A dichotomous search-based heuristic for the three-dimensional sphere packing problem

Mhand Hifi and Labib Yousef

Cogent Engineering (2015), 2: 994257
A dichotomous search-based heuristic for the three-dimensional sphere packing problem

Mhand Hifi and Labib Yousef

Abstract: In this paper, the three-dimensional sphere packing problem is solved by using a dichotomous search-based heuristic. An instance of the problem is defined by a set of $n$ unequal spheres and an object of fixed width and height and, unlimited length. Each sphere is characterized by its radius and the aim of the problem is to optimize the length of the object containing all spheres without overlapping. The proposed method is based upon beam search, in which three complementary phases are combined: (i) a greedy selection phase which determines a series of eligible search subspace, (ii) a truncated tree search, using a width-beam search, that explores some promising paths, and (iii) a dichotomous search that diversifies the search. The performance of the proposed method is evaluated on benchmark instances taken from the literature where its obtained results are compared to those reached by some recent methods of the literature. The proposed method is competitive and it yields promising results.

1. Introduction
Cutting and Packing problems (CP) is a family of natural combinatorial optimization problems, admitted in numerous real-world applications from industrial engineering, logistics, manufacturing, production process, automated planning, etc. One of the more recent papers addressing an optimization with a packing problem is due to Sutou and Dai (2002), where the unequal sphere problem has been used for tackling an application of the automated radiosurgical treatment planning. Wang (1999) has also considered sphere packing problems as an optimization tool for the radiosurgical treatment planning.
Other problems of the CP family have been described and redefined in Wascher, Haussner, and Schumann (2007) 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, …).

This paper deals with the problem of packing spheres (called items) in a container (parallelepiped or object) with unlimited length. The problem is known as the Three-Dimensional Sphere Packing Problem (3DSPP). An instance of 3DSPP is defined by a set $N$ of $n$ unequal items and an object $P$ of fixed width $W$ and height $H$ and, unlimited length (for the rest of the paper, the length representing the variation of $P$’s length is denoted $L$). Moreover, each item $i \in N = \{1, \ldots , n\}$ is characterized by its radius $r_i$. The goal of the problem is to optimize the length $L$ of the object $P$ such that all items of $N$ are packed (positioned) in the target object, without overlapping.

The 3DSPP may be formulated as follows:

$$\min L$$  \hspace{1cm} (1)

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \geq (r_i + r_j)^2 \quad \forall (i, j) \in N^2, i < j$$  \hspace{1cm} (2)

$$r_i \leq x_i \leq L - r_i \quad \forall i \in N$$  \hspace{1cm} (3)

$$r_i \leq y_i \leq H - r_i \quad \forall i \in N$$  \hspace{1cm} (4)

$$r_i \leq z_i \leq W - r_i \quad \forall i \in N$$  \hspace{1cm} (5)

$$L \leq L \leq \bar{L}$$  \hspace{1cm} (6)

where the objective function (1) minimizes the length of the object $P$ containing all the items of $N$, Equation (2) ensures the nonoverlap constraint of any pair of distinct items $(i, j)$ of $N$; that is, the distance between the centers of both items which must be greater than or equal to the sum of their radii and, Equations (3)–(5) ensure that all items of $N$ belong to the target object $P$ of dimensions $(L, W, H)$. Note that since the aim of the problem is to find the smallest length of the object containing all items of $N$, then it is easy to start any method trying to solve the above problem by a trivial value representing the sum of the spheres’ area affected to $L$ (Equation (6)) and a quick (feasible) solution value with a greedy algorithm affected to $\bar{L}$.

The rest of the paper is organized as follows. Section 2 gives a literature review for the 3DSPP and its variants. The problem representation is discussed in Section 3.1. Section 3.2 exposes the principle of the greedy selection phase which serves to determine a subset of eligible positions employed for packing the predefined items in the target object. Section 3.3 presents a truncated tree search using a width beam search phase that serves to explore some promising paths. Section 3.3.2 exposes the dichotomous search where the first two phases are embedded in an interval search in order to diversify the search process. Section 4 proposes an experimental part in which the performance of the proposed algorithm is given: its provided results are compared to those reached by some best algorithms of the literature. Finally, Section 5 concludes by summarizing the contribution of this paper.

2. Related work

The 3DSPP, belonging to Cutting and Packing (CP) family (Wascher et al. (2007)), has received very little attention. Most published papers focus on packing (or cutting) items in two-dimensions (cf. Birgin & Sobral, 2008; Farr, 2013; Hifi, M’Hallah, and Saadi, 2008; Kubach, Bortfeldt, Tilli, & Gehring, 2009; Stoyan, Yaskow, & Scheithauer, 2003) but very few papers addressing the problem of packing the sphere packing are available.
Among existing papers addressing sphere packing problems, we cite Lochmann, Oger, and Stoyan (2006) who proposed a statistical analysis for packing random spheres with variable radius distribution. Li and Ji (2013) 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 (2013) studied the problem of random close packing fractions of lognormal distributions of hard spheres. The author tailored a one-directional approach in order to predict a close packing of spheres of lognormal distributions of sphere sizes.

Packing spheres into a container has been addressed by Sutou and Dai (2002) who proposed a global optimization approach. Stoyan et al. (2003) 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, Alkandari, and Mladenovic (2013) 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 programs. M’Hallah and Alkandari (2012) considered the principle used in M’Hallah et al. (2013) to solve the problem of packing identical spheres into the smallest containing cube. Soontrapa and Chen (2013) tackled the problem of packing identical spheres into a smallest containing sphere by using a random search according to Monte Carlo’s method. Finally, Birgin and Sobral (2008) proposed twice-differentiable nonlinear 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, ALGENCE solver was used for generating a multiple starts solutions (an extensive efficient models and methods for packing both circular and sphere problems were reviewed in Hifi & M’Hallah, 2009b).

In this paper, we propose a three phase solution procedure that combines the width-beam search and a greedy selection procedure for approximately solving the 3DSPP. The last two phases are embedded in a dichotomous procedure that is used for searching the best length \( L^* \) for the object \( \mathbb{P} \). Note that the proposed algorithm can be viewed as an adaptation of the method proposed in Hifi et al. (2008) for solving the two-staged two-dimensional cutting problem and later in Akeb and Hifi (2008) for solving another variant of CP family. The novelty in this approach is based upon (i) a new representation used for defining a restricted space search; that is, a structured set based on target eligible positions which serves to limit the search process without degrading the quality of solutions (Sections 3.1 and 3.2) and (ii) a nice dichotomous search (Section 3.4) that tries to avoid premature convergence toward local stagnant solutions whenever the width-beam search is applied (Section 3.3).

3. Tackling 3DSPPP with a dichotomous search
Herein, the problem representation and strategies used by the proposed algorithm are discussed. First, Section 3.1 describes how an item can be represented in a space and which eligible positions may be chosen for assigning items of \( N \) to the target object \( \mathbb{P} \). Second, Section 3.2 describes the principle of the simple constructive procedure which tries to guaranty a feasible packing of all items of \( N \) for the 3DSPPP according to the set of eligible positions. Such a simple constructive procedure is based on the minimum distance between items and the edges of the objet \( \mathbb{P} \) that serves to create a set of eligible positions for the current item to pack. Third and last, Section 3.3 discusses the principle of the proposed algorithm and its main steps.

3.1. Representation of 3DSPPP
In this section, we present the principle of the local strategy used for filtering the huge number of possible positions when an item will be assigned to the object \( \mathbb{P} \). The local strategy is based on the simple Greedy Principle (called GP) where the minimum distance position is favored for packing a predefined item. GP is then used as an evaluation operator for finding a subset of eligible positions of the next item to pack. Furthermore, the GP can also be used either as a complete solution procedure that provides a final feasible solution for the 3DSPPP or as a greedy strategy in order to complete a partial solution.
For the rest of the paper, the following notations are introduced:

- The bottom-left-depth corner of $P$ is positioned at $(0,0,0)$ and $P$ is characterized by a set formed with six labels (namely faces): $F = \{\text{left, top, right, bottom, depth, front}\}$. Then, $P$ is represented in the Euclidian space, as illustrated in Figure 1.
- The center of the $i$th item belonging to $N$ is positioned at $(x_i, y_i, z_i)$.
- The distance, namely $\delta_{ij}$, between two different items $i$ and $j$ belonging to $N$ is computed as follows:
  \[
  \delta_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 - (r_i + r_j)}, \quad \forall (i,j) \in N^2
  \] (7)
Observe that positioning any item $i$ of $N$ into $P$ induces to define a distance between an item belonging to $N$ and faces of $F$. Indeed, for instance, assigning an item $i \in N$ to an eligible position of $P$ while respecting the nonoverlap of the left-face of $P$ requires to satisfy the following distance: $\delta_{i,\text{left}} = x_i - r_i$. For all faces of $P$, Table 1 reports the distance to be satisfied whenever a preselected item $i$ of $N$ is assigned to an eligible position of $P$.

### 3.2. 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 using some selection criteria in order to provide either partial or final solutions for the problem to solve. Herein, we consider a simple GP which is based on searching the position realizing the minimum distance position between items and facets. In fact, GP is used as a selection criterion for defining a set of eligible positions to assign to the predefined item $i$ (not already positioned) among all eligible positions representing the set $P_i$.

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

| $f$ | $\delta_{ij} \mid i \in N, f \in F$ |
|-----|---------------------------------|
| 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 | $W - z_i - r_i$                |
I, denotes the set of items of N already positioned in the current object P.

T, contains the items of N which are not yet assigned to P.

P, denotes the set of distinct eligible positions for the next item i to pack given the set of packed items I.

An eligible position p+, ∈ P, (for the item i) is determined by using three elements e, e, and e, where an element is either an item of N already positioned (representing the set I) or one of the six faces belonging to F.

T, represents the set composed of the three elements e, e, and e.

Figure 1 illustrates the mechanism used by the GP strategy on a small example. We assume that the first four items are already positioned in the object P; then, there are three eligible positions that emerge for the next item 5 to pack. Following the above notations, I = {1, 2, 3, 4} and P, = {p, j = 1, ..., 3}. First, the position p touches both items 1 and 3 and, the “bottom” face of P. Second, the position p is obtained by using the item 2 and both faces “left” and “bottom” of P. Third and last, the position p is computed by using the item 1 and both faces “left” and “depth” of P. Finally, it follows that T = {1, 2, bottom}, T = {2, left, bottom} and T = {1, left, depth}.

We can observe that the item 4 is positioned around the three items 1–3 that means T = {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 P that gives T = {1, 3, bottom}. We recall that the objective of the problem is to minimize the length of the target object P. It means that the right face of P can be omitted and only the five faces can be considered when optimizing the length of the target P. Hence, all eligible positions may be obtained by using the fifth faces, the positioned items and the current item to pack.

Moreover, GP works as follows: the value corresponding to the (i + 1)th item to pack, when positioned at the eligible position p+, ∈ P, is computed as follows:

\[
\Delta(p,+) = \min_{j \in I, \cup F \setminus \{\text{right}\}} \delta_{(i+1,j)}
\]

where \(F' = F \setminus \{\text{right}\}\).

Finally, when GP is used, then it starts by positioning the first item i = 1 at the bottom-left-depth position, i.e. at the position (1, 2, left, depth), while the remaining n − 1 items are successively positioned according to the minimum distance rule (cf. Equation (8)). As illustrated in Figure 1, the item 5 will be placed at position p because its corresponding distance realizes the minimum value (of course, this choice may also be modified along the execution of the algorithm).

Of course, one can use GP as a greedy procedure for searching a feasible solution for the 3DSPP. Indeed, by setting the starting length of the target object P to ∞ (i.e. \(L_p = \infty\)) and by applying successively GP on the rest of the nonpacked items, GP terminates by building a feasible solution with length \(L_p\). The provided length \(L_p\) can also be considered as an upper bound for starting the proposed algorithm.

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

In this section, we first describe the principle of the standard beam search (Section 3.3.1). Second and last, we show how beam search can be adapted for solving the 3DSPP (Section 3.3.2), especially when using a width-beam almost of the standard beam.

3.3.1. A standard beam search

Beam Search (BS) has been first proposed in Ow and Morton (1988) 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, Ghirardi, & Tadei, 2004; Hifi & M’Hallah, 2009a;
Hifi et al., 2008; Hifi & Saadi, 2010; Yavuza, 2013). 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.

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 \( \omega \) that is called the beam width. Generally, these selected \( \omega \) 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.

Algorithm 1 describes the main steps of the standard beam search which is characterized by a beam width \( \omega \) used for filtering the set of offspring nodes \( B_\omega \). A node corresponds to a partial feasible solution and the set \( B \) of current nodes is initialized to the root node \( B_0 \) whereas \( B_\omega \) containing the offspring nodes is initialized to the empty set. Each node of \( B \) generates a set of offspring nodes, and moves them to \( B_\omega \). If a node \( \eta \) of \( B_\omega \) is a leaf (i.e. no further branching is possible out of \( \eta \)), then its objective function value \( z_\eta \) is computed and compared to \( z^* \). If \( z_\eta > z^* \), then the incumbent solution is set to the leaf node; \( z^* \) is then updated: \( z^* = z_\eta \); and \( \eta \) is removed from \( B_\omega \). All nodes belonging to \( B_\omega \) are assessed using an evaluation function, and then ranked in decreasing order of their values. The \( \min(\omega, |B_\omega|) \) best nodes (called elite nodes) of \( B_\omega \) are appended to the set \( B \), whereas the remaining nodes of \( B_\omega \) are discarded and \( B_\omega \) is reset to the empty set. This process is iterated until no further branching is possible, i.e. until \( B = \emptyset \).

3.3.2. Adaptation of the beam search for the 3DSPP
Applying BS to 3DSPP requires defining the nodes of the tree and the branching mechanism out of the nodes of \( B \). Herein, a node \( \eta \) is represented by the pair of subsets:

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

The first subset represents a partial solution realized when assigning all the items to the object \( P \) and the second subset serves to provide a complementary solution. Such a complementary solution is computed by applying the MLDP used as greedy procedure (as discussed in Section 3.2).
Moreover, branching out of a node $\eta_i$ is equivalent to create at most $|P_i|$ branches emanating out of the current node (related to the eligible positions as described in Section 3.2). 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 $N$.

As mentioned above, because of the huge number of feasible positions that a predefined item may generate, we preferred the use of the width-beam search almost of the standard beam search. Indeed, as we shall see in Section 3.4, the proposed algorithm uses a dichotomous search according to a series of values assigned to $\omega$ (the beam width). Therefore, in order to try the exploration of a maximum number of promising paths, the beam search applies the width search where all nodes belonging to the same level are simultaneously evaluated following an estimator operator and only the best ones are selected for the rest of the search.

---

**Algorithm 2**. Adaptation of the Beam Search for the 3DSSP: BS

**Input.** A node $\eta^\ell$.

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

1: **Initialization Step.**
2: Let $\omega$ be a predefined beam width.
3: Set $B = \{\eta^\ell\}$, where $\eta^\ell$ denotes the input node associated to the $\ell$-th level.
4: Set the variable feasible to false /* no feasible solution at hand */
5: **Iterative Step.**
6: while $((B \neq \emptyset)$ and (the runtime limit is not performed)) do
7: Branch from the current level $\ell$ by selecting the $\omega$ eligible positions for each node $\eta_i \in B$;
8: Insert all obtained nodes into $B_{\ell}$;
9: Evaluate the potential of each node belonging to $B_{\omega}$ using GP for completing the path.
10: If a feasible solution is given by GP, then set feasible to true and exit;
11: Replace $B$ by the best $\omega$ nodes of $B_{\omega}$ realizing highest densities and increment the level $\ell$.
12: end while

The aforementioned process is described by Algorithm 2. The adapted algorithm works according to a given node, namely $\eta^\ell$, taken at the level $\ell$ of the developed tree. Thereafter, the initialization step (Lines 1–3) is applied for starting the set $B$ containing the best provided nodes regarding the starting node $\eta^\ell$ and the initialization of the variable feasible to false. This variable controls the (un) feasibility of the series of the solutions built. The main loop (lines 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 3.2). Second, all created nodes are stored in a provisional set $B_{\omega}$, 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 3.2). Thereafter, for each final solution (either feasible or unfeasible for the target object $P$), the potential of a node $\eta \in B_{\omega}$ is represented by the density of the positioned items in $P$. Whenever one of these constructed solutions provides a feasible solution (line 10), i.e. all items are positioned in the target object $P$, 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 11) by the $\omega$ nodes which realizing the highest densities and the current level of the developed tree is incremented. Finally, the process is iterated until either $B$ is reduced to an empty set or when the fixed runtime limit is performed.

3.4. Using a dichotomous search

The proposed algorithm applies a series of BS for a predefined interval search. First, the interval search starts by $[L, \bar{L}]$, where $L$ denotes a lower bound for the 3DSSP and $\bar{L}$ its upper bound (in the case of a feasible solution exists, its objective value is assigned to $L$). Second, for each fixed interval, BS tries to construct the best feasible solution by packing all the items into the current target object; that is, $(L^*, W, H)$, where $L^* \in [L, \bar{L}]$. 

---
The main steps of the dichotomous search are summarized in Algorithm 3. First, its starts by defining the initial interval \([L, U]\) where the upper bound \(U\) is obtained by applying GP as a heuristic on the open object, i.e. \((\infty, W, H)\). The main loop repeat (cf. lines 7–12) of the dichotomous procedure serves to explore a series of neighborhoods depending on the values of \(\omega\).

4. Computational results

In this section, we investigate the effectiveness of the width beam search-based heuristic (DSBH) on two sets of benchmark instances: Set1 and Set2.

The first set “Set1” contains six instances (SYS1,...,SYS6) extracted from Stoyan et al. (2003), where the number of predefined items varies from 25 to 60. These instances have been already tested using Stoyan et al.’s (2003), Birgin and Sobral’s (2008) and Kubach, Bortfeldt, Tilli, and Gehring (2011) approaches.

The second set “Set2” contains six instances (KBTG1, KBTG2, KBTG3, KBTG7, KBTG8, and KBTG9) taken from Kubach et al. (2011). For each instance, both dimensions \(W\) and \(H\) of the object are fixed to 10 whereas the number of 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. (2011) where they represent the six instances with unequal spheres.

The rest of the experimental part is organized as follows. First, the behavior of the proposed algorithm DSBH is evaluated on the first set of instances Set1 (Section 4.1). The results reached by DSBH are thereafter compared to those reached by four heuristics: Stoyan et al.’s (2003) method-based heuristic (noted SYS), Birgin and Sobral’s (2008) method-based heuristic and both sequential and parallel heuristics proposed by Kubach et al.’s (2009, 2011) (noted KBTG\(_s\) and KBTG\(_p\), respectively).
Second, for the instances of Set2 (Section 4.2), the results provided by DSBH are compared to those reached by Kubach et al. (2011). Note also that 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.

4.1. Performance of DSBH vs. three 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 3.3.2, DSBH uses two parameters: the beam width \( \omega \) and the maximum runtime limit to fix. Our computational study was conducted by varying \( \omega \) in the discrete interval \( \{5,6,7,\ldots\} \) and the maximum runtime limit was fixed to 3600 s (which can be considered as a standard runtime limit considered by algorithms of the literature). Of course, the upper value of \( \omega \) depends on the limited runtime and the size of the instance. In order to show the effect of these parameters, we discuss the quality of the solutions obtained by DSBH when the beam width \( \omega \) increases.

First, we do it on an instance extracted from Stoyan et al. (2003) by varying the value of \( \omega \) from 1 (the minimum value assigned to \( \omega \) in Algorithm 3) to the upper value (incrementing \( \omega \) with one unit) till attaining the fixed runtime limit. Figure 2(a) shows the behavior of DSBH when varying the beam width \( \omega \) for the instance SYS5. From the curve labeled Sol., one can observe that the value of the length \( \ell^* \) of the target object \( P \) oscillates but globally its average solution value decreases and converges toward one of the best solutions (generation of a series of local optima). This is the reason why we proposed to reiterate the GP rule in order to get the final solution realizing the best local optimum. On the other hand, according to the second curve labeled Best, one can observe that the behavior of DSBH provides a series of decreased solution values; in this case, only the best solution was retained for the whole values of \( \omega \). For SYS5, the best local optimum reached by DSBH realizes the value closest to 10.94 and its solution configuration is illustrated by Figure 2(b). In fact, the provided solution is equal to 10.9359 (as shown later in the second part of the experimental part) and it dominates both values 11.2170 and 10.9359 reached by the sequential KBTG algorithm (Kubach et al., 2011) and its parallel version (Kubach et al., 2009) representing the best solution values of the literature. Note that \( \omega = 12 \) seems to give the best solution value that balances between the quality of the solution and the fixed runtime since for higher values of \( \omega \) the solution remains unchanged.

Second, for the instances of Set1, Table 3 compares the results of DSBH to those reached by four algorithms: SYS (cf. Stoyan et al., 2003), BSA (Birgin & Sobral, 2008), KBTG \({}_s\) (Kubach et al., 2011) and its parallel version noted KBTG \({}_p\) (proposed in Kubach et al., 2009), where the known solutions of the literature are taken from Kubach et al., 2009, 2011.

Columns from 1 to 4 show the instance label, problem size \( n \) and both height \( H \) and width \( W \) of \( P \). Column 5 displays the solution value \( \ell^*_\text{SYS} \) reached by STS whereas column 6 displays BSA’s solution values (noted \( \ell^*_\text{BSA} \)). Columns 7 and 8 report the best solutions (noted \( \ell^*_\text{KBTG} \)) provided by KBTG algorithm for both fixed runtimes: 300 and 3600 seconds. Column 8 shows the best solutions reached by
the parallel KBTG version (noted $L_{KBTG}^*$) without fixing the runtime limit. Columns 9 and 10 display the solution (noted $L^*$) realized by the proposed algorithm DSBH for both fixed runtimes (300 and 3600 s, respectively). Finally, column 11 reports the best value of $\rho$ for which its best solution for the second runtime limit is performed.

All results of Table 2 are summarized in Table 3, where it tallies the percentage improvement (when it happens) yielded by DSBH when compared to the results reached by the four considered algorithms. Table 3 is composed of two parts: the first part contains the informations about the instance and the second part shows the percentage improvement realized by DSBH over the four heuristics tested (noted $\%SYS$, $\%BSA$, $\%KBTG_s$ and $\%KBTG_p$ according to the heuristics SYS, BSA, KBTG$_s$ and KBTG$_p$, respectively) especially for the second runtime limit for DSBH.

The analysis of the results of both Tables 2 and 3 follows:

1. First, DSBH outperforms the three algorithms SYS, BSA, and KBTG$_s$ whenever the runtime limit was fixed to 300 s. Indeed, it is able to reach the best solution for all instances of Set1.
2. Second, DSBH outperforms SYS, BSA and both KBTG$_s$ and KBTG$_p$ algorithms since for the instances of Set1 it is able to provide better results.
3. Third, when comparing DSBH’s results to SYS’s ones, one can observe that the percentage of the improvement varies from 6.49% (instance SYS5) to 8.82% (instance SYS3). This percentage improvement remains interesting when comparing DSBH’s results to those reached by BSA: in this case, such improvement varies between 5.96% (instance SYS1) and 7.65% (instance SYS6). Moreover, it stills interesting when comparing DSBH’s results to those provided by KBTG algorithm. Indeed, the improvement varies from 0.48% (instance SYS1) to 6.26% (SYS4).

### Table 2. Performance of BSBH versus the four heuristics of the literature on instances of Set1. The symbol “•” (resp. “⋄”) means that the value for this instance is not available (resp. corresponds to the best solution)

| Label | n | H  | W  | $L_{SYS}^*$ | $L_{BSA}^*$ | $L_{KBTG_s}^*$ | $L_{KBTG_p}^*$ | $L^*$ | $L^*$ | $\rho^*$ |
|-------|---|----|----|------------|------------|---------------|---------------|------|------|--------|
| SYS1  | 25| 5.5| 6.9| 9.912      | 9.7942     | 9.5397        | 9.2656        | 9.2635| 9.2431|        |
| SYS2  | 35| 6.5| 7.9| 9.623      | -          | 9.2608        | 8.9301        | 9.1585| 8.9164|        |
| SYS3  | 40| 5.5| 6.9| 9.473      | 9.3090     | 9.1794        | 8.7178        | 8.9085| 8.7056|        |
| SYS4  | 45| 8.5| 9.9| 11.086     | 11.0962    | 10.9991       | 10.8760       | 10.9224| 10.2357|        |
| SYS5  | 50| 8.5| 9.9| 11.646     | 11.6211    | 11.4877       | 11.3494       | 11.9865| 11.1772|        |
| SYS6  | 60| 8.5| 9.9| 12.842     | 12.7215    | 12.7110       | 12.3745       | 11.8399| 12.4149|        |
| Average|  |    |    | 11.029     | 11.4877    | 11.0962       | 10.8760       | 10.9224| 10.2357|        |

### Table 3. Percentage improvements between all tested heuristics: BSBH, SYS, BSA and both KBTG$_s$ and KBTG$_p$ on instances of Set1

| Label | n | H  | W  | $\%SYS$ | $\%BSA$ | $\%KBTG_s$ | $\%KBTG_p$ |
|-------|---|----|----|---------|---------|------------|------------|
| SYS1  | 25| 5.5| 6.9| 7.24    | 5.96    | 0.48       | 0.24       |
| SYS2  | 35| 6.5| 7.9| 7.92    | -       | 2.37       | 0.15       |
| SYS3  | 40| 5.5| 6.9| 8.82    | 6.93    | 3.21       | 0.14       |
| SYS4  | 45| 8.5| 9.9| 8.31    | 8.41    | 6.26       | 1.65       |
| SYS5  | 50| 8.5| 9.9| 6.49    | 6.27    | 3.78       | 0.46       |
| SYS6  | 60| 8.5| 9.9| 8.67    | 7.65    | 4.71       | 0.19       |
| Average|  |    |    | 7.91    | 7.04    | 3.47       | 0.47       |
Fourth and last, DSBH’s results remain better than those reached by the parallel algorithm KBTG_p (we recall that the parallel algorithm ran without runtime limit). Indeed, all the solution values for the instances of Set1 are improved. In this case, the percentage of improvement varies from 0.14% (instance SYS3) to 1.65% (instance SYS4), which remains very interesting for a sequential algorithm.

Figure 3 shows the behavior of DSBH on the instances of Set1 where each curve represents the variation of the improvements realized according to the algorithm SYS, BSA and both KBTG_s and KBTG_p respectively. Figure 4(a) illustrates GP’s configuration (when applied as a heuristic for providing a starting solution for DSFBG) and Figure 4(b) displays DSBH’s configuration. From the starting configuration (cf. Figure 4(a)), one can observe the failure of GP when used as a heuristic: in this case, all items are concentrated on the bottom-left-depth position and all positions located along the length of P to the top are omitted. However, the configuration performed by DSBH shows the correction made when GP’s process is repeated.

4.2. Performance of DSBH versus KBTG heuristic: Set2
This section compares the results reached by DSBH to those reached by KBTG_s (note that, for this type of instances, KBTG_s realizes the best solution values of the literature). This comparison is performed on the instances of the second group Set2 extracted from Kubach et al. (2011). For this type of instances, instead of determining the minimum length $L^\star$ of the target container $P$, Kubach et al. (2011) computed the density of all packed items in the final object $P$. Therefore, in addition to the density, the best length $L^\star$ of the final object $P$ corresponding to that density is reported in the same table.

The results realized by the two tested methods (DSBH and KBTG_s, respectively) are reported in Table 4. Columns 1–4 display all the characteristics related to the instance. Column 5 reports the solution value (expressed in term of density) realized by KBTG_s’s algorithm (extracted from Kubach et al. (2009, 2011)). The next three columns report the best length $L^\star$ reached by DSBH, its corresponding density and the best value of $\alpha$ for which the best solution is reached. Finally, the last column displays the percentage improvement realized by DSBH according to the solution values reached by KBTG_s. The analysis of the results of Table 4 follows.
(1) DSBH remains competitive since it improves most solutions reached by KBTG. 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 KBTG.

(2) For the improved solutions, DSBH realizes an improvement varying from 3.42% (instance KBTG1) to 4.68% (instance KBTG8). Globally, the average improvement over all instances is equal to 3.63%.

(3) In term of density, DSBH realizes a density of 49.932% which is much better than those realized by KBTG (48.120%), as illustrated in Figure 5.

Finally, Figure 6(a) illustrates the starting solution reached by iterating GP till packing all the items, whereas Figure 6(b) shows the final solution when DSBH is applied. We can observe that the starting configuration leaves also an important unoccupied area along the length of the object P to the depth. However, the configuration leads a better packing when GP rule is repeated on different eligible positions of the search space.

### Table 4. Performance of BSBH vs. KBTG on instances of Set2

| Label | #Inst. | Dichotomous search-based heuristic |
|-------|--------|-----------------------------------|
| KBTG1 | 30     | H | W | \(d^*_{\text{KBTG}}\) | \(L^*\) | \(d^*\) | \(\omega^*\) | %KBTG |
| KBTG1 | 30     | 10 | 10 | 56.096 | 10.903 | 56.0092 | 18 | 3.42 |
| KBTG2 | 30     | 10 | 10 | 30.071 | 1.990 | 30.071 | 23 | 0.00 |
| KBTG3 | 30     | 10 | 10 | 51.387 | 18.241 | 53.6243 | 21 | 4.17 |
| KBTG7 | 50     | 10 | 10 | 55.372 | 13.099 | 57.5662 | 17 | 3.81 |
| KBTG8 | 50     | 10 | 10 | 45.060 | 2.582 | 47.004 | 15 | 4.14 |
| KBTG9 | 50     | 10 | 10 | 52.732 | 27.803 | 55.3203 | 13 | 4.68 |
| Average | 48.120 | 49.932 | 3.63 |

Figure 5. Variation of the percentage of the density realized by both DSBH and KBTG on the instances of Set2.

Figure 6. Illustration of DSBH’s behavior on KBTG7 instance (Set2): (a) the starting solution with a length equal to 15.3817 and (b) the best solution’s structure and its length (13.0997) reached by DSBH.
5. Conclusion
In this paper, the three-dimensional sphere packing problem was solved with a dichotomous search-based heuristic. The proposed method is an adaptation of the beam search which combines the following components: (i) a greedy selection phase for performing a local restricted space search that contains eligible positions, (ii) a standard width-beam search phase for exploring some promising paths, and (iii) a dichotomous search for diversifying the search space. Through extensive testing on different instances of the literature, the proposed method is able to improve a significant number of the solutions provided by recent methods available in the literature.

Acknowledgements
The authors thank the anonymous referees for their helpful comments and suggestions which contributed to the improvement of the contents of this paper.

Funding
The authors received no direct funding for this research.

Author details
Mhand Hifi1
E-mail: hifi@u-picardie.fr
Labib Yousef1
E-mail: labib.yousef@etud.u-picardie.fr
1 Laboratory EPROAD – EA 4669 (ROAD Team), UFR des Sciences, University of Picardy Jules Verne, 7 rue du Moulin Neuf, 80000 Amiens, France.

Citation information
Cite this article as: A dichotomous search-based heuristic for the three-dimensional sphere packing problem, M. Hifi & L. Yousef, Cogent Engineering (2015), 2: 994257.

References
Akeb, H., & Hifi, M. (2008). Algorithms for the circular two-dimensional open dimension problem. International Transactions in Operational Research, 15, 685–704.
Birgin, E. G., & Sobral, F. N. C. (2008). Minimizing the object dimensions in circle and sphere packing problems. Computers & Operations Research, 35, 2357–2373.
Della Croce, F., Gharardi, M., & Tadei, R. (2006). Recovering beam search approach for combinatorial optimization problems. Journal of Heuristics, 10, 89–104.
Farr, R. S. (2013). Random close packing fractions of lognormal distributions of hard spheres. Powder Technology, 245, 28–34.
Hifi, M., & M’Hallah, R. (2009a). Beam search and non-linear programming tools for the circular packing problem. International Journal of Mathematics in Operational Research, 1, 476–503.
Hifi, M., & M’Hallah, R. (2009b). A literature review on circle and sphere packing problems: models and methodologies. Advances in Operations Research, 22 pp. doi:10.1155/2009/150624. Article ID 150624.
Hifi, M., M’Hallah, R., & Saadi, T. (2008). Algorithms for the constrained two-staged two-dimensional cutting problem. INFORMS Journal on Computing, 20, 212–221.
Hifi, M., & Saadi, T. (2010). A cooperative algorithm for constrained two-staged two-dimensional cutting problems. International Journal of Mathematics in Operational Research, 9, 104–124.
Kubach, T., Bortfeldt, A., Tilli, T. & Gehring, H. (2009). Parallel greedy algorithms for packing unequal spheres into a cuboidal strip or a cuboid. (Working Paper No 440). Hagen: Department of Management Science, University of Magdeburg, (Diskussionsbeitrag der Fakultät für Wirtschaftswissenschaft der FernUniversität in Hagen).
Kubach, T., Bortfeldt, A., Tilli, T., & Gehring, H. (2011). Greedy algorithms for packing unequal sphere into a cuboidal strip or a cuboid. Asia-Pacific Journal of Operational Research, 28, 739–753.
Li, Y., & Ji, W. (2013). Stability and convergence analysis of a dynamics-based collective method for random sphere packing. Journal of Computational Physics, 250, 373–387.
Lochmann, K., Oger, L., & Stoyan, D. (2006). Statistical analysis of random sphere packings with variable radius distribution. Solid State Sciences, 8, 1397–1413.
M’Hallah, R., & Alkandari, A. (2012). Packing unit spheres into a cube using VNS. Electronic Notes in Discrete Mathematics, 39, 201–208.
M’Hallah, R., Alkandari, A., & Mladenovic, N. (2013). Packing unit spheres into the smallest sphere using VNS and NLP. Computers & Operations Research, 40, 603–615.
Ow, P. S., & Morton, T. E. (1988). Filtered beam search in scheduling. International Journal of Production Research, 26, 297–307.
Soontrapa, K., & Chen, Y. (2013). Mono-sized sphere packing algorithm development using optimized Monte Carlo technique. Advanced Powder Technology, 24, 955–961.
Stoyan, Y., Yaskow, G., & Scheithauer, G. (2003). Packing various radii solid spheres into a parallelepiped. Central European Journal of Operational Research, 11, 389–407.
Sutou, S., & Dai, Y. (2002). Global optimization approach to unequal sphere packing problems in 3D. Journal of Optimization Theory and Applications, 114, 671–694.
Wang, J. (1999). Packing of unequal spheres and automated radiosurgical treatment planning. Journal of Combinatorial Optimization, 3, 453–463.
Wascher, G., Haussner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. European Journal of Operational Research, 183, 1109–1130.
Yauzuo, M. (2013). Iterated beam search for the combined car sequencing and level scheduling problem. International Journal of Production Research, 51, 3698–3718.
