A Look-Forward Exploring Strategy for the Sphere Packing Problem

H. Akeb
ISC Paris, Paris, France

***Abstract*—Cutting and Packing (C&P) problems are largely studied in the literature. In this work, a look-forward exploring strategy is proposed and applied to the sphere strip packing problem to place a set of n spheres of known radii into a parallelepiped of fixed width and height but variable length. The objective is then to minimize the length. The computational results, compared with other already published works in the literature, show the effectiveness of the proposed algorithm since it improves most of the results.**

***Index Terms*—Cutting-and-Packing, Local Search, Look-Forward Strategy, Sphere Packing, Strip Packing.**

I. INTRODUCTION

Cutting and Packing (C&P) problems are optimization problems that are studied by many authors in the literature because they have many applications in a large variety of fields. A C&P problem consists of packing/cutting a given set of items into/from a container/larger item. The items and the container may have various regular or irregular shapes and have a two- or three-dimensional (2D/3D) form. One of the most popular applications in the 2D case is the cutting of the largest set of items from a large plate and the objective is to minimize the waste. Many applications exist in wood, metal, textile and glass industries. An application in the glass industry is for example described in [1]. In the three-dimensional case, the objective is to place a set of items (bins, spheres, irregular 3D shape objects, …) into a larger 3D container. For example, in [2], the authors studied the case of storage and transportation of items. An extensive literature review of the 2D/3D C&P problems is given in [3].

The problem studied in this work is the three-dimensional strip packing problem (3DSPP). More precisely, given a set S that contains n spheres s_i of known radii r_i $(i = 1, \dots, n)$ and a parallelepiped of fixed depth D and height H but of unlimited length L, the problem to solve is then to pack the n spheres into the container of minimum length.

II. LITERATURE REVIEW

The sphere packing problem has various applications like in the industry to model solid states or objects [4], logistics, and health [5]. This problem is also encountered in Medicine [5] and Physics to model solid and liquid states [4]. Several methods and algorithms exist in the literature for the sphere packing problem. For example, the authors in [6] proposed a Monte Carlo based algorithm for packing identical spheres inside a cube. In [7], a global optimization approach for packing unequal spheres was proposed. Authors in [8] developed a variable-neighborhood-search-based heuristic for packing equal spheres into the smallest cube. Finally, in [9] and [10], the authors have studied 3DSPP and proposed dichotomous search-based algorithms to place a set of non-identical spheres inside a parallelepiped, this is then the same problem as that studied in this paper.

In this work, a look-forward-based strategy is explored to solve 3DSPP. The proposed algorithm explores more search space than the standard look-forward strategy (studied for example in [11]) to try to improve the results.

III. THE PROPOSED ALGORITHM

This section describes the sphere packing process, the look-forward principle and finally the proposed algorithm.

Let S be a set of n spheres $s_i \in S, (i = 1, \dots, n)$ of known radii r_i to pack into a 3D a parallelepiped of dimensions $L \times H \times D$ (length, depth and height respectively) placed with its bottom-left-back corner corresponding to the origin $(0,0,0)$ of axes in the 3D

Euclidian space as shown in Fig. 1. The six faces of the 3D container are denoted by the set $F = \{front, back, bottom, top, left, right\}$ where the front face is parallel to the 2D-plane formed by $\vec{X}$ and $\vec{Y}$ axes (cf. Fig. 1), i.e., points with coordinates (x, y, D) where $0 \leq x \leq L$, $0 \leq y \leq H$, and $z = D$.

In the general case, let S_{in} the set of spheres already placed inside the container and S_{out} is the set of spheres that are not yet placed (note that $S = S_{in} \cup S_{out}$). In Fig. 1, two spheres are already packed (blue spheres), i.e., $S_{in} = \{s_1, s_2\}$. Six possible corner positions to pack the next sphere s_3 are also indicaded. These positions correspond to set $P = \{p_3^1, \dots, p_3^6\}$ where p_3^k is the k^{th} position for sphere s_3. Each corner position touches at least three other items (those in fact used to compute this position). An item may be another sphere or one of the six faces of the parallelepiped. For example, position p_3^1 touches the left and back faces, and sphere s_1 , then we note $T(p_3^1) = \{s_1, back, left\}$.

A. A greedy algorithm for packing spheres (3DMHD)

Such an algorithm often contains the following steps:

- Place the first sphere at the bottom-left-back corner
- At each step of the algorithm, compute the possible positions for each sphere that is not yet packed. Each position is evaluated by using a criterion (see below for an example)

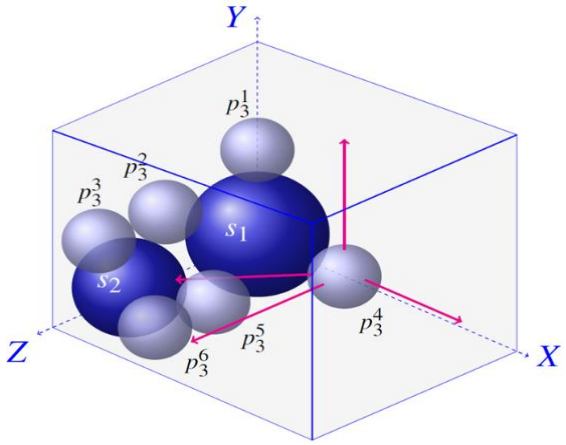


Fig. 1. The heuristic basic principle for placing the next sphere inside the container

- The best evaluated corner position is then chosen to place the next sphere
- The two steps explained above are repeated until all the spheres are placed (a successful solution is obtained) or no position is possible to place the next sphere (the algorithm fails to find a solution, meaning that the container length has to be greater).

In this work, the greedy algorithm used is MHD (Maximum Hole Degree), proposed in [11], for packing circles into a two-dimensional (2D) container but adapted here to the 3D case. The corner positions are evaluated by computing the "hole degree" (noted λ) parameter or criterion. More precisely, the hole degree for corner position $p_{i+1}^k \in P$ is computed using (1) where F represents the six faces of the container, $d_{i+1,j}^k$ is the distance between the possible sphere of radius r_{i+1}^k, to place at position p_{i+1}^k, and the other items , i.e., the already placed spheres S_{in} and the six faces of the container (F) but excluding elements $T(p_{i+1}^k)$ used to compute the corner position since this distance is always equal to zero. Fig. 1 shows for example the distance between position p_3^4 and the other items. The MHD heuristic places, at each step, the next sphere at the position that has the greatest (maximum) hole degree.

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

In 3DSPP (2D strip packing problem), the length of the container is decreased each time the greedy heuristic succeeded to pack all the spheres and the length is increased otherwise. A dichotomous search, based on lower and upper bounds (values for the length L) may then be used to compute the final solution.

B. The look-forward principle for packing spheres

To improve the solution quality of the greedy algorithm, a strategy consists of using the look-forward (LF), also called look-ahead, strategy. The principle of LF is as follows: at a given step i of the packing process, i spheres are already and definitely packed inside the container. To pack the next sphere, each corner position $p_{i+1}^k \in P$ is evaluated by the greedy MHD heuristic itself, i.e., a final solution is computed by using each corner position. Finally, the position that had led to the "best" solution is chosen to definitely place the next sphere s_{i+1}.

C. The extended look-forward (ELF) algorithm for packing spheres

Several works, like in [11], demonstrated the effectiveness of the look-forward strategy compared to the greedy heuristic. This is because the search space explored (the number of possible corner positions) is greater. In this work, we propose an extended version of look-forward, denoted by ELF, that consists of using two nested look-forward searches. Fig. 2 indicates the obtained algorithm.

1. $L \leftarrow L_{max}$;
2. Compute set P of corner positions in the eight corners of the parallelepiped;
3. $i \leftarrow 0$;
4. Repeat
5. for each of the α best positions $p_j \in P$ do
6. Let $P^{\alpha} \leftarrow P$;
7. Place the next sphere s_{i+1}^{α} at position $p_j \in P^{\alpha}$ and update P^{α};
8. for each of the β best positions $p_{j'} \in P^{\alpha}$ do
9. Let $P^{\beta} \leftarrow P^{\alpha}$;
10. Pack the next sphere s_{i+2}^{β} at position $p_{j'} \in P^{\beta}$;
11. Compute a final solution by the greedy 3DMHD heuristic;
12. if all the spheres are successfully placed then
13. $L \leftarrow L - \Delta L$;
14. goto step 2;
15. end if
16. end for (second level)
17. end for (first level)
18. Pack the next sphere s_{i+1} at position $p^* \in P$ that had led to the densest final solution;
19. $i \leftarrow i + 1$;
20. Until set P becomes empty;
21. Exit with the best computed length L^*;

Fig. 2. The extended look-forward algorithm (ELF)

Algorithm ELF (Fig. 2) begins by computing the upper-bound, at step 1, for the container, i.e., L_{max}. This value is computed by using the 3DMHD heuristic that places the spheres in a container of fixed depth and height but unlimited length. The upper-bound corresponds to the sphere located furthest to the right, i.e., $L_{max} = \max(x_i + r_i), i = 1, \dots, n$.

At step 2 of algorithm ELF, the corner positions are computed. Note that these positions are placed in the eight corners of the parallelepiped. The number of definitely placed spheres (i) is set to 0 (step 3). The main loop then takes place (lines 4-20). At each step of this loop, i spheres are already placed, and the objective is to choose the next sphere s_{i+1} to place definitely in the container. The two other nested "for" loops implement the extended (two levels) look-forward strategy. In the first level (line 5), the α "best" positions (those having the greatest values λ, c.f. (1)) are chosen from the set of corner positions P. For each position, a copy of the partial current solution is created (step 6), and a new sphere (s_{i+1}^{α}) is placed (step 7). The second loop (level) starts at step 8, where the α best positions are chosen from the current set of positions P^{α}. For each position, a copy of the current solution is created (step 9) and a new sphere (s_{i+2}^{β}), is placed (step 10). Step 11 continues then to place the remaining spheres using the 3DMHD (greedy) heuristic. If all the spheres are placed, then a successful solution is obtained. We can then reduce the length of the container (step 13) by a given value ΔL (fixed to 0.05 in the computational investigation). The algorithm goes then to step 2 to execute again the extended look-forward strategy with a smaller value of the container length. If all the spheres are not placed after the execution of the two nested loops, then the best position $p^* \in P$ is chosen to definitely place the next sphere s_{i+1} inside the container. Note that p^* is the position that had led to the densest non-feasible solution obtained by the 3DMHD heuristic. The density of a solution is equal to the sum of the volumes of all placed spheres divided by the volume of the container ($L \times H \times D$). Finally, the algorithm exists, at step 21, with the best computed length L^*.

Note that algorithm ELF uses dynamic parameters to manage the extended (two level) look-forward search. These parameters are α and β whose values are chosen in the interval [0%, 100%]. For example, if $\alpha = 50\%$, then only half of positions are considered. The objective is to find the best values that lead to good solutions in reasonable computation time. Indeed, the computation time decreases when the values of α and β decrease.

IV. RESULTS

Algorithm ELF was implemented with C++ language under Linux environment and run on a 2.5 GHz processor and 8 GB of RAM. Table I indicates the values fixed for the different parameters.

Table I. Parameters used by algorithm ELF

Parameter	Description
Time limit	2 hours
$\alpha = 100\%$	Percentage of positions explored at the first level of the search
$\beta = 50\%$	Percentage of positions explored at the second level of the search
$\Delta L = 0.05$	Value with which the length of the container is decreased each time a successful solution is obtained

Algorithm ELF was tested on several benchmark instances from the literature, called SYS, that contain from 65 to 100 non-identical spheres. The first step of the computational investigation was to set the best values for α and β (the proportion of corner positions to explore in the two levels of the look-forward search). The maximum values ($\alpha = \beta = 100\%$) will explore more search space since all corner positions are evaluated before placing the next sphere into the container. But this will lead to greater computational time. Several combinations were then tested on some representative instances that contain different numbers of spheres. More precisely, the values that have been tested for each parameter are $\alpha = 100\%$ and β equal to 100%, 75%, 50%, and 25%. The best combination that had led to the best average solution (average of the container length obtained within the computation time limit of 2 hours) is $\alpha = 100\%$ and $\beta = 50\%$.

Table II. Results of various algorithms on the benchmark SYS instances.

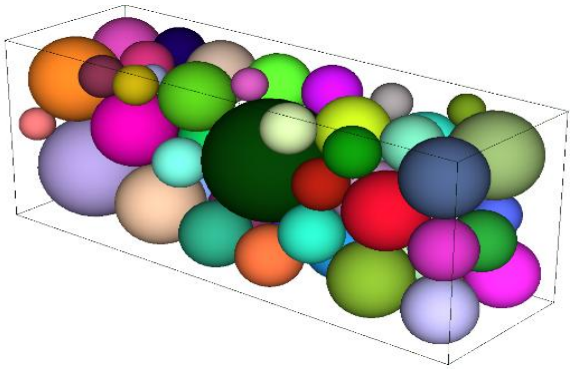
SYS	n	H	D	HY15	HY18	ELF
13a	65	5.5	6.9	18.0193	17.6541	17.5827
13b	65	5.5	6.9	17.9288	17.8331	17.5827
14a	70	5.5	6.9	34.0449	33.7071	33.3585
14b	70	8.5	9.9	14.5082	14.4541	14.0453
15a	75	5.5	6.9	34.8934	34.8762	34.3104
15b	75	8.5	9.9	15.0323	14.7818	14.5404
16a	85	5.5	6.9	37.7972	37.3114	36.6762
16b	85	8.5	9.9	16.2319	15.9001	15.5450
23a	75	6.5	7.9	15.4148	15.0821	14.9317
23b	75	5.5	6.9	21.2641	21.1228	20.7023
24a	80	6.5	7.9	26.6540	26.4165	25.8254
24b	80	8.5	9.9	15.9796	15.8345	15.4568
25a	85	6.5	7.9	27.5623	27.0778	26.6795
25b	85	8.5	9.9	16.2175	16.1536	15.8196
26a	95	6.5	7.9	29.3127	29.2441	28.4995
26b	95	8.5	9.9	17.4633	17.4224	16.8939
34a	85	5.5	6.9	32.6665	32.5005	31.9785
34b	85	8.5	9.9	14.2483	14.0865	13.7839
35a	90	5.5	6.9	33.9108	33.4542	33.0778
35b	90	8.5	9.9	14.7881	14.2232	14.7346
36a	100	5.5	6.9	36.7125	36.2041	35.6158
36b	100	8.5	9.9	15.8566	15.6591	15.2691

Table II contains the results obtained by two algorithms, namely HY15 [9] and HY18 [10], compared to those obtained by the proposed algorithm in this work (ELF). More precisely, column 1 gives the name of the instance, column 2 indicates its size (number of spheres n). The next two columns show the height (H) and the depth (D) of the 3D container. Finally, columns 5 to 7 contain the best length computed by the three compared algorithms:

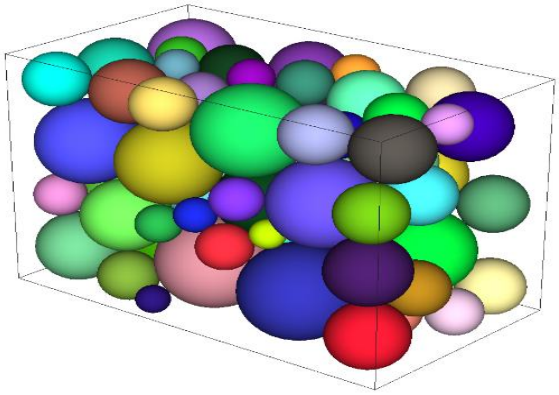
- HY15: that uses a dichotomous-based search (that is in fact based on a tree-search called beam search) [9]
- HY18: that also implements a global dichotomous-based search [10]
- ELF: Extended look-forward strategy that combines a double look-forward and the maximum hole degree (MHD) heuristic.
-

Note that algorithms HY15 and HY18 were executed under a similar environment (programming language and processor speed). The comparison with these algorithms is then fair.

From the results of Table II, we can notice that the proposed algorithm improves all the solution (length of the container) computed by the two other algorithm (H15 and HY18) except for one instance (35b) where HY18 computed a better (smaller) length. These results prove the effectiveness of algorithm ELF.



(a). Instance 13a (n = 65, $L^* = 15.5827$)



(a). Instance 35b (n = 100, $L^* = 14.7346$)

FIG. 2. Example of two solutions obtained by algorithm ELF

FIG.2 shows two solutions obtained by the proposed algorithm ELF on two different benchmarks instances. FIG. 2–(a) displays the solution on the smallest benchmark instance that contains 65 spheres, while FIG. 2–(b) corresponds to one of the largest instances, here 35b, containing 100 spheres.

V. CONCLUSION

In this paper, a look-forward based algorithm (ELF) is proposed to solve the sphere packing problem. ELF was tested on a set of benchmark instances that contain a large number of spheres (between 65 and 100 spheres). The computational investigation indicated that ELF outperforms other algorithms already published in the literature since it computes better quality solutions within the same computation time limit.

In a future work, it will be interesting to implement other metaheuristics such as tabu search and ant colony to solve the same problem.

REFERENCES

[1] Benedict, R., Yeh, W.C., Chou, W.Y., Liu, Z., Zhu, W., and Huang C.L. (2026). Simplified swarm optimization for cutting optimization problems. Engineering Applications of Artificial Intelligence, 164(B), 113284. https://doi.org/10.1016/j.engappai.2025.113284

[2] Joung, Y.K. and Noh, S.D. (2014). Intelligent 3D packing using a grouping algorithm for automative container engineering. Journal of Computational Design and Engineering, 1, 140–151.

[3] Senergues, V., Brahimi, N., Cherri, A.C., Klein, F., and Péton, O. (2026). Cutting stock problem with usable leftovers: A review. European Journal of Operational Research, 328(1), 1-14. https://doi.org/10.1016/j.ejor.2025.03.014

[4] Čapek, P. (2026). Random packing of polydisperse spheres as a model of shrunken solidified suspensions. Powder Technology, 476, 122384, https://doi.org/10.1016/j.powtec.2026.122384.

[5] Wang, J. (1999). Packing of unequal spheres and automated radio-surgical treatment planning. Journal of Combinatorial Optimization, 3, 453-463

[6] Soontrapa, K. and Chen, Y. (2013). Mono-sized sphere packing algorithm development using optimized Monte Carlo technique. Advanced Powder Technology, 24, 955–961

[7] Sutou, A., and Dai, Y. (2002). Global optimization approach to unequal sphere packing problems in 3D. Journal of Optimization Theory and Applications, 114, 671-694

[8] M'Hallah, R. and Alkandari, A. (2012). Packing unit spheres into a cube using VNS. Electronic Notes in Discrete Mathematics, 39, 201–208

[9] Hifi, M., and Yousef, L. (2015). A dichotomous search-based heuristic for the three-dimensional sphere packing problem. Cogent Engineering, 2, 994257. https://doi:10.1080/23311916.2014.994257.

[10] Hifi, M., and Yousef, L. (2018). A global dichotomous search-based heuristic for the three-dimensional sphere packing problem. International Journal of Operational Research, 33, 139–160. https://doi:10.1504/ijor.2018.095195.

[11] Huang, W. Q., Li, Y., Akeb, H., and Li, C. M. (2005). Greedy algorithms for packing unequal circles into a rectangular container. Journal of the Operations Research Society, 56, 539–548