Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 397–404
DOI: 10.15439/2014F145
ACSIS, Vol. 2

A Look-Forward Heuristic for Packing Spheres into a Three-Dimensional Bin

Hakim Akeb
ISC Paris Business School
22 boulevard du Fort de Vaux
75017 Paris, France
Email: hakeb@iscparis.com

***Abstract*—In this paper a look-forward heuristic is proposed in order to solve the problem of packing spheres into a three-dimensional bin of fixed height and depth but variable length. The objective is to pack all the spheres into the bin of minimum length. This problem is also known under the name of three-dimensional strip packing problem. The computational investigation, conducted on a set of benchmark instances taken from the literature, shows that the method is effective since it improves most of the best known results.**

I. Introduction

PACKING spheres can be used to model many solid state systems. Indeed, the association of different-sized spheres for example can approximate a given solid form. Packing (non-)identical spheres is for example used in the domain of stereotactic radio surgery radiation therapy (see for example the works of Gavriliouk [3], Sutou and Dai [13], and Wang [15]) where the target areas are delimited by spheres of different sizes.

The problem of packing spheres into a container was studied by several authors in the literature. The spheres can be of identical or different sizes (radii). The problem of packing non-identical spheres into a given 3D container was for example considered by Li and Ji [8] where a dynamics-based collective method for random sphere packing was proposed as well as an application to the problem of packing spheres into a cylinder container. The authors studied also the stability of the method and the convergence of their algorithm. Sutou and Dai [13] used a global optimization approach (including Linear Programming relaxation and branch-and-bound) in order to place unequal spheres inside a three-dimensional container. More precisely, the objective is to maximize the volume of the container (of fixed size) occupied by the placed spheres. This is also called the *Knapsack* version of the problem, i.e., the objective is not to place all the objects but those maximizing the obtained profit. The profit used often corresponds to the volume of the corresponding object placed. Stoyan, Yaskov, and Scheithauer [12] developed a mathematical model in order to place different-sized spheres inside a parallelepiped of fixed length and width but with variable height. The objective is then to minimize the height of the container. The proposed method uses different tools including extreme points and neighborhood search. Solutions are given for a set containing eight instances (designed by the authors) where the number of spheres varies from 20 to 60.

For the case of identical spheres, M'Hallah, Alkandari, and Mladenović [9] for example studied the problem of packing spheres of the same radius into the smallest containing sphere by using Variable Neighborhood Search (VNS) and Non-Linear Programming (NLP). VNS here consists to move some spheres situated in the neighborhood of a given placed sphere, then a NLP procedure is called in order to remove overlapping between spheres. M'Hallah and Alkandari [10] applied the same principle (VNS and NLP) as in [9] to solve the problem of packing unit spheres into the smallest cube. Soontrapa and Chen [11] considered the problem of packing identical spheres into a cube by using a random search technique based on the Monte Carlo method. The problem concerns actually the development of a fuel catalyst layer.

Finally Birgin and Sobral [2] studied the problem of packing identical and non-identical spheres into different three-dimensional containers. The objective is to minimize the dimension of the container. The method proposed by the authors is based on twice-differentiable models as well as non-linear programming.

The problem to solve in this paper is the Three-Dimensional Strip Packing Problem (3DSPP) which is known to be NP-Hard [6]. Given a set S containing n spheres $s_i, 1 \leq i \leq n$ where each sphere has radius r_i and is placed with its center at coordinates (x_i, y_i, z_i) in the Euclidean space. Let also $\mathbb{B}$ be a three-dimensional bin (rectangular cuboid or parallelepiped) of fixed height and depth (H, D) respectively but of unconstrained length L. The objective is then to place the n spheres inside the parallelepiped of minimum length such that no sphere overlaps another sphere and no sphere exceeds the container boundaries. The method presented is based on the use of several tools including the *Maximum Hole Degree* (MHD) heuristic, a modified look-forward strategy, and an interval search.

II. Problem formulation

The three-dimensional bin $\mathbb{B}$ has six faces $\mathbb{F} = \{\text{left}, \text{top}, \text{right}, \text{bottom}, \text{back}, \text{front}\}$ and is placed such that its bottom-left-back corner corresponds to the origin $O(0,0,0)$ of the axes in the Euclidean space as shown in Fig. 1. The length L, the height H, and the depth D of the container are associated with the $\overrightarrow{Ox}$, $\overrightarrow{Oy}$, and $\overrightarrow{Oz}$ axes respectively. Moreover, each

978-83-60810-58-3/$25.00 © 2014, IEEE

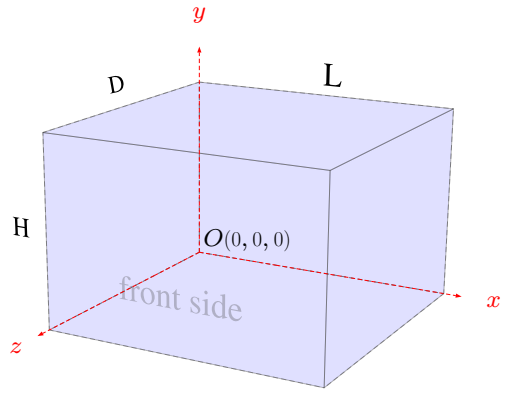

Fig. 1. The three-dimensional bin container placed with its bottom-left-back corner at the origin of the axes in the Euclidean space.

sphere $s_i \in S$ has radius r_i and its center's coordinates are (x_i, y_i, z_i). The 3DSPP can then be formulated as follows:

$$\min L \tag{1}$$

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \geq (r_i + r_j)^2 \quad \text{for } 1 \leq i < j \leq n \tag{2}$$

$$x_i \geq r_i \qquad \forall i \in [1, \ldots, n] \tag{3}$$

$$x_i \leq L - r_i \qquad \forall i \in [1, \ldots, n] \tag{4}$$

$$y_i \geq r_i \qquad \forall i \in [1, \ldots, n] \tag{5}$$

$$y_i \leq H - r_i \qquad \forall i \in [1, \ldots, n] \tag{6}$$

$$z_i \geq r_i \qquad \forall i \in [1, \ldots, n] \tag{7}$$

$$z_i \leq D - r_i \qquad \forall i \in [1, \ldots, n] \tag{8}$$

Equation 1 indicates the objective (value) to minimize (the length L of the bin). Equation 2 is the non-overlapping constraint that verifies that any pair of distinct spheres $(s_i, s_j) \in S^2$ do not overlap each other. Equations 3–8 mean that each sphere must not exceed the boundaries of the container.

The distance between the edges of two distinct spheres s_i and s_j, denoted by $d_{i,j}$, is defined as follows:

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} - r_i - r_j \quad \text{for } i \neq j \tag{9}$$

III. The 3DMHD heuristic for packing spheres into a three-dimensional bin

In this section, a greedy heuristic, denoted by 3DMHD (Three-Dimensional Maximum Hole Degree), for packing spheres into a three-dimensional bin is described. This is in fact the adaptation of the Maximum Hole Degree (MHD) heuristic [4], designed for packing circles, to the three-dimensional case.

Note that a simple way to pack the spheres inside the container consists for example to place the first sphere s_1 at the bottom-left-back corner, i.e., at coordinates (r_1, r_1, r_1). After that, at each step $i, (1 < i \leq n)$ a new sphere is chosen and is placed at the *best* position (that has the maximum hole degree). More precisely, let:

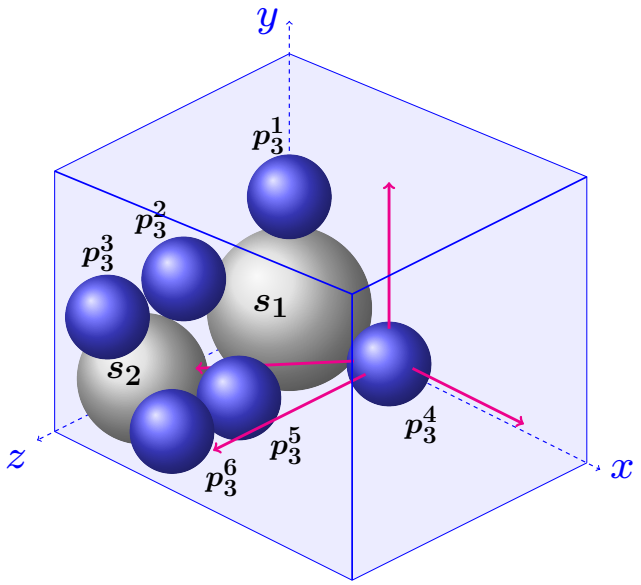

Fig. 2. The 3DMHD heuristic for packing spheres into a three-dimensional bin.

- S_{in} define the set of spheres already placed inside the container.
- S_{out} is the complementary set containing the spheres that are not yet placed (outside). Note that $S_{\text{in}} \cup S_{\text{out}} = S$.
- P denotes the set of possible positions (called *corner positions*) for the spheres of set S_{out}.

Fig. 2 shows an example where two spheres s_1 and s_2 (the two greatest ones) are already placed inside the container $\mathbb{B}$, i.e., $S_{\text{in}} = \{s_1, s_2\}$. The figure also indicates six possible corner positions for packing another sphere s_3. These positions are denoted by $\{p_3^1, \ldots, p_3^6\}$. Each position p_3^k is computed by using three objects, an object may be a sphere already placed or one of the six faces of the parallelepiped. These three objects denote set $\mathscr{T}(p_3^k)$ associated to this position. For example, position p_3^1 is computed by using sphere s_1, the left-edge and the back-face of the container, then $\mathscr{T}(p_3^1) = \{s_1, \text{left}, \text{back}\}$. Similarly, $\mathscr{T}(p_3^2) = \{s_1, s_2, \text{left}\}$.

Generally, let position $p_{i+1}^k \in P$, associated to a sphere of radius r_{i+1}^k, be one of the possible corner positions for the next sphere s_{i+1} to place. Then, the 3DMHD value for position p_{i+1}^k is defined as follows:

$$\lambda(p_{i+1}^k) = \max_{j \in S_{\text{in}} \cup \mathbb{F} \setminus \mathscr{T}(p_{i+1}^k)} 1 - \frac{d_{i+1,j}^k}{r_{i+1}^k} \tag{10}$$

Equation 10 means that the hole degree $\lambda(p_{i+1}^k)$ is computed for each position in the set of positions P (associated with set S_{out}) for the next sphere to place. This value uses the distance $d_{i+1,j}^k$ between the edge of position p_{i+1}^k and the nearest object j in the set $S_{\text{in}} \cup \mathbb{F} \setminus \mathscr{T}(p_{i+1}^k)$ that contains the spheres already placed, the six faces $(\mathbb{F})$ of the container but

Algorithm 1 The 3DMHD greedy heuristic

Require: Set S_{in} containing spheres already placed, S_{out} containing the remaining spheres to place, set P indicating the possible positions for spheres in S_{out} and the current length L of the container.

Ensure: TRUE if all the spheres are packed into the container, FALSE otherwise.

1: $i \leftarrow |S_{\text{in}}|$;
2: **while** $(P \neq \emptyset)$ **do**
3: Compute/update the 3DMHD value for each corner position $p \in P$;
4: Place the next sphere s_{i+1} at position p^* that has the maximum hole degree as shown in equation 11.
5: Move sphere s_{i+1} from S_{out} to S_{in};
6: Remove from set P the positions that overlap the new inserted sphere;
7: Compute new positions by using the new inserted sphere and the other objects already placed;
8: $i \leftarrow i+1$;
9: **end while**
10: **if** $(i = n)$ **then**
11: Set $L \leftarrow \max(x_i + r_i)$;
12: Update the best known length if L is smaller than this value;
13: **return** TRUE;
14: **else**
15: **return** FALSE;
16: **end if**

excluding set $\mathscr{T}(p^k_{i+1})$. The distance is divided by the radius r^k_{i+1} of the sphere corresponding to position p^k_{i+1}. Note that if a given position touches more than three objects, then $\lambda = 1$, meaning that this positions has a high probability to be chosen for placing the next sphere.

For example, Fig. 2 indicates the distance between position p^4_3 and four other objects: sphere s_2, the front face, the top face, and finally the right face of the container.

Then, the 3DMHD heuristic places the next sphere at position $p^* \in P$ that corresponds to the maximum value of $\lambda(p^k_{i+1})$ as indicated in equation 11.

$$p^* = \arg\max_{p^k_{i+1}} \lambda(p^k_{i+1}) \tag{11}$$

Algorithm 1 explains how the 3DMHD heuristic proceeds in order to place a set of spheres inside the container $\mathbb{B}$ of dimensions $(L \times H \times D)$. Procedure 3DMHD receives a partial solution $\{S_{\text{in}}, S_{\text{out}}, P\}$ indicating the spheres already packed into the container, the remaining spheres, and the set of corner positions for spheres in S_{out} respectively. The current length L of the container is also transmitted to the procedure. The heuristic's output is a boolean value indicating whether yes or no all the spheres were successfully packed into the container. So procedure 3DMHD is able to start with any partial solution were the number of spheres already packed is greater than or equal to zero.

At line 1 in algorithm 1 counter i indicating the number of spheres already packed is set to the number of spheres inside S_{in}. After that, in the **while** loop, the 3DMHD value is computed for each position $p \in P$ (line 3), this is done by using the formula of equation 10. At line 4, the best position p^* is chosen in order to place the next sphere s_{i+1}. After that, the new sphere moves from set S_{out} to set S_{in} (line 5) and the set of positions P is updated by removing those overlapping the new inserted sphere (line 6) and by computing new positions by using the new inserted sphere (line 7). Counter i is then incremented at line 8. The **while** loop ends when the set of positions P becomes empty meaning that no additional sphere can be packed. Then two cases can be distinguished: if $i = n$ then all the n spheres were successfully packed into the container. In this case the procedure computes at line 11 the exact value for L which is equal to $\max(x_i + r_i)$, i.e, using the most right placed sphere $s_i \in S_{\text{in}}$. If the obtained value L is smaller than the best known length then this value is updated (line 12) and the procedure returns TRUE (line 13). If $i < n$ then a feasible packing was not obtained and the procedure returns FALSE (line 15), this means that the current length L of the container has to be changed.

Note that one can test several values for the length L of the bin in order to try to compute a feasible solution with the 3DMHD heuristic (not necessarily a binary search but other more efficient strategies). This can be done for example by decreasing the length from an upper bound to a lower bound. Indeed, this strategy may escape from local optima (see Section IV below).

A. A Multi-Level Look-Forward strategy for the 3DSPP

This section describes a look-forward algorithm designed for the three-dimensional strip packing problem.

Look-forward (LF) strategies (see for example [7], [4], [1]) are often used in order to improve the results obtained by different algorithms. Its objective is to evaluate the future behavior of a decision (choice) made at a given step of the problem solving process. For example, in a greedy algorithm, the *best* decision among all the possible decisions is made at step i in order to move to the next step $i+1$. The look-ahead strategy tries several (or all) choices at step i and see what will be obtained when executing the greedy algorithm few steps ahead of until the end (this is often executed on a copy of the partial solution). After that, the decision actually made at step i is the one that had the best behavior or led to the best outcome.

In packing problems, the look-forward strategy often uses a parameter called *density* of a solution. The density of a solution S_{in}, denoted by $density(S_{\text{in}})$ is equal to the sum of the volumes of spheres in S_{in} divided by the volume of the container as indicated in Equation 12. The look-forward strategy selects then the decision that will obtain the highest density.

$$density(S_{\text{in}}) = \frac{4 \times \pi \times \sum_{i=1}^{|S_{\text{in}}|}(r_i^3)}{3 \times L \times H \times D} \tag{12}$$

Algorithm 2 LF-3DMHD

Require: Sets S_{in}, S_{out}, P, and the current length L of the container.
Ensure: TRUE if all the spheres are packed into the container, FALSE otherwise.

1: $i \leftarrow |S_{\text{in}}|$;
2: found $\leftarrow$ FALSE;
3: **while** ($P \neq \emptyset$ and found=FALSE) **do**
4: Sort the positions of set P in decreasing order of their hole degree (λ) value;
5: **for each** of the first $\psi_1 \times |P|$ positions $p \in P$ **do**
6: Let $S'_{\text{in}} \leftarrow S_{\text{in}}$, $S'_{\text{out}} \leftarrow S_{\text{out}}$ and $P' \leftarrow P$;
7: Insert the next sphere s'_{i+1} into S'_{in} at position p and update sets $S'_{\text{in}}, S'_{\text{out}}$, and P';
8: $density^* \leftarrow 0$;
9: Sort the positions of set P' in decreasing order of their hole degree (λ) value;
10: **for each** of the first $\psi_2 \times |P'|$ positions $p' \in P'$ **do**
11: Let $S''_{\text{in}} \leftarrow S'_{\text{in}}$, $S''_{\text{out}} \leftarrow S'_{\text{out}}$ and $P'' \leftarrow P'$;
12: Insert the next sphere s''_{i+2} into S''_{in} at position p' and update sets $S''_{\text{in}}, S''_{\text{out}}$, and P'';
13: found $\leftarrow$ 3DMHD($S''_{\text{in}}, S''_{\text{out}}, P'', L$);
14: **if** (found=TRUE) **then**
15: Set L equal to the length computed by 3DMHD;
16: **return** TRUE;
17: **else**
18: **if** ($density(S''_{\text{in}}) > density^*$) **then**
19: $density^* \leftarrow density(S''_{\text{in}})$;
20: **end if**
21: **end if**
22: **end for**
23: Assign to position $p \in P$ the density $density^*$ obtained after calling 3DMHD;
24: **end for**
25: Let $p^* \in P$ be the position that has obtained the highest density $density^*$;
26: Place the next sphere s_{i+1} at position p^* and move sphere s_{i+1} from S_{out} to S_{in};
27: Remove from set P the positions that overlap the new inserted sphere;
28: Compute new positions by using the new inserted sphere;
29: $i \leftarrow i+1$;
30: **end while**
31: **if** ($i = n$) **then**
32: Set $L \leftarrow \max(x_i + r_i)$ where x_i and r_i are the $x-$coordinate and the radius of sphere $s_i \in S_{\text{in}}$;
33: Update the best known length if L is smaller than this value;
34: **return** TRUE;
35: **else**
36: **return** FALSE;
37: **end if**

The algorithm that implements the look-forward strategy, denoted by LF-3DMHD, is described in algorithm 2. It receives as input parameters a partial solution $\{S_{\text{in}}, S_{\text{out}}, P\}$ where $|S_{\text{in}}|$ spheres are already packed, set S_{out} denotes the spheres that remain to pack and P contains the corner positions for spheres of set S_{out}. The algorithm receives also the current length (L) of the container. Algorithm LF-3DMHD returns TRUE if it succeeds to compute a feasible solution, FALSE otherwise.

Instruction at line 1 of algorithm 2 sets the counter i indicating the number of spheres already packed. At line 2, a boolean value (found) is set to FALSE (this indicator is set to TRUE if a feasible solution is obtained).

The difference between the look-forward strategy and the 3DMHD heuristic (described in algorithm 1) is that the look-forward tries (evaluates) several positions at each step of the packing process while the greedy heuristic 3DMHD selects, at each step, only one position (the best one) in order to pack the next sphere. Moreover, the look-forward used here contains two levels, i.e., it places the two next spheres and continues the placement of the remaining spheres by using the greedy heuristic 3DMHD (algorithm 1). This is implemented by using two nested **for** loops that begin at lines 5 and 10 respectively. In addition, the first **for** loop considers only the best $\psi_1 \times |P|$ positions with $0 < \psi_1 \leq 1$ and P is the set of corner positions in the first level. In the second **for** loop the algorithm considers only the best $\psi_2 \times |P'|$ with $0 < \psi_2 \leq 1$ and P' is the set of corner positions in the second level. So if for example $\psi_1 =$

0.5, then only the half best positions in the list of positions are considered in the first level of the look-forward strategy, and if $\psi_1 = 1$, then this means that all the positions will be considered. Using a value of ψ_1 and ψ_2 lower than 1 will of course decrease the computation time of the algorithm.

More precisely, the positions in set P are sorted in decreasing order of their hole degree value (λ). This is done at line 4. In the first **for** loop, the algorithm expends the current solution $\{S_{\text{in}}, S_{\text{out}}, P\}$ by choosing at each time a position $p \in P$ by creating a copy of the current solution denoted by $\{S'_{\text{in}}, S'_{\text{out}}, P'\}$ (line 6) and inserts the next sphere s'_{i+1} at that position (line 7). At line 8, a variable called $density^*$ is set to 0. This parameter is used in order to store the best density obtained in the second level of the look-forward. The corner positions of set P' are after that sorted in decreasing order of their λ value (line 9). The second **for** loop starts at line 10, after placing sphere s'_{i+1}. Like in the first level, only a proportion $\psi_2 \times |P'|$ of the best corner positions are taken into account in set P'. Then for each selected position $p' \in P'$, the procedure creates a copy, denoted by $\{S''_{\text{in}}, S''_{\text{out}}, P''\}$, for the current partial solution $\{S'_{\text{in}}, S'_{\text{out}}, P'\}$ (line 11). After that, the next sphere s''_{i+2} is placed at position p' (line 12). Then, the partial solution is evaluated by calling the 3DMHD heuristic (algorithm 1) at line 13 in order to try to pack the remaining $n-i-2$ spheres. If 3DMHD succeeded to pack all the remaining spheres, then it returns TRUE (line 14), the current length of the container is then set to the length computed by 3DMHD (line 15). The algorithm then exits at line 16 since it has succeeded to pack all the spheres (it returns TRUE). Otherwise (found=FALSE), this means that 3DMHD did not succeed to place all the remaining spheres, then the density of the obtained solution $density(S''_{\text{in}})$ is assigned to the best known density $density^*$ if a better value is obtained (line 19). The second **for** loop ends when all the selected positions $p' \in P'$ are evaluated and the best obtained density $(density^*)$ is assigned to position $p \in P$ that is currently considered in the first **for** loop.

At the output of the two **for** loops, the next sphere s_{i+1} is placed at position p^* (line 26) that has obtained the best density after calling 3DMHD. The set P of positions is then updated at line 27 by removing those that overlap the new inserted sphere and new positions are computed at line 28. The number of placed spheres (i) is incremented at line 29.

Instructions of the **while** loop (lines 3–30) are executed until a feasible solution is obtained (found=TRUE) or the set of positions P becomes empty. So if $i = n$ (line 31), this means that a feasible solution is reached, then the true length of the container is computed at line 32 and the best known length is updated if a better one is obtained (line 33). The algorithm returns TRUE (line 34). If $(i < n)$, then this means that algorithm LF-3DMHD did not succeed to compute a feasible solution and returns FALSE (line 36).

Finally, algorithm 2 can for example be called by an interval-search procedure that modifies the value of the length L of the container at each call as described in section III-B below.

Algorithm 3 (LF2)

Require: Instance S containing n spheres, the height H, and the depth D of the three-dimensional bin $\mathbb{B}$;

Ensure: The best length L^* obtained and the corresponding density $density^*$;

1: Set $L_{\min} \leftarrow \max\left(\frac{4\times\pi\times\sum_{i=1}^{n}(r_i^3)}{3\times H\times D}, 2\times r_{\max}\right)$ be the lower bound of the interval search;
2: Set $L_{\max} \leftarrow 3 \times L_{\min}$;
3: Set $\Delta L \leftarrow 0.01$;
4: $L \leftarrow L_{\max}$;
5: $L^* \leftarrow L$;
6: $density^* \leftarrow 0$;
7: **while** $(L \geq L_{\min})$ **do**
8: $S_{\text{in}} \leftarrow \emptyset$;
9: $S_{\text{out}} \leftarrow S$;
10: Create set P of positions corresponding to the placement of each sphere $s_i \in S$ of radius r_i at position (r_i, r_i, r_i) in the bin of dimensions $L \times H \times D$;
11: found $\leftarrow$ LF-3DMHD$(S_{\text{in}}, S_{\text{out}}, P, L)$;
12: **if** (found = TRUE) **then**
13: Update L if a lower value was obtained by LF-3DMHD;
14: $L^* \leftarrow L$;
15: Update the best density $density^*$;
16: **end if**
17: $L \leftarrow L - \Delta L$;
18: **end while**

B. Interval Search for Computing the Best Packing

This section describes the interval search, denoted by LF2 and described in algorithm 3, used in order to compute the best feasible packing. The search principle consists to decrease the value of the bin length L from an upper bound $L_{\max}$ by a given step ΔL until matching the lower bound $L_{\min}$. The search may also stop if the computation time limit is reached.

Algorithm 3 (LF2) explains how the heuristic proceeds in order to compute the best packing of the n spheres into the three-dimensional bin of minimum length. Procedure LF2 receives as input parameters the instance $S = \{s_1, ..., s_n\}$ containing n spheres of radii $r_1, ..., r_n$ respectively as well as the height H and the depth D of the three-dimensional bin $\mathbb{B}$. The output of the algorithm is the best length found L^* and the corresponding density $(density^*)$ that is equal to the sum of the volumes of the spheres divided by the volume of the bin $(L^* \times H \times D)$.

The continuous lower bound for the length of the container is used as the minimum value $(L_{\min})$ of the interval search (line 1). Note that if this value is lower than the diameter of the greatest sphere, then this diameter $(2 \times r_{\max})$ is used as the lower bound. The upper bound $L_{\max}$ of the interval search is set equal to $3 \times L_{\min}$. The step ΔL with which the length is decreased at each step is defined at line 3, this value is set to 0.01. The length of the container is then set equal to the upper bound $L \leftarrow L_{\max}$ (line 4) and the best length L^* is set equal to

L at line 5. The next instruction serves to initialize the value of the best known density ($density^*$) associated with the best length L^* (line 6).

After that, at each step in the **while** loop (lines 7–18) a starting configuration is created where the set S_{in} of spheres already packed is set equal to the empty set (line 8) and the set of the remaining spheres to pack (S_{out}) is set equal to the instance S. List P of positions for spheres in set S_{out} is then computed (line 10) so that each position is placed at (r_i, r_i, r_i). This is a novel method because most of the greedy heuristics start by placing one or several objects, here only the list of positions is computed and no object is placed.

Algorithm LF-3DMHD is then called at line 11 in order to try to compute a feasible solution (packing the n spheres into the bin of dimensions $L \times H \times D$.) If procedure LF-3DMHD succeeded to pack the n spheres (found=TRUE) then the value of L is updated if a lower value was computed by LF-3DMHD (line 13) and the best length L^* is set equal to L (line 14). The best density $density^*$, corresponding to L^* is then updated at line 15. The value of the length L is after that deceased (line 17), even if a feasible solution was not obtained by procedure LF-3DMHD. Indeed, this method is, to our opinion, preferable to a basic dichotomous search where the dimensions of the container are increased when a feasible solution was not obtained. This is not always a good strategy because, in our case for example, if a feasible solution is not obtained by using a given value of the length L, it may be obtained by using a lower value $L - \Delta L$. In fact, decreasing the value of the length L is a good strategy to escape from local optima in order to increase the solution quality.

Algorithm LF2 stops when the value of L becomes lower than the lower bound $L_{\min}$ or when the computation time limit is reached.

IV. Computational Results

In order to evaluate the performance of the proposed algorithm LF2 (algorithm 3), two sets of instances were considered:

- Six instances, denoted by SYS, proposed by Stoyan, Yaskov, and Scheithauer [12]. The number of spheres varies from 25 to 60. All the spheres have different radii in each instance (strongly heterogeneous instances).
- Twelve instances, denoted by KBG1,...,KBG12, proposed by Kubach, Bortfeldt, Tilli, and Gehring [5]. Here, the number of spheres is equal to 30 for the first six instances and 50 spheres for the six last ones. Moreover, instances KBG1–KBG3 and KBG7–KBG9 are strongly heterogeneous since all the radii are different. The other six instances KBG4–KBG6 and KBG10–KBG12 are weakly heterogeneous because there are only $n/10$ different radii in each instance, each radius is duplicated 10 times.

The different procedures and algorithms are coded in C++ language and executed under Linux environment on a computer with a 2.4 GHz processor. The results obtained are compared to those given the B1.6 algorithm [5] that is mainly based on a look-forward strategy and starting configurations, the results taken from [5] were also obtained on a 2.4 GHz processor. Algorithm B1.6 is in fact the adaptation of algorithm B1.5 [4] for placing circles inside a rectangular container to the three-dimensional case. Algorithm B16 however tries more starting configurations than B1.5 does. In addition, B1.6 uses a parameter denoted by τ $(0 < \tau \leq 1)$ that serves to indicate the proportion of corner positions evaluated at each step of the look-forward process. The authors in [5] tried two values: $\tau = 0.8$ and $\tau = 1$. The first case means that only 80% of positions are evaluated by the look-forward while the second case means that all positions are evaluated. So in fact, algorithm B1.6 is executed two times (60 minutes for each value of τ). It is to note that the proposed algorithm LF2 is executed only once during 60 minutes on each instance.

In algorithm LF2, the number of positions evaluated by the look-forward is set to 50% in the two levels ($\psi_1 = \psi_2 = 0.5$). So at each time the corner positions are sorted in decreasing order of their hole degree λ and only the first half ones are evaluated. The objective is of course to save computation time.

Table I shows the results obtained by the different algorithms. Column 1 indicates the instance's name and column 2 its size. The two next columns indicate the height H and the depth D of the container. Column 5 (SYS) indicates the results (best length) obtained by the SYS method [12] on instances SYS1–SYS6. Columns 6 and 7 contain the best results (the best length L and the corresponding density respectively) obtained by algorithm B1.6 on the 18 instances (SYS and KBG) when parameter τ is set equal to 0.8 (80% of positions are evaluated by the look-forward). The next two columns display the same results as the two previous columns but when parameter τ is set equal to 1 (all the positions are evaluated in the look-forward). Columns 10–14 contain the results obtained by the proposed algorithm LF2 on all the considered instances. Column 10 (L) gives the best length obtained and column 11 the corresponding density. Column 12 (t^*) indicates the time needed by algorithm LF2 for computing the best solution. The two last columns of table I indicate the percentage of improvement obtained by the proposed algorithm LF2 on algorithm B1.6. Column "Imp. 0.8" shows the improvement obtained when considering B1.6 with $\tau = 0.8$ and the last column "Imp. 1" is the percentage of improvement when B1.6 with $\tau = 1$ is considered. Note that the percentage of improvement is computed as follows: Imp. $= \frac{Density(\text{LF2}) - Density(\text{B1.6})}{Density(\text{LF2})}$. Finally, note that some solutions for KBG instances are optimal, this is the case for instances KBG2, KBG4, and KBG10. This is why there is an "*" before each value in the three columns that contain the corresponding density in table I.

The results of table I indicate that the proposed algorithm LF2 improves all the results obtained by the SYS method on the first six instances (the results of the SYS method are not known for instances KBG). Algorithm LF2 improves B1.6 with $\tau = 0.8$ in 11 cases out of 18 and the two algorithms reach the optimal value of the container length for instances KBG2, KBG4, and KBG10 since the computed length is equal to the greatest diameter in the instance. Algorithm B1.6 with $\tau = 0.8$ remain better than LF2 on instances KBTG5, KBTG6,

TABLE I
RESULTS OBTAINED BY THE PROPOSED METHOD LF2 ON INSTANCES SYS AND KBTG

				SYS	**B1.6 $\tau = 0.8$ (3600 s)**		**B1.6 $\tau = 1$ (3600 s)**		**Algorithm LF2 (3600 s)**				
Instance	n	H	D	L	L	*Density*	L	*Density*	L	*Density*	t^*	Imp. 0.8	Imp. 1
SYS1	25	5.5	6.9	9.8668	9.5397	53.160	9.2874	54.604	9.2234	**54.983**	1911	3.32	0.69
SYS2	35	6.5	7.9	9.6221	9.2608	55.077	9.1280	55.878	9.1138	**55.965**	2680	1.59	0.16
SYS3	40	5.5	6.9	9.4729	9.0540	53.554	8.9850	53.965	8.9316	**54.288**	2900	1.35	0.59
SYS4	45	8.5	9.9	11.0862	10.8932	53.771	10.8760	53.856	10.7653	**54.410**	3600	1.17	1.02
SYS5	50	8.5	9.9	11.6453	11.2170	54.975	11.3494	54.334	11.1948	**55.084**	2030	0.20	1.36
SYS6	60	8.5	9.9	12.8416	12.5339	54.346	12.3745	55.046	12.2519	**55.597**	3330	2.25	0.99
KBG1	30	10	10		–	53.772	–	54.096	11.2063	**54.494**	2400	1.32	0.73
KBG2	30	10	10		–	* **30.071**	–	* **30.071**	1.9900	* **30.071**	2	0.00	0.00
KBG3	30	10	10		–	50.614	–	51.387	18.9231	**51.693**	3300	2.09	0.59
KBG4	30	10	10		–	* **37.765**	–	* **37.765**	1.9960	* **37.765**	1	0.00	0.00
KBG5	30	10	10		–	**48.278**	–	**48.278**	1.9279	48.181	1930	-0.20	-0.20
KBG6	30	10	10		–	**48.966**	–	47.792	18.8807	48.847	3400	-0.24	2.16
KBG7	50	10	10		–	54.623	–	55.372	13.5075	**55.824**	2030	2.15	0.81
KBG8	50	10	10		–	44.924	–	45.060	2.6027	**46.639**	326	3.68	3.39
KBG9	50	10	10		–	52.210	–	**52.732**	29.7023	51.783	3420	-0.82	-1.83
KBG10	50	10	10		–	* **51.866**	–	* **51.866**	1.8100	* **51.866**	9	0.00	0.00
KBG11	50	10	10		–	51.629	–	**52.708**	5.2640	52.658	420	1.95	-0.09
KBG12	50	10	10		–	**52.120**	–	51.757	22.2060	52.063	1000	-0.11	0.59
Average						50.096		50.365		**50.678**		1.09	0.61

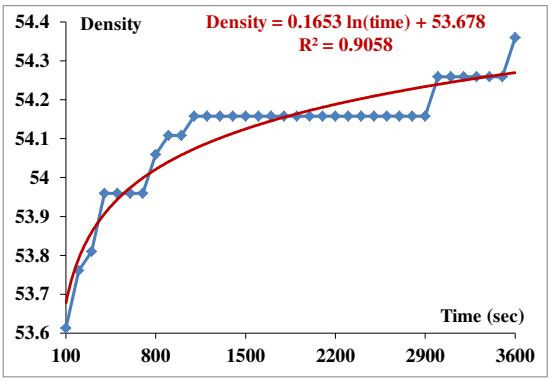

Fig. 3. Estimation of the evolution of the best density obtained with the time by algorithm LF2 on instance SYS4 ($n = 45$) spheres. The regression is based on a logarithmic function, $R^2 = 90.58\%$.

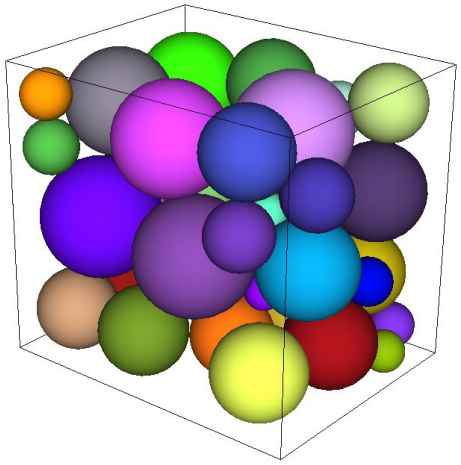

Fig. 4. Solution obtained by algorithm LF2 on instance SYS4 ($n = 45$, $L = 10.7653$ ($Density = 54.410\%$)).

KBTG9, and KBG12). Finally, the last row of the table (column "Imp. 0.8") indicates that algorithm LF2 improves B1.6 with 1.09% in average.

Table I indicates also that algorithm LF2 improves B1.6 with $\tau = 1$ in 12 cases. The two algorithm reach the optimal solution on instances KBG2, KBG4, and KBG10. And algorithm B1.6 with $\tau = 1$ is better than LF2 on instances KBTG5, KBTG9, and KBG11) but the percentage of improvement has decreased to 0.61%.

Fig. 3 indicates the evolution of the density of the solution according to the computation time on instance SYS4 (n=45 spheres). The evolution follows a logarithmic function with a coefficient of determination $R^2 > 90\%$. This means that the density of the obtained solution begins by increasing quickly since the length is near to the upper bound $L_{\max}$ and it is then easier to compute a feasible solution. The improvement of the density slows down when the length approaches the lower bound $L_{\min}$.

Fig. 4 gives the solution obtained by the proposed algorithm LF2 on instance SYS4 that has 35 spheres. The best length obtained is equal to 10.7653 (the best value obtained by B1.6 was 10.8760), this corresponds to an improvement of 1.02%.

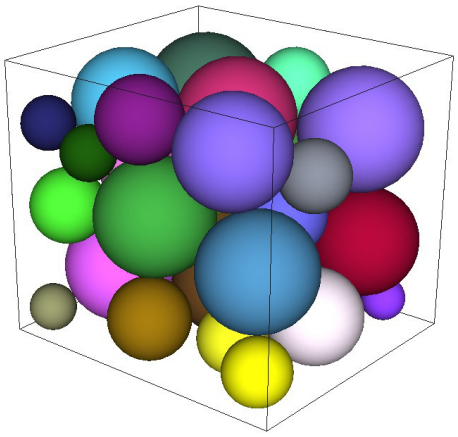

Fig. 5. Solution obtained by algorithm LF2 on instance KBG1 ($n = 30$, $Density = 54.494\%$).

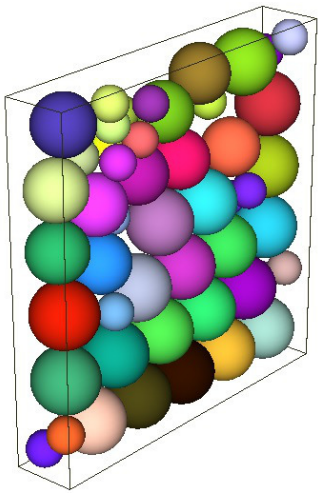

Fig. 6. Optimal solution obtained by algorithm LF2 on instance KBG10 ($n = 50$, $Density = 51.866\%$).

Fig. 5 displays the solution obtained by algorithm LF2 on instance KBG1 that contains 30 spheres. The density obtained is equal to 54.494% (the best value obtained by B1.6 was 54.096%), so the improvement obtained is equal to 0.73%.

Finally, Fig. 6 displays the optimal solution obtained by algorithm LF2 on instance KBG10 that contains 50 spheres but where the number of different radii is only 5. The optimal density is in this case equal to 51.866% and the corresponding optimal length is equal to $2 \times r_{\max}$, i.e., $L = 1.810$, where $r_{\max} = 0.905$ is the greatest radius in the instance.

V. Conclusion

In this paper, a look-forward heuristic was proposed in order to solve the problem of packing spheres into a three-dimensional bin. The first novelty is that method starts with an empty configuration instead of placing one or several pieces inside the container. The second difference is that the interval search proceeds by decreasing the value of the length of the bin instead of using a dichotomous search, the objective is to escape from local optima. Finally the look-forward procedure uses a double search (two levels) instead of one level.

The obtained results on the tested instances showed that the proposed method is effective since it has succeeded to improve or reach almost all the best known results published in the literature. As a future work, it will be interesting to design a new heuristic for packing weakly heterogeneous spheres because it is well-known that the MHD heuristic was designed for packing strongly heterogeneous circles and spheres.

References

[1] H. Akeb, M. Hifi, and D. Lazure, "An Improved Algorithm for the Strip Packing Problem," Proceedings of the Federated Conference on Computer Science and Information Systems, FedCSIS 2012, Wroclaw, Poland, pp. 357–364. ISBN 978-83-60810-51-4. https://fedcsis.org/proceedings/2012/pliks/93.pdf

[2] E.G. Birgin and F.N.C. Sobral, "Minimizing the object dimensions in circle and sphere packing problems," *Compt. Oper. Res.*, vol. 35, 2008, pp. 2357–2375. http://dx.doi.org/10.1016/j.cor.2006.11.002

[3] E. O. Gavriliouk, "Unequal sphere packing problem in the context of stereotactic radiosurgery," Shaker Verlag, 2007, 96 pages.

[4] W. Q. Huang, Y. Li, H. Akeb, and C. M. Li, Y., "Greedy algorithms for packing unequal circles into a rectangular container," *J. Oper. Res. Soc.,* vol. 56, 2005, pp. 539–548. http://dx.doi.org/10.1057/palgrave.jors.2601836

[5] T. Kubach, A. Bortfeldt, T. Tilli, and H. Gehring, "Greedy algorithms for packing unequal sphere into a cuboidal strip or a cuboid," *Asia Pac. J Oper. Res.*, vol. 28, 2011, pp. 739–753. http://dx.doi.org/10.1142/S0217595911003326

[6] L.K. Lenstra and A.H.G Rinnooy Kan, "Complexity of packing, covering, and partitioning problems," In Schrijver A (ed.), Packing and Covering in Combinatorics. Amsterdam: Mathematisch Centrum, 1979, pp. 275–291.

[7] M. Lin, R. Chen, and J. S. Liu, "Lookahead Strategies for Sequential Monte Carlo," *Statist. Sci.*, vol 28, 2013, pp. 69–94. http://dx.doi.org/10.1214/12-STS401

[8] Y. Li and W. Ji, "Stability and convergence analysis of a dynamics-based collective method for random sphere packing," *J. Comput. Phys.*, vol. 250, 2013, pp. 373–387. http://dx.doi.org/10.1016/j.jcp.2013.05.023

[9] R. M'Hallah, A. Alkandari, and N. Mladenović, "Packing unit spheres into the smallest sphere using VNS and NLP," *Compt. Oper. Res.*, vol. 40, 2013, pp. 603–615. http://dx.doi.org/10.1016/j.cor.2012.08.019

[10] R. M'Hallah and A. Alkandari, "Packing unit spheres into a cube using VNS," *Electron. Notes Discrete Math.*, vol. 39, 2012, pp. 201–208. http://dx.doi.org/10.1016/j.endm.2012.10.027

[11] K. Soontrapa and Y. Chen, "Mono-sized sphere packing algorithm development using optimized Monte Carlo technique," *Adv. Powder Technol.*, vol. 24(6), 2013, pp. 955–961. http://dx.doi.org/10.1016/j.apt.2013.01.007

[12] Y. Stoyan, G. Yaskow, and G. Scheithauer, "Packing of various radii solid spheres into a parallelepiped," *Cent. Europ. J. Oper. Re.*, vol. 11, 2003, pp. 389–407.

[13] A. Sutou and Y. Dai, "Global optimization approach to unequal sphere packing problems in 3D," *J. Optimiz. Theory App.*, vol. 114, 2002, pp. 671–694. http://dx.doi.org/10.1023/A:1016083231326

[14] W. Visscher and M. Bolsterli, "Random packing of equal and unequal spheres in two and three dimensions," *Nature*, vol. 239, 1972, pp. 504–507. http://dx.doi.org/10.1038/239504a0

[15] J. Wang, "Packing of unequal spheres and automated radiosurgical treatment planning," *J. Comb. Optim.*, vol. 3, 1999, pp. 453–463. http://dx.doi.org/10.1023/A:1009831621621