A Partial Depth-Search Heuristic for Packing Spheres

HAKIM AKEB
ISC Paris Business School
22 Bd du Fort de Vaux, 75017 Paris
FRANCE
hakeb@iscparis.com

Abstract: This paper proposes a new heuristic for packing non-identical spheres into a three-dimensional container of fixed dimensions. Given a set that contains n spheres, the objective is to place a subset of spheres so as to maximize the volume occupied by these ones. The proposed heuristic is based on an idea that applies a two-level look-forward search. The computational investigation indicates that the heuristic is effective since it improves most of the best known results in the literature on the used instances.

Key–Words: Packing problems, Packing spheres, Heuristic, Look-Forward, Knapsack

1 Introduction

Cutting & Packing (C&P) problems have many industrial and engineering applications. Two main cases are known: the two-dimensional (2D) version and the three-dimensional (3D) one.

The two dimensional version of a C&P problem consists to cut/pack a given object from/into a given container. The most known applications concern the wood, textile, metal, and glass industries. The objective is often the same: saving material for cutting problems and saving space for packing ones. The objects to cut or to pack may have rectangular shapes (cf. Martello and Monaci [10] or Lópes and Beasley [9]), circular shapes (cf. George et al. [2]), or irregular shapes (cf. Alvez-Valdes et al. [1] and Martinez-Sykora et al. [11]). For example, when cutting a given set of objects from a rectangular plate, the objective is to minimize the waste (surface between the cutted shapes).

Even if 3D cutting-and-packing problems are known since the 70's (see for example [14]), they received more attention in the recent years. In a 3D C&P problem, the objects, as well as the container, are three-dimensional. Packing of cubes inside a containing cube is the most known problem because it has a great number of real-life applications such as storage and transportation (cf. Joung and Noh [6]).

A sphere packing problem (SPP)consists to pack identical or different–sized spheres into a container that may be a cube, a sphere or any other 3D geometrical shape. SPP is for example used in Physics to model some solid states or objects (cf. Zauner [15]) as well as in Medicine (cf. Wang [16]).

Two main families can be distinguished in SPP. The first one consists to pack identical spheres that is for example used in random sphere packing which is in relation with the study of granular material behavior. For example, Soontrapa and Chen [13] studied the problem of packing identical spheres into a cube by applying a Monte Carlo based algorithm. M'Hallah et al. [12] proposed a Variable Neighborhood Search based algorithm, coupled with a Non–Linear Programming in order to pack identical sphere into the smallest containing sphere. Li and Ji [8] proposed a Quasi-Dynamics Method for random sphere packing and showed an example of packing identical spheres into a cylinder.

The second family in SPP consists to pack non-identical spheres. Hiti and Bernacki [4] used spheres in order to model powder–based micro–structures and proposed an algorithm that is based on a drop–and–roll principle. The Monte Carlo technique was for example used by He et al. [3] for the problem of random sphere packing of unequal spheres into a cube.

In this paper, we consider the problem of packing non-identical spheres into a three-dimensional bin of fixed dimensions. Given a set $S = \{s_1, ..., s_n\}$ of spheres, where sphere $s_i \in S$ has radii r_i, as well as a three dimensional bin $B = L \times H \times D$ where L, H, D represent the length, the height and the depth of B respectively, the objective is to place a subset $S' \subseteq S$ of spheres in order to maximize the volume occupied in the bin by the placed spheres. It is to note that each placed sphere must not overlap any other placed sphere and no sphere exceeds the bin's boundary. The proposed method is based on a modified look-forward strategy coupled with a local search method.

2 Problem Formulation and a Constructive Heuristic for SPP

In this section, the mathematical formulation for the studied problem will be given. Before that, here are some definitions.

Definition 1 *A partial solution $S_{in} \subseteq S$ corresponds to the set of spheres already placed inside the three-dimensional bin B.*

Definition 2 *The set of spheres that are not placed is denoted by S_{out}. Note that $S = S_{in} \cup S_{out}$ at any time.*

Definition 3 *The set of possible positions to place spheres of set S_{out} is denoted by P such that each placed sphere s_i at position $p \in P$ must touch three elements, an element may be an already placed sphere or one of the six faces of the bin. The set containing these three elements is denoted by $T(p)$.*

Definition 4 *The density of a partial solution, denoted by $\text{density}(S_{in})$, corresponds to the sum of volumes of spheres in set S_{in} divided by the volume of B, i.e., $L \times H \times D$.*

Definition 5 *Bin B is characterized by its six faces $F = \{left, right, top, bottom, front, back\}$.*

Definition 6 *The distance between spheres s_i and s_j, denoted by d_{ij}, is equal to the distance between the centers of s_i and s_j minus the sum $r_i + r_j$ as indicated in (1). This is in fact the minimum distance between the boundaries of the two spheres.*

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} - (r_i + r_j) \quad (1)$$

In addition to the distance d_{ij} between two spheres s_j and s_j, we need to define the distance between sphere s_i and the border of the container.

Definition 7 *The distance between sphere s_i, placed at coordinates (x_i, y_i, z_i), and the six face of the bin are defined as follows: $d_{i,left} = x_i - r_i$, $d_{i,right} = L - x_i - r_i$, $d_{i,bottom} = y_i - r_i$, $d_{i,top} = H - y_i - r_i$, $d_{i,back} = z_i - r_i$, $d_{i,front} = D - z_i - r_i$.*

2.1 Problem Formulation

Container B is placed with its bottom–left–back corner at coordinates $(0, 0, 0)$ in the 3D-Euclidean space as indicated in Fig. 1. In addition, each sphere $s_i \in S, 1 \leq i \leq n$ has radius r_i and is placed with its center at coordinates (x_i, y_i, z_i) in the Euclidean space.

The formulation for the problem to solve is given below. The objective to maximize is indicated in (2) that represents the density of the placed spheres inside the container. Note that t_i is a binary variable indicating whether sphere s_i is placed $(t_i = 1)$ or not $(t_i = 0)$. Non overlapping constraints are indicated in (3), meaning that the distance between the boundaries of the placed spheres is greater than or equal to zero. Constraints (4)–(6) ensure that each sphere does not exceed the container boundary. For example, the $x-$coordinate x_i of sphere s_i must belong to the interval $[r_i, \ldots, L - r_i]$.

$$\max \frac{4\pi}{3 \times L \times H \times D} \sum_{i=1}^{n} (r_i)^3 t_i \quad (2)$$

s.t.

$$(d_{ij})\, t_i t_j \geq 0 \quad \text{for} \quad 1 \leq i < j \leq n \quad (3)$$

$$r_i \leq x_i \leq L - r_i \quad \text{for} \quad 1 \leq i \leq n \quad (4)$$

$$r_i \leq y_i \leq H - r_i \quad \text{for} \quad 1 \leq i \leq n \quad (5)$$

$$r_i \leq z_i \leq D - r_i \quad \text{for} \quad 1 \leq i \leq n \quad (6)$$

$$t_i \in \{0, 1\} \quad \text{for} \quad 1 \leq i \leq n \quad (7)$$

2.2 A Constructive Heuristic for SPP

A basic algorithm in C&P problems consists in a constructive heuristic that acts like a greedy algorithm. The main steps of this heuristic can be summarized as follows:

- Staring phase: Place the first sphere s_1 at a given place inside the container and compute the possible positions for the other spheres that are not yet placed.
- Iterative phase: At each, step, place the next sphere at the *best* position thanks to a given criterion.
- Final phase: Stop when no additional sphere can be placed.

So constructing an effective heuristic needs to find a good criterion that can evaluates the possible positions for the next sphere to place (see Definition 3). One of the most known criterion in C&P problems is called MHD (Maximum Hole Degree) used for the first time in [5] for packing circles into a rectangular container (so this is a two-dimensional C&P problem) and can easily be adapted to the three-dimensional case. The obtained heuristic is denoted by 3DMHD and is explained below.

For example, Fig. 1 represents a given step of the 3DMHD packing procedure when two spheres (s_1, s_2) are already placed and there are six possible positions where to place sphere s_3. These positions

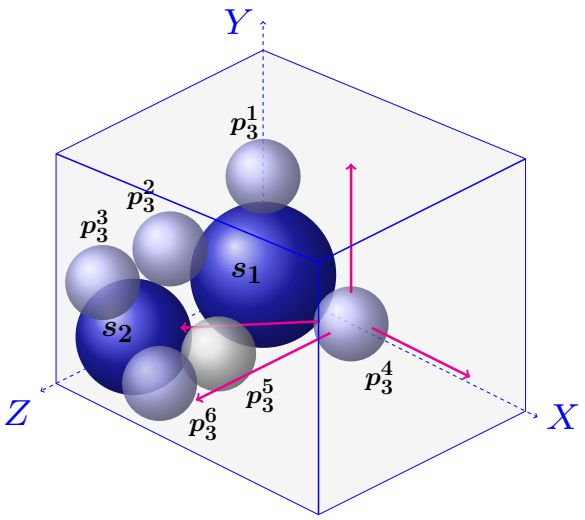

Figure 1: The 3D-MHD heuristic

are denoted by $P = \{p_3^k\}, k = 1, \ldots, 6$. Note that, as indicated in Definition 3, each sphere placed must touch three elements, for example if s_3 is placed at position p_3^2, then it will touch spheres s_1, s_2 and the left–edge of the container. So $T(p_3^2) = \{s_1, s_2, left\}$.

The MHD heuristic uses a measure called *Hole Degree* of a position. So at step i of the packing process, i spheres are already placed, this corresponds to set S_{in}. Let $P_{i+1} = \{p_{i+1}^k\}$ be the set of positions to place the next sphere $s_{i+1} \in S_{out}$. The hole degree of position $p_{i+1}^k \in P_{i+1}$, denoted by $\lambda(p_{i+1}^k)$, is defined as follows:

$$\lambda(p_{i+1}^k) = 1 - \frac{\min_{j \in S_{\text{in}} \cup F \setminus T(p_{i+1}^k)} \left(d_{i+1,j}^k\right)}{r_{i+1}^k} \quad (8)$$

So the hole degree considers the minimum distance between sphere s_{i+1} to place at position p_{i+1}^k of radius r_{i+1}^k and all the other placed spheres (S_{in}) and the six faces of the bin (F) but excluding the elements of $T(p_{i+1}^k)$ since the distance in this case is zero.

MHD (as well as 3DMHD) selects the position $\tilde{p} \in P_{i+1}$ with the maximum hole degree value to place the next sphere s_{i+1} as indicated in (9).

$$\tilde{p} = \underset{p_{i+1}^k \in P_{i+1}}{\arg\max}\ \lambda(p_{i+1}^k) \quad (9)$$

Algorithm 1 summarizes how the constructive 3DMHD heuristic works. The algorithm receives as input sets S_{in}, S_{out} and P. In the general case, S_{in} may contain more than one sphere already placed.

Algorithm 1 The 3DMHD greedy heuristic

Require: Set of already placed spheres S_{in}, S_{out} that contains the spheres that are not yet placed and set P of possible positions for spheres in S_{out}.

Ensure: A feasible packing and its corresponding density.

1: **while** $(P \neq \emptyset)$ **do**
2: Compute/update the hole degree value λ for each corner position $p \in P$;
3: Place the next sphere s_{i+1} at position $\tilde{p}$ that has the maximum hole degree as shown in (9).
4: Move sphere s_{i+1} from S_{out} to S_{in};
5: Update set P by removing positions that overlap the new inserted sphere and compute new positions by using the new inserted sphere and the other objects already placed;
6: **end while**

The output of Algorithm 1 is a feasible solution where a given set of spheres is placed inside the bin and no additional sphere can be placed, i.e., $P = \emptyset$. At each step of the 3DMHD heuristic, we compute or update the hole degree value λ of each position $p \in P$, this is done in Step 2. Step 3 consists then to place the next sphere at the best position $\tilde{p}$. The new placed sphere is then removed from S_{out} and added to S_{in}. We then update the set of positions P (Step 5) by of course removing the positions that overlap the new inserted sphere and by computing new positions by using this sphere. Steps 2–5 are repeated until P becomes empty meaning that no additional sphere can be placed. Note that if all spheres are placed, then the solution obtained is optimal.

In order to compare the quality of two distinct solutions, we use the density of each solution. The density is indicated in Definition (4). The best solution is the solution that corresponds to the highest density.

2.3 The Look-Forward Principle

A greedy heuristic consists to select, at each step, the best choice by using a given evaluation criterion. This method can obtain solutions very quickly but the quality of these solutions is often not sufficient.

A simple way to improve the quality of the solution consists to evaluate several choices or all choices at each step i of the greedy algorithm. This is done by executing the greedy algorithm itself in order to compute a final solution as indicated in Algorithm 1. The choice that obtained the best final density is then chosen in order to move to step $i + 1$. The process is then repeated at each level. This principle is for example

used in [5] in order to improve the result obtained by the MHD heuristic.

3 A New Look-Forward Based Heuristic for SPP

In this work, we propose an enhanced look-forward strategy by adding a new level to the standard look-forward in order to enlarge the search space and then to try to improve the solution quality.

3.1 Importance of the Starting Configuration

It is well-known in Cutting–and–Packing (C&P) problems, that the placement of the first object(s) is very important. One of the first heuristics proposed for C&P problems is Bottom–Left that places the first object at the bottom-left corner of the container. The results indicate also that it is recommended to sort the objects to cut or to pack in decreasing order of their size. Indeed, packing for example the biggest objects increases the chance to reach solutions of better quality.

This principle is adopted in the heuristic proposed in this work. So the spheres in set S are sorted in decreasing order of their radii. But instead of placing the first object (sphere) in the bottom–left–back corner of the bin, we only compute the coordinates of each sphere at each corner (there are eight corners) of the bin. Since these positions are "strategic", we associate the maximum possible value for the hole degree, i.e., $(\lambda = 1)$ for each of these positions. The objective is to allow the packing procedure to place spheres at these positions at the earliest levels of the solution construction.

The proposed strategy acts then like a depth–first search by on only two levels of the search tree. This is then a "partial" depth search.

3.2 Local Search to Improve the Final Solution

In combinatorial optimization, a way to try to improve a computed solution is to use a local search. Local Search (LS) can be applied at each step of the construction of the solution or when the final solution is obtained.

LS methods consist for example to modify a part of the solution in order to escape from local optima. For example, the 2-opt technique is very useful in Vehicle Routing Problems (VRP). It consists to "break" two arcs in the final solution and then rearrange these arcs by connecting the corresponding vertices differently. The objective is to improve the solution quality. LS can be applied to a large variety of optimization problems.

In the heuristic proposed in this paper, we propose to change the last sphere packed inside the bin. The objective is to try to place other additional spheres in order to improve the total volume of the sphere, i.e., the density.

3.3 A Partial Depth-Search Heuristic (PDSH)

This section describes the proposed heuristic for packing spheres into a three-dimensional bin of fixed dimensions. We remember that the objective is to maximize the volume of the packed spheres. The solution quality is then equivalent to the density of the packing, i.e., the sum of volumes of placed spheres divided by the volume of the container. Note of course that the density is a real number that belongs to $[0, 1]$.

Algorithm 2 describes the proposed heuristic. PDSH receives as input parameters the set of spheres $S = \{s_i, i = 1, \ldots, n\}$to pack as well as the dimensions of the bin. PDSH returns the best or optimal solution computed.

In Step 1 of PDSH, the spheres are sorted in decreasing order of their radii values. After that, at step 2, different sets are initialized. Then, S_{out} contains the set of spheres to pack and set S_{in} is equal to the empty set, meaning that, at this step, no sphere is placed inside the bin. Finally, set of positions P is also initialized to the empty set. Step 3 consists to generate the set of all possible positions at the eight corners of the three–dimensional bin. After that, at step 4, the value of the hole degree λ is set to 1 for each position $p \in P$ as described in Section 3.1.

PDSH contains a main loop (the **while** loop) that corresponds to steps 5–23. This loop contains two other nested **for** loops. In the **while** loop, we use a variable denoted by *Density* (step 6) that will indicate the best density obtained during the execution of the two **for** loops. This is why the value of variable *Density* is reset to zero at each time.

The first **for** loop in Algorithm 2 corresponds to the first level of the look–forward search and the second one to the second look–forward level. The first **for** loop consists to apply a look–forward on the current configuration that can be denoted by $\{S_{in}, S_{out}, P\}$. So the algorithm considers each position $p_i \in P$. To do so, a copy $\{S^1_{in}, S^1_{out}, P^1\}$ of the current configuration is created (step 8). The algorithm "forces" the placement of the next sphere at position p_i and the different sets are then updated by moving the new inserted sphere from S^1_{out} to S^1_{in}

Algorithm 2 Heuristic PDSH

Require: Set S of spheres to pack and a three-dimensional bin B of dimensions $L \times H \times D$;

Ensure: A feasible packing and its corresponding density;

1: Sort spheres of S in decreasing order of their radii;
2: Set $S_{out} = S$; $S_{in} = \emptyset$; and $P = \emptyset$;
3: Generate all the possible positions for each sphere $s_i \in S_{in}$ at the eight corners of the bin B, add these positions to set P;
4: Set $\lambda_p = 1$ for each position $p \in P$;
5: **while** $(P \neq \emptyset)$ **do**
6: *Density*=0;
7: **for each** position $p_i \in P$ **do**
8: Set $S_{in}^1 = S_{in}$, $S_{out}^1 = S_{out}$ and $P^1 = P$;
9: Place the next sphere at position p_i and update sets S_{in}^1, S_{out}^1 and P^1;
10: **for each** position $p_j^1 \in P^1$ **do**
11: Set $S_{in}^2 = S_{in}^1$, $S_{out}^2 = S_{out}^1$ and $P^2 = P^1$;
12: Place the next sphere at position p_j^1 and update sets S_{in}^2, S_{out}^2 and P^2;
13: Call 3DMHD(S_{in}^2, S_{out}^2, P^2);
14: Perform a local search on the solution computed by 3DMHD (see Section 3.2);
15: Update the best known solution and value of *Density* if a new best solution is computed;
16: **if** all spheres are placed **then**
17: Stop the algorithm because an optimal solution is obtained;
18: **end if**
19: **end for**
20: **end for**
21: Place the next sphere in the bin at the position $p_i^* \in P$ that led to the best value of *Density*;
22: Update sets S_{in}, S_{out} and P;
23: **end while**

and by updating the set of positions P^1. Just after that, the second level of the look–forward search starts (**for** loop at step 10). Here, we consider each position $p_j^1 \in P^1$. Like in the previous level, a copy $\{S_{in}^2, S_{out}^2, P^2\}$ is created (step 11). We place the next sphere at position p_j^1 and the different sets are then updated (step 12). Now PDSH calls the 3DMHD algorithm (Algorithm 1) in order to compute a final solution by executing the MHD greedy principle until no additional sphere can be placed. 3DMHD returns then to heuristic PDSH the final solution found and its corresponding density. After that, heuristic PDSH performs a local search , at step 14, on the solution returned by 3DMHD (see section 3.2) by removing the last sphere placed in order to try to place other spheres, the objective is to try to increase the final density. PDSH updates then the best solution found is a new better solution is obtained (step 15). If all spheres are placed, then an optimal solution is computed, heuristic PDSH then stops (step 17). If no optimal solution is obtained after performing the two **for** loops, then the next sphere is effectively placed at position p_i^* that led to the best value of *Density* (step 21). Sets S_{in}, S_{out} and P are then updated in order to take into account the new inserted sphere. Finally, heuristic PDSH stops if set P becomes empty. The best solution found so far is then taken as the output of the algorithm.

Table 1: Description of the instances used

Inst.	n	L	H	D
KBG1	30	11.103	10	10
KBG2	30	1.99	10	10
KBG3	30	17.985	10	10
KBG4	30	1.996	10	10
KBG5	30	1.924	10	10
KBG6	30	16.768	10	10
KBG7	50	13.71	10	10
KBG8	50	2.207	10	10
KBG9	50	27.965	10	10
KBG10	50	1.81	10	10
KBG11	50	5.04	10	10
KBG12	50	21.02	10	10

4 Computational Results

In order to study the performance of the proposed heuristic (PDSH), a computational investigation was conducted on a set of 12 instances denoted by $\mathrm{KBG}_i, i = 1, \ldots, 12$ taken from Kubach et al. [7]. The characteristics of these instances are indicated in Table 1. The six first instances KBG1,...,KBG6 contain $n = 30$ spheres and the six other ones KGB7,...,KBG12 contain 50 spheres. Moreover, instances KBG1, KBG2, KBG3, KBG7, KBG8, KBG9 are strongly heterogeneous because the radii of the spheres in each instance are all different. In the six other instances, the spheres are weakly heterogeneous

because the number of spheres of different radii is only 10%, so each radii (sphere) is duplicated ten times. Finally, Table 1 indicates the size of the container where only the length (L) differs from an instance to another one, the height (H) and the depth (D) are all fixed to 10.

Heuristic PDSH was developed in C++ and executed under Linux environment on a computer with a 2.4GHz of frequency and 4 GB of Memory. The results of PDSH can then be directly compared to those obtained by algorithm B1.6 [7] since the computer used in this reference has the same characteristics.

The results of PDSH were compared to those obtained by algorithm B1.6 proposed by Kubach et al. [7]. B1.6 is a direct translation of algorithm B1.5, proposed by Huang et al. [5] to pack circles into a rectangular container, to the three-dimensional case. It is to note that the principle of B1.5, in addition to the use of MHD heuristic and a look–forward strategy, is that it uses a great number of starting configurations that consist to pack two circles in two different corners of the rectangle as well in the bottom-left corner of the container. This lead to a great number of starting configurations (exactly $M = 5 \times n(n-1)/2$ configurations). So this is equivalent to execute the placement procedure M times. Kubach et al. [7] uses exactly the same principle by placing two spheres before calling the packing procedure but they limited the number of starting configurations to 1000.

In the proposed heuristic (PDSH), we generate only positions at the eight corners of the container, no sphere is effectively placed (this is done by the packing procedure). Moreover, we associate a value $\lambda = 1$ for the hole degree of each of these "strategic" positions generated at the initial step.

Table 2 indicates the results obtained by algorithm B1.6 (that has, to our knowledge, the best known results in the literature on these instances) for a computation time limit of one hour (3600 seconds). Column 2 gives then the density (*Dens.*) of the solution obtained by B1.6 and column 3 the corresponding computation time to obtain this solution. Values indicated with a "*" correspond to optimal solutions, i.e., the method succeeded to place all the n spheres inside the container. The results of PDSH are indicated in the three last columns of table 2. Note that PDSH succeeded to reach all the optimal solutions as B1.6 did, but it also succeeded to outperform the results of B1.6 on four other instances (KBG3, KBG6, KBG8 and KBG9). B1.6 is better in only one case (KBG11). The second observation concens the computation time, those obtained by the new heuristic (PDSH) are often lower than those of B1.6. The optimal solution is for example reached in less than 0.01 seconds for instances KBG2, KBG4 and KBG10. For

Table 2: Results obtained by the proposed heuristic PDSH on KBG instances

	B1.6 algorithm		PDSH heuristic		
Inst.	*Dens.*	$t^*(s)$	*Dens.*	$t^*(s)$	Imp.
KBG1	***55.001**	821	***55.001**	96	0.00%
KBG2	***30.071**	<1	***30.071**	$\ll 1$	0.00%
KBG3	52.375	3600	**52.502**	3600	0.24%
KBG4	***37.765**	<1	***37.765**	$\ll 1$	0.00%
KBG5	***48.278**	4	***48.278**	77	0.00%
KBG6	50.022	58	**50.099**	101	0.15%
KBG7	***55.000**	389	***55.000**	3	0.00%
KBG8	46.517	3600	**46.590**	3600	0.16%
KBG9	53.173	3600	**53.210**	3600	0.07%
KBG10	***51.866**	<1	***51.866**	$\ll 1$	0.00%
KBG11	**54.412**	2055	53.651	1374	-1.40%
KBG12	***55.001**	1715	***55.001**	4	0.00%

KBG12, PDSH computes the optimal solution after only 4 seconds (1715 seconds for B1.6). Finally, the last column of the table indicates the improvement obtained by PDSH comparing to B1.6 (in %). Improvement is equal to $100 \times \frac{(\text{density}_{\text{PDSH}} - \text{density}_{\text{B1.6}})}{\text{density}_{\text{B1.6}}}$. PDSH improves the solution or obtained the optimal value in 11 cases out of 12.

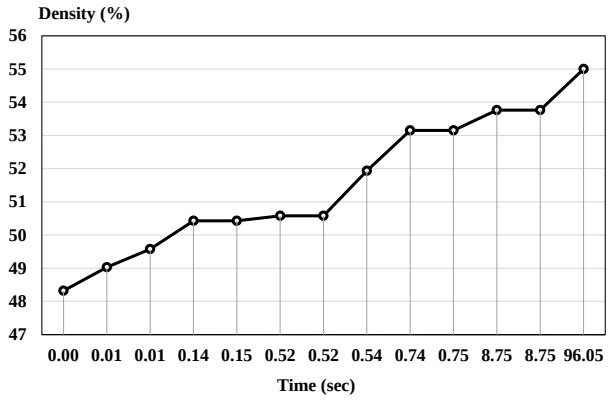

Figure 2: Evolution of the solution quality on the first instance KBG1, optimal density (55.001%) is obtained after 96 seconds.

Figure 2 indicates the evolution of the solution quality in the first instance KBG1 that contains 30 spheres. The density varies from 47% to 55.001% (optimal solution) in 96 seconds.

Figure 3 gives the optimal solution obtained by the proposed heuristic PDSH on the first instance (KBG1).

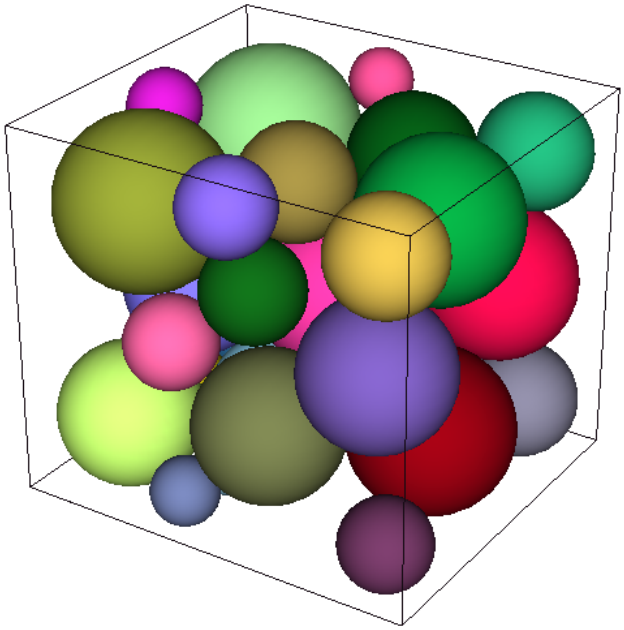

Figure 3: Optimal solution obtained on instance KBG1, $n = 30$, density=55.001%

Finally, figure 4 gives the new best solution obtained by heuristic PDSH on instance KBG6. The density obtained is equal to 52.099%, the old best value is 50.022%.

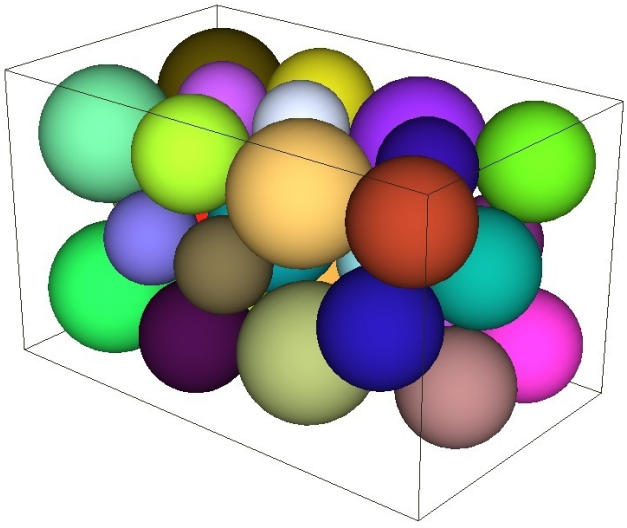

Figure 4: Better solution obtained on instance KBG6, $n = 50$, density=52.099%

5 Conclusion

In this work, the problem of packing spheres into a three–dimensional bin was considered. The proposed heuristic, denoted by PDSH, uses the look–forward principle but introduces a second level in the search tree leading to the enlargement of the search space by exploring more eventualities. This can then be assimilated to a dept-first search with two levels.

The computational investigation, conducted on a set of twelve instances taken from the literature, shows the effectiveness of the proposed method since it reaches all the known optimal values on some instances and succeeds to improve the best known results in four other instances.

As a future work, it will be interesting to apply the new look-forward strategy for other optimization problems.

References:

[1] R. Alvarez-Valdes, A. Martinez and J. M. Tamarit, A branch & bound algorithm for cutting and packing irregularly shaped pieces, *Int. J. Prod. Econ.* 145, 2013, pp. 463–477.

[2] J. A. George, J. M. George and B. W. Lamar, Packing different-sized circles into a rectangular container, *European J. Oper. Res.* 84, 1995, pp. 693–712.

[3] D. He, N. N. Ekere and L. Cai, Computer simulation of random packing of unequal particles. *Phys. Rev. E.* 60, 1999, 7098.

[4] K. Hitti and M. Bernacki, Optimized droping and rolling (ODR) method for packing of polydisperse spheres, *Appl. Math. Model.* 37, 2013, pp. 5715–5722.

[5] W. Q. Huang, Y. Li, H. Akeb and C. M. Li, Greedy algorithms for packing unequal circles into a rectangular container. *J. Oper. Res. Soc.* 56, 2005, pp. 539–548.

[6] Y.–K. Joung and S. D. Noh, Intelligent 3D packing using a grouping algorithm for automative container engineering, *J. Comput. Des. Eng.* 1, 2014, pp. 140–151.

[7] 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.* 28, 2011, pp. 739–753.

[8] Y. Li and W. Ji, Stability and convergence analysis of a dynamics–based collective method for random sphere packing, *J. Comput. Phy.* 250, 2013, pp. 373–387.

[9] C. O. Lópes and J. E. Beasley, A formulation space search heuristic for packing unequal circles in a fixed size circular container, *European J. Oper. Res.* 251, 2016, pp. 64–73.

[10] S. Martello and M. Monaci, Models and algorithm for packing rectangles into the smallest square, *Comput. Oper. Res.* 63, 2015, pp. 161–171.

[11] A. Martinez-Sykora, R. Alvarez-Valdes, J. Bennell and J. M. Tamarit, Constructive procedures to solve 2-dimensional bin packing problems with irregular pieces and guillotine cuts, *Omega* 52, 2015, pp. 15–32.

[12] R. M'Hallah, A. Alkandari and N. Mladenović, Packing unit spheres into the smallest sphere using VNS and NLP. *Comput. Oper. Res.* 40, 2013, pp. 603–615.

[13] K. Soontrapa and Y. Chen, Mono-sized sphere packing algorithm development using optimized Monte Carlo technique. *Ad. Powder Technol.* 24, 2013, pp. 955–961.

[14] W. Visscher and M. Bolsterli, Random packing of equal and unequal spheres in two and three dimensions, *Nature.* 239, 1972, pp. 504–507.

[15] T. Zauner, Application of a force field algorithm for creating stringly correlated multiscale sphere packings. *J. Comput. Phys.* In Press, 2016.

[16] J. Wang, Packing of unequal spheres and automated radiosurgical treatment planning. *J. Comb. Optim.* 3, 1999, pp. 453–463.

[17] Y. Wu, W. Li, M. Goh and R. Souza, Three-dimensional bin packing problem with variable bin height, *European J. Oper. Res.* 202, 2010, pp. 347–355.