optimized Monte Carlo technique. *Advanced Powder Technology*, 24(6), 955–961, 2013 (DOI: 10.1016/j.appt.2013.01.007).
- [17] Y. Stoyan, G. Yaskow, and G. Scheithauer. Packing of various radii solid spheres into a parallelepiped. *Central European Journal of Operational Research*, 11, 389–407, 2003.
- [18] A. Sutou and Y. Dai. Global optimization approach to unequal sphere packing problems in 3D. *Journal of Optimization Theory and Applications*, 114, 671–694, 2002 (DOI: 10.1023/A:1016083231326).
- [19] J. Wang. Packing of unequal spheres and automated radio-surgical treatment planning. *Journal of Combinatorial Optimization*, 3, 453–463, 1999 (DOI: 10.1023/A:1009831621621).
- [20] G. Wascher, H. Haussner and H. Schumann. An improved typology of cutting and packing problems. *European Journal of Operational Research*, 183, 1109–1130, 2007 (DOI: 10.1016/j.ejor.2005.12.047).
- [21] M. Yavuz. Iterated beam search for the combined car sequencing and level scheduling problem. *International Journal of Production Research*, 51, 3698–3718, 2013 (DOI:10.1080/00207543.2013.765068).