Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 357–364
ISBN 978-83-60810-51-4

An Improved Algorithm for the Strip Packing Problem

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

Mhand Hifi, Dominique Lazure
Université de Picardie Jules Verne,
UR EPROAD,
7 Rue du Moulin Neuf,
80039 Amiens, France
Email: {mhand.hifi, dominique.lazure}@u-picardie.fr

***Abstract*—This paper solves the strip packing problem (SPP) that consists in packing a set of circular objects into a rectangle of fixed width and unlimited length. The objective is to minimize the length of the rectangle that will contain all the objects such that no object overlaps another one. The proposed algorithm uses a look-ahead method combined with beam search and a restarting strategy. The particularity of this algorithm is that it can achieve good results quickly (faster than other known methods and algorithms) even when the number of objects is large. The results obtained on well-known benchmark instances from the literature show that the algorithm improves a lot of best known solutions.**

I. Introduction

CUTTING & PACKING (C&P) problems are well known in Operations Research since they have many practical applications. They can be encountered in the storage and transportation of objects of different shapes (Baltacioglu *et al.* [1]; Conway and Sloane [2]; Lewis *et al.* [3]). In this case, the objective is to arrange these objects in order to save space. C&P problems are also used in the industry when a set of pieces of predetermined shapes have to be cut from a rectangular plate (Menon and Schrage [4]). The objective in this second example is to minimize the waste due to the space between the pieces to cut.

This paper studies the problem of cutting (or packing) a set N of n circular pieces C_i of known radii $r_i, i \in N$, from (or into) a strip S of fixed width W and unlimited length L. The objective is to place the n pieces inside the smallest rectangle R of dimensions $W \times L^*$ such that no piece overlaps another one and no piece exceeds the limits of the rectangle. This problem is known as the *Strip Packing Problem* or SPP (see Wäscher *et al.* [5]).

The mathematical formulation for SPP is as follows:

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

$$(x_i - x_j)^2 + (y_i - y_j)^2 \geq (r_i + r_j)^2, \text{ for } j < i,\ (i,j) \in N^2 \tag{2}$$

$$x_i - r_i \geq 0,\ \forall i \in N, \tag{3}$$

$$y_i - r_i \geq 0,\ \forall i \in N, \tag{4}$$

$$L - x_i - r_i \geq 0,\ \ \forall i \in N \tag{5}$$

$$W - y_i - r_i \geq 0,\ \ \forall i \in N \tag{6}$$

$$L \geq \frac{\pi}{W} \times \sum_{i=1}^{n} r_i^2 \tag{7}$$

Equation 1 indicates the objective to minimize, i.e., the length of the target rectangle that will contain the n pieces. Equation 2 ensures that any pair of distinct circles C_i and C_j do not overlap each other, i.e., the distance between their centers must be greater than or equal to the sum of their radii $r_i + r_j$. Equations 3–6 mean that any circle C_i does not exceed the container boundary. Finally, Equation 7 indicates that the problem contains a lower bound limit for L, this limit, denoted by $\underline{L}$, is equal the sum of the surfaces of the n circles divided by the width of the rectangle W. Any value for L cannot then be smaller than this lower bound.

A solution for the strip packing problem consists to find the minimum value for the length of the rectangle that will contain all the pieces while verifying the constraints represented by Equations 2–7.

II. Related Literature

The problem of packing circular objects of different radii into a container is well known and very studied in the literature. Since there is no method calculating exact solutions, the authors use generally heuristic-based approaches in order to compute approximate solutions for the problem. Two categories of containers can be distinguished: the first one corresponds to a circle and the second one to a rectangle.

Packing different-sized circles into the smallest circle was for example studied by Huang *et al.* [6] where the authors proposed greedy algorithms based on the Maximum Hole Degree (MHD) heuristic. Hifi and M'Hallah [7] proposed a dynamic adaptive local search where the radius of the containing circle is increased when placing the circles. For the same problem, Akeb *et al.* [8] used beam-search based algorithms.

The problem of packing circles of different radii into a rectangular container is more studied in the literature because of its many applications. For example George *et al.* [9] proposed several rules based essentially on the use of a genetic algorithm as well as a random strategy. Stoyan and Yaskov [10] designed a mathematical model whose objective is to search for feasible local optima by combining a tree-search procedure and a reduced-gradient. A genetic algorithm was also used by Hifi and M'Hallah [11]. Huang *et al.* [12] designed two greedy algorithms for the strip packing problem, the algorithms, denoted by B1.0 and B1.5, are based on the Maximum Hole Degree (MHD) heuristic. Birgin *et al.* [13] used a non-linear approach for placing circles inside a rectangle. Kubach *et al.* [14] proposed a parallel version of the MHD heuristic for tackling the strip packing problem. Finally, Akeb *et al.* [15] proposed a beam-search based algorithm coupled with a restarting strategy for SPP.

In this paper, an improved algorithm is proposed for the strip packing problem. This algorithm combines beam search, a restarting strategy, and a look-ahead method. The objective of the look-ahead is to accelerate the search to obtain quickly solutions. In addition, the parameters of the restarting and the look-ahead strategies are studied in order to adapt them to the characteristics of each instance.

The rest of the paper is organized as follows. Section III explains how to use beam search in order to resolve the strip packing problem (SPP). Section IV returns on some existing beam-search based algorithms for SPP. Section V details the improved algorithm denoted by IA. Section VI discusses the results obtained by IA on the most known instances in the literature. Finally, Section VII summarizes the results obtained and indicates some orientations for future work.

III. Beam Search for Resolving SPP

Beam search (BS) [16] is a tree-based search and is an adaptation of the best first search. BS selects, at each level of the tree, the most promising nodes to expand. The number of the nodes chosen at each level is denoted by ω and represents the *beam width*.

This section is organized as follows. First, the different notations used throughout the paper are given. After that, a greedy procedure, denoted by MLDP (Minimum Local Distance Position) is described. The objective of MLDP is to try to place the n circles inside the current rectangle $R = W \times L$, i.e., when the length of the rectangle is fixed to a given value L.

A. Notations

In order to simplify the reading of the paper, here are the different notations used throughout the document :

1) $N = \{1, ..., n\}$ is the set of circles to pack into the strip S placed with its bottom left corner at $(0, 0)$ in the Euclidean plan,
2) $M = \{1, ..., m\}$ is the set of circle types (the set of different radii in the instance),
3) S_{left}, S_{top}, S_{right}, and S_{bottom} are the four edges of S,
4) The circular piece C_i of radius r_i is placed with its center at coordinates (x_i, y_i),
5) I_i corresponds to the set of circles already packed inside the strip ($|I_i| = i$),
6) $\overline{I}_i$ contains the circles not yet placed ($I_i \cup \overline{I}_i = N$),
7) P_{I_i} is the set of distinct corner positions for the next circle to place C_{i+1} given the set I_i,

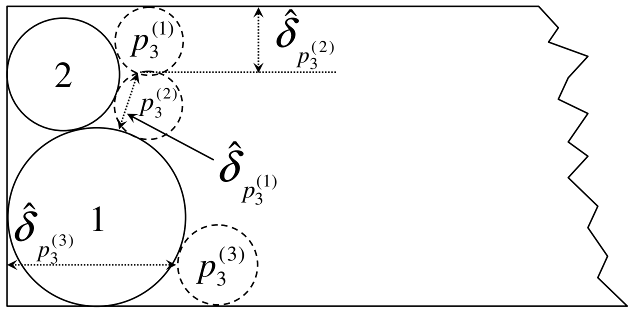

Fig. 1. The MLDP strategy

8) A corner position $p_{i+1} \in P_{I_i}$ for C_{i+1} is computed by using two elements e_1 and e_2. An element is either a piece already placed (set I_i) or one of the three edges of S (S_{left}, S_{top}, S_{bottom}). $T_{p_{i+1}}$ denotes the set composed of both elements e_1 and e_2.

B. The MDLP Greedy Procedure

The Minimum Local Distance Position (MLDP) procedure can be used as a greedy algorithm in order to compute a solution. Indeed, given the set I_i of circles already placed inside the current rectangle and the set of corner positions P_{i+1} for the next circle C_{i+1}, MLDP selects the *best corner position* for this circle. This process is repeated until all the circles are placed or no additional circle can be placed. Fig 1 explains the mechanism of MLDP where two circles are already placed, thus $i = 2$ and $I_2 = \{C_1, C_2\}$. There are also three possible positions to place the next circle C_3: $P_{I_2} = \{p_3^{(k)}, k = 1, .., 3\}$. The first corner position $p_3^{(1)}$ touches circle C_2 and the top-edge of the strip S_{top}, then $T_{p_3^{(1)}} = \{C_2, S_{\text{top}}\}$. For the two others corner positions, $T_{p_3^{(2)}} = \{C_1, C_2\}$ and $T_{p_3^{(3)}} = \{C_1, S_{\text{bottom}}\}$.

Let C_{i+1} be the circular piece to place at position p_{i+1} and $\delta_{i+1}(edge)$, $edge \in E_{\text{edge}} = \{S_{\text{left}}, S_{\text{bottom}}, S_{\text{top}}\}$, the three distances defined as follows: $\delta_{i+1}(S_{\text{left}}) = x_{i+1} - r_{i+1}$, $\delta_{i+1}(S_{\text{bottom}}) = y_{i+1} - r_{i+1}$, and $\delta_{i+1}(S_{\text{top}}) = W - y_{i+1} - r_{i+1}$.

The euclidean distance from the edge of the next circle to pack C_{i+1} (when positioned at p_{i+1}) and C_j is denoted by $\delta_{i+1}(j)$ and is computed as follows:

$$\delta_{i+1}(j) = \sqrt{(x_{i+1} - x_j)^2 + (y_{i+1} - y_j)^2} - (r_{i+1} + r_j) \quad (8)$$

The MLDP of the circular piece C_{i+1} when placed at $p_{i+1} \in P_{I_i}$ is calculated as follows:

$$\hat{\delta}_{p_{i+1}} = \min_{\alpha \in I_i \cup E_{\text{edge}} \setminus T_{p_{i+1}}} \{\delta_{i+1}(\alpha)\} \quad (9)$$

Equation (9) gives the MLDP of C_{i+1} which is computed by using the distances between the piece to place at position p_{i+1} and the elements of the set $I_i \cup \{S_{\text{left}}, S_{\text{bottom}}, S_{\text{top}}\} \setminus T_{p_{i+1}}$ containing the pieces already placed, the three edges of the strip, but by excluding the two elements of $T_{p_{i+1}}$ used for computing the coordinates of C_{i+1} because the corresponding distance is always equal to zero. Note however that the MLDP is equal to zero when C_{i+1} touches more than two elements because one of the three elements does not belong to the set $T_{p_{i+1}}$ and then the distance to this element can be taken into account. Fig. 1 indicates the MLDP $\hat{\delta}_{p_3^{(k)}}$ of each position $p_3^{(k)},\ k = 1, 2, 3$.

For calculating a packing of the pieces when using the MLDP procedure, the following process is executed: MLDP starts by placing the first circular piece C_1 at the bottom-left corner (at coordinates $(r_1,\ r_1)$), the $n-1$ remaining pieces are successively packed by using the MLDP rule as explained above. For example, in Fig. 1, C_3 will be placed at position $p_3^{(1)}$ since the corresponding MLDP has the minimum value.

IV. Beam Search-Based Algorithms for SPP

Akeb *et al.* [15] proposed an augmented beam search algorithm, denoted by SEP-MSBS, for the strip packing problem. SEP-MSBS combines two main techniques:

- A strategy based on the use of separate beams that aims to diversify the search space compared to the standard beam search,
- A restarting strategy that consists to rerun the search by changing the first circle to place. The objective of this second technique is to escape from local optima.

The separate-beams mechanism is displayed in Fig. 2. The root node η_1 at level $\ell = 1$ contains the starting configuration (one circle placed in the bottom-left corner of the rectangle) as well as the possible positions for the $n-1$ remaining circles. The best positions (having the smallest MLDP values) are chosen for branching, this creates the second level $\ell = 2$ (note that each branching consists to choose a position where to place the next circle). From this second level, separate beams are initiated. More precisely, a beam of width $\omega = 1$ is initiated from the first node (the best node), a beam of width $\omega = 2$ is initiated from the second best node, and so on. Thus, the node at position i in the second level is explored by applying a beam search of width $\omega = i$. This is to say that the best nodes do not require an extensive search, the beam width has then a small value, unlike the last nodes in the level that need larger values for the beam width. The separate-beams strategy was shown in [15] to be better than the standard beam search.

Even if SEP-MSBS obtains good results (often the best results in the literature) on the instances used, its run time remains too large. This is mainly due to the restarting strategy, which is executed m times (the number of different circles (radii) in the instance).

V. An Improved Algorithm for SPP

In this paper, we try to improve the SEP-MSBS algorithm by adding a look-ahead strategy. The look-ahead-based mechanism will be described in Section V-A. The proposed improved algorithm, denoted by IA, will be given and explained in Section V-B. Some adjustments are introduced in algorithm IA in order to reduce the computation time, these adjustments concern the number of corner positions to explore by the look-ahead strategy as well as the number of circles to take into account in the restarting strategy.

A. A Look-Ahead Based Algorithm

Algorithm SEP-MSBS [15] selects, at each level of the tree, the best nodes by using the MLDP rule (Sect. III-B). This can be assimilated to a *local evaluation*, the packing process does not take into account the remaining circles to place. The look-ahead proceeds differently. Indeed, given the set of nodes $B = \{\eta_\ell^1, ..., \eta_\ell^\omega\}$ of the current level ℓ in the tree, each node η_ℓ^i is characterized by the set I_{ℓ_i} of ℓ circles already placed in the current rectangle and the set P_{ℓ_i} of corner positions for the remaining circles, the look-ahead evaluates each position $p \in P_{\ell_i}$ by continuing the placement of the remaining circles by using the MLDP rule. The objective is to compute final solutions which will help to choose the actual positions for branching from the current level ℓ. This strategy is implemented in the Look-Ahead Branching Procedure (LABP) displayed in Algorithm 1.

In addition to the set of nodes B, LABP (Algorithm 1) receives as input parameter an indicator `feasible` set to the value `false` as well as a real number $0 < \psi \leq 1$. Parameter ψ serves to determine the proportion of corner positions to evaluate by the look-ahead, for example, if $\psi = 0.8$, then only the best 80% of corner positions (those having the smallest MLDP values) are evaluated. The objective of this parameter is to accelerate the algorithm for large instances (those containing a large number of circles, and therefore a large number of corner positions at each step).

The set Π of positions to evaluate by the look-ahead, as explained above, is constructed in Steps 2 and 3. After that, LABP considers each position $p_j \in \Pi$ (Step 4) by packing

Algorithm 1 The Look-Ahead Branching Procedure (LABP)

Require: A set $B = \{\eta_\ell^1, ..., \eta_\ell^\omega\}$ of ω nodes, a boolean indicator `feasible=false`, and $0 < \psi \leq 1$

Ensure: A feasible solution if `feasible=true`, or a set B_ω of ω nodes (those leading to the highest densities through the MLDP packing procedure).

1: Let P_{ℓ_i} denotes the set of corner positions of node $\eta_\ell^i \in B$;
2: Let Π be the set of all corner positions of B, i.e., $\Pi = \bigcup P_{\ell_i}$;
3: Reduce Π to the $\lceil \psi \times |\Pi| \rceil$ best corner positions (having the best MLDP values);
4: **for all** corner positions $p_j \in \Pi$ **do**
5: Pack $C_{\ell+1}$ in p_j and insert the resulting node $\eta_{\ell+1}$ into B_ω;
6: Place in $\eta_{\ell+1}$ the remaining circles by using the MLDP packing procedure;
7: **if** all circles are placed **then**
8: `feasible = true`;
9: `exit` with a feasible solution;
10: **else**
11: Assign to $\eta_{\ell+1}$ the density obtained by MLDP;
12: **end if**
13: **end for**
14: Reduce B_ω to the ω nodes that led to the highest densities by MLDP;
15: **return** B_ω.

the corresponding circle in p_j (Step 5). This generates a new node $\eta_{\ell+1}$ that is added to the set of offspring nodes B_ω. The new node is then processed by placing the remaining circles by using the MLDP rule (Step 6). Two cases may then be distinguished:

- A feasible packing is obtained (Step 7), meaning that the n circles were successfully placed inside the current rectangle. In this case, the procedure stops with `feasible=true` (Steps 8 and 9), meaning that the length L of the rectangle could be decreased;
- A feasible packing was not obtained (the n circles cannot be placed into the current rectangle by MLDP). In this second case, the procedure assigns to the node $\eta_{\ell+1}$ the density of the circles placed (Step 11). The density of a given packing is equal to the sum of the surfaces of the circles placed divided by the surface of the rectangle $L \times W$.

Finally, when all the corner positions are processed without obtaining a feasible packing, then the ω best nodes (those that have led to the highest densities) are returned (Steps 14, 15). This means that the current length of the rectangle is too small and should be increased.

Note that procedure LABP (Algorithm 1) is called by a beam search algorithm denoted by BSLA (Algorithm 2, Line 10). BSLA implements a width-first beam search. It uses an interval search $[\underline{L}, \overline{L}]$ in order to compute the best length

Algorithm 2 Beam Search Look-Ahead algorithm (BSLA)

Require: A node η_ℓ, the beam width ω, the bounds of the interval search $(\underline{L}, \overline{L})$, and $0 < \psi \leq 1$

Ensure: The best value for the rectangle's length (L_{best}) and the corresponding feasible packing.

1: Let B denote the set of nodes to be considered;
2: Let B_ω denote the set of descendants of the nodes in B;
3: Let L_{best} be the best length found so far;
4: Let `feasible` be a boolean indicator;

5: **while** $(\overline{L} - \underline{L} > \delta)$ **do**
6: Set $B = \{\eta_\ell\}$, where η_ℓ is a starting node of level ℓ characterized by I_ℓ, $\overline{I}_\ell$, and P_{I_ℓ};
7: $L^* = (\overline{L} + \underline{L})/2$;
8: `feasible = false`;
9: **while** ($B \neq \emptyset$ and `feasible=false`) **do**
10: B_ω = LABP(B, `feasible`, ψ);
11: **if** `feasible=true` **then**
12: $L_{\text{best}} = L^*$; $\overline{L} = L^*$;
13: **else**
14: $\ell = \ell + 1$; $B = B_\omega$; $B_\omega = \emptyset$;
15: **end if**
16: **end while**
17: **if** `feasible=false` **then**
18: $\underline{L} = L^*$;
19: **end if**
20: **end while**

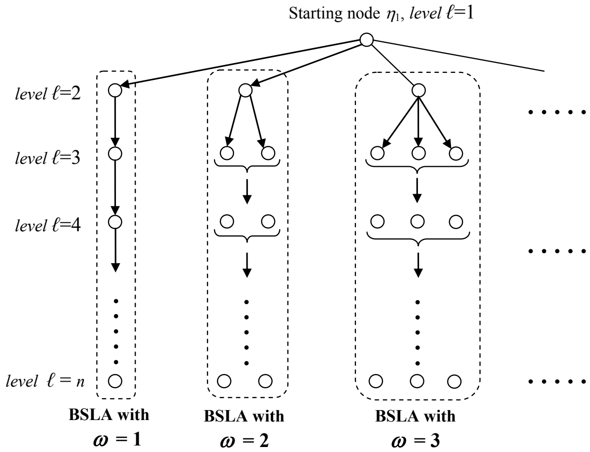

Fig. 2. Separate beams and Look-ahead

of the rectangle containing all the circles. BSLA receives several input parameters: the starting node η_ℓ containing the starting configuration, the beam width value ω, the values of the interval search, and parameter ψ indicating the proportion of corner positions to process by LABP.

BSLA calls, at Step 10, the LABP procedure (Algorithm 1) for each value of the rectangle's length L^*. If LABP has computed a feasible packing with the current value of L, then the best length (L_{best}) is updated (Step 12) and the upper bound of the interval search $\overline{L}$ is set equal to the current length. Otherwise, level ℓ is incremented by 1, the best expanded nodes returned by LABP (Step 10) replace the nodes of the current level in the tree $(B = B_\omega)$, and B_ω is reset to the empty set (Step 14). If LABP did not succeed to compute a feasible packing with the current value of the rectangle's length (Step 17), then the lower bound of the interval search $\underline{L}$ is set equal to the current value of L, i.e., $\underline{L} = L^*$ (Step 18) meaning that the rectangle's length is too small. Finally, it is to note that the binary interval search is stopped when the difference between $\underline{L}$ and $\overline{L}$ becomes less than or equal to a given gap δ.

B. The Improved Algorithm (IA)

The improved algorithm, denoted by IA, is given in Algorithm 3. It combines three main techniques: separate beam search, a restarting strategy, and look-ahead. Fig.2 shows how algorithm IA works.

IA receives as input parameters the beam width ω, parameter τ that serves to indicate the proportion of circles taken into account by the restarting strategy, and parameter ψ used to choose the proportion of corner positions to evaluate by the look-ahead branching procedure LABP (Algorithm 1). The output of algorithm IA is a feasible packing and the corresponding best length of the rectangle L_{best}.

At Step 1 of algorithm IA, the best length L_{best} is set equal to the upper bound of the length $\overline{L}$ which is computed by an Open Strip Generation Solution Procedure (OSGSP_a) [17]. The lower bound of the interval search $\underline{L}$ is set equal to

Algorithm 3 The Improved Algorithm (IA)

Require: The beam width ω, parameters τ and ψ

Ensure: A feasible packing with the best length L_{best} for the strip

1: $L_{\text{best}} = \overline{L}$; $\underline{L} = (\pi \times \sum_{i=1}^{n} r_i^2)/W$;
2: Rank the pieces of N in decreasing value of their radii;
3: Let $\mathcal{T}$ be the set of circle types (different circles in N);
4: Reduce $\mathcal{T}$ by keeping only $\lceil \tau \times |\mathcal{T}| \rceil$ circles;
5: Set $i_{order} = 1$, where i_{order} is the index of the first circular piece of the set $\mathcal{T}$;
6: **while** $(i_{order} \leq |\mathcal{T}|)$ **do**
7: Generate the node η_1, characterized by I_1, $\overline{I}_1$, and P_{I_1}, by placing the first circle $C_{i_{order}}$ inside the current rectangle and let $B = \eta_1$;
8: Branch out of B and generate the list of offspring nodes B_ω;
9: Let $B = \min(\omega, |B_\omega|)$ nodes having the best MLDPs and corresponding to distinct corner positions and reset $B_\omega = \emptyset$;
10: Let η_2 be the node at position ω in B;
11: `feasible` = BSLA($\eta_2, \omega, \underline{L}, \overline{L}, \psi$);
12: **if** `feasible`= `true` **then**
13: $\overline{L}$ and L_{best} are updated if a better length is obtained by BSLA;
14: **end if**
15: $\underline{L} = (\pi \times \sum_{i=1}^{n} r_i^2)/W$;
16: $i_{order} = i_{order} + 1$;
17: **end while**
18: **exit** with the best target length L_{best}.

the natural lower bound, i.e., $\underline{L} = (\pi \times \sum_{i=1}^{n} r_i^2)/W$ which corresponds to a density equal to 1, this density is of course not possible to obtain because there is always a non-occupied space between the circles and between the circles and the edges of the rectangle. The pieces are then ranked by decreasing value of their radii (Step 2). The set of circle types $\mathcal{T}$ to use in the restarting strategy is constructed in Steps 3 and 4. The index serving to indicate the first circle to place in the bottom-left corner of the current rectangle is initialized in Step 5.

The root node η_1 (cf. Fig. 2) is generated in Step 7. This corresponds to the placement of circle $C_{i_{order}}$ in the bottom-left corner of the rectangle. In Step 8, the list of offspring nodes B_ω is generated. The set B is after that set equal to the ω best nodes of B_ω, this correspond to level $\ell = 2$ in Fig. 2. Since the separate-beams mechanism is used, then only the node at position ω in this level is explored. The node chosen (η_2) is then transmitted to the Beam Search Look-Ahead algorithm BSLA (Algorithm 2) in order to try to compute a feasible solution (Step 11). If BSLA have reached a feasible packing, then the upper bound $\overline{L}$ of the interval search and the best solution L_{best} are updated (Step 13). Indeed, the upper bound $\overline{L}$ is set to the best value obtained L_{best}. After that the lower bound $\underline{L}$ is reset to the natural lower bound (Step 15). In Step 16, the next circle in set $\mathcal{T}$ is chosen in order to restart the algorithm.

It is to note that the main interest of the look-ahead strategy is that it allows algorithm IA, for which the mechanism is described in Fig.2, to compute feasible solutions from the second level ($\ell = 2$) in the search tree in opposite to the other beam search-based algorithms where feasible solutions are obtained in the last level ($\ell = n$).

VI. Computational Results

The algorithms are coded in C++ language and run on a computer with a 3-GHz processor and 256 MB of RAM. Eighteen instances are considered containing from 20 to 200 circles (note that the problem is considered to be large when the number of pieces is at least $n = 100$). The first six instances, denoted by SY1, SY2, SY3, SY4, SY5, and SY6, contain from 20 to 100 circles. They were proposed by Stoyan and Yaskov [10] and are the most known ones in the literature for the strip packing problem, they were for example used in [10], [12], [17], [14], [15]. Twelve additional instances were proposed by Akeb and Hifi [17], these instances are obtained by concatenating the six original instances of Stoyan and Yaskov and contain from 45 to 200 pieces.

It is to note that all these instances are strongly heterogeneous, i.e., the pieces are practically all of different radii ($m \lessapprox n$) where n is the number of circles in the instance and m the number of circle types (different radii).

A. Varying the Beam Width When the Look-Ahead is Used

In a standard beam-search-based algorithm, like for algorithm BSBIS [17], it is difficult to know in advance what value to use for the beam width (ω). Indeed, increasing the value of ω does not necessarily improve the solution, even if that increase the search space. This can be explained by the fact that a standard beam search is based on a local evaluation (e.g. MLDP rule) for branching from the current level of the search tree in order to create the next level. As a result, the value of the solution (the length of the rectangle L) oscillates when increasing the beam width. An example is shown in Fig. 3 where BSBIS was executed on the first instance ($SY1, n = m = 30$) for all the values of $1 \leq \omega \leq 75$. Note that this phenomenon concerns also algorithm SEP-MSBS [15] since this one is based on the MLDP selection strategy.

But when the look-ahead is introduced (see Algorithm 2, BSLA), the solution L oscillates much less (as indicated in Fig. 3) and the value of L often decreases when the value of the beam width ω increases. In addition, the solution obtained by the look-ahead (BSLA) is practically always better than that given by BSBIS and the example shown in Fig. 3 is very representative since this phenomenon was shown for all the instances. It is then not necessary to run a look-ahead-based algorithm with all the possible values of ω. In fact, the computational investigation showed that starting with the value $\omega = 10$ and increasing this value by step of 5, i.e.,

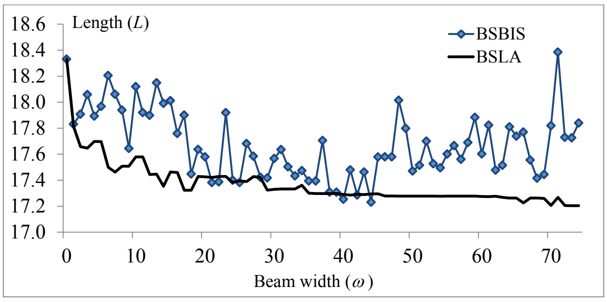

Fig. 3. Effect of the use of the look-ahead on the solution quality, here when varying ω from 1 to 75 on instance SY1 $(m = n = 35)$ circles.

$(\omega = 10 + 5 \times k,\ k \in \mathbb{N})$ corresponds to a good setting. Then, the proposed algorithm (IA) is run with these values of ω.

B. Values of Parameters ψ and τ

Another investigation was conducted. It concerns the values of parameter ψ corresponding to the proportion of positions to evaluate by the look-ahead branching procedure (see Algorithm 1) at each level of the tree, and parameter τ that indicates the proportion of circles to use for restarting algorithm IA (see Algorithm 3).

Each parameter ψ and τ was varied in the discrete interval $\{0.5, 0.75, 1\}$, which gives 9 possibilities. The nine possibilities were tested on 3 sets of instances:

- the two smallest instances SY2 $(n = m = 20)$ and SY3 $(n = m = 25)$,
- two medium-sized instances SY23 $(n = m = 45)$ and SY14 $(n = m = 65)$,
- two large instances SY6 $(n = 100, m = 98)$ and SY1234 $(n = 110, m = 105)$.

Table I indicates the best values for parameters ψ and τ according to the size of the instance. For example, when considering a small-sized instance $(n < 40)$, then all the corner positions have to be processed by the look-ahead $(\psi = 1)$ and each circle type have to be used by the restarting strategy $(\tau = 1)$. The results of algorithm IA presented in Table II and Table III are obtained by using the values indicated in Table I.

C. Solution Quality of Algorithm IA

Table II shows the results obtained by algorithm IA as well as those obtained by different other algorithms. Column 1 contains the name of the instance and Column 2 (W) indicates the width of the strip. Column 3 (n) gives the size of the instance and Column 4 (m) is the number of circle types in the instance. Column 5 (MHD) represents the best length of the rectangle obtained by the Maximum Hole Degree (MHD) heuristic (Huang *et al.* [12]). The next column (B16) contains the result obtained by a parallel version of MHD (Kubach *et al.* [14]), symbol "–" means that the result of B16 is not known for the corresponding instances. Column 7 indicates the result obtained by the Beam Search Binary Interval Search algorithm (Akeb and Hifi [17]), the value between parentheses correspond to the value of the beam width with which the solution was obtained. The solution obtained by algorithm SEP-MSBS (Akeb *et al.* [15]) is given in Column 8 as well as the corresponding beam width. Column 8 (Best Lit.) shows the best known solution in the literature for the studied instances. Finally, the last column contains the result obtained by the Improved Algorithm (IA).

TABLE I
BEST VALUES FOR PARAMETERS ψ AND τ ACCORDING TO THE SIZE OF THE INSTANCE

Instance size	ψ	τ
small $(n < 40)$	1	1
medium $(40 \leq n < 100)$	0.75	0.75
large $(n \geq 100)$	0.5	1

It is to note that the beam-search based algorithms (BSBIS, SEP-MSBS, and IA) were run by using a beam width limit $\bar{\omega} = 100$ and a computation time limit of thirty hours (as in [15]). For a fair comparison, MHD was also run (on the same computer) by using a time limit of thirty hours.

From the results of Table II, we can see clearly that the new algorithm (IA) has improved twelve results out of eighteen, i.e, 67% of the best known results in the literature. Algorithm SEP-MSBS remains better on four instances (SY12, SY13, SY23, and SY123) and algorithm B16 is better on instances SY5 and SY6.

The computation time is not indicated in Table II for algorithm IA because the limit of thirty hours was reached for all the instances except for the smallest one (SY2, $n = m = 20$) for which the algorithm has attained the beam width limit $(\bar{\omega} = 100)$ and terminated after 13 hours. For the SEP-MSBS algorithm [15], the time limit was reached for thirteen instances out of eighteen (except for instances SY1, SY2, SY3, SY4 and SY23), i.e., when $n \leq 45$. The reason for which algorithm IA reached the time limit is that the look-ahead strategy consumes a lot of time.

What will be the behavior of the proposed algorithm (IA) when fixing a relatively short time limit? Another investigation, in which the time limit was fixed at thirty minutes, was conducted. Table III displays the comparison between the beam search-based algorithms (BSBIS, SEP-MSBS, and IA) when using this new time limit. The first column (Instance) contains the name of the instance. Column 2 contains the best value obtained by the BSBIS algorithm (based on a standard beam search) as well as the corresponding beam width. Column 3 indicates the cumulative computation time (in seconds) in order to obtain the best value L in Column 2. The results obtained by the two other algorithms (SEP-MSBS and IA) are indicated in Columns 4–7. Column 8 gives the percentage of improvement obtained by the new algorithm IA on BSBIS, the improvement is computed as $\frac{L_{\text{BSBIS}} - L_{\text{IA}}}{L_{\text{BSBIS}}} \times 100\%$. In the same way, the last column contains the percentage of improvement obtained by algorithm IA on algorithm SEP-MSBS.

TABLE II
SOLUTION QUALITY OF ALGORITHM IA

Instance	W	n	m	MHD	B16	BSBIS	SEP-MSBS	Best Lit.	IA
SY1	9.5	30	30	17.291	17.247	17.2315 (45)	17.2070 (50)	17.2070	**17.0954** (20)
SY2	8.5	20	20	14.535	14.536	14.6277 (86)	14.5287 (24)	14.5287	**14.4548** (15)
SY3	9	25	25	14.470	14.467	14.5310 (78)	14.4616 (44)	14.4616	**14.4017** (80)
SY4	11	35	35	23.555	23.717	23.6719 (42)	23.4921 (66)	23.4921	**23.3538** (10)
SY5	15	100	99	36.327	**35.859**	36.0796 (95)	36.1818 (22)	35.8590	36.0045 (15)
SY6	19	100	98	36.857	**36.452**	36.8456 (85)	36.7197 (26)	36.4520	36.5573 (10)
SY12	9.5	50	48	30.0668	–	29.7011 (52)	**29.6837** (61)	29.6837	29.7024 (30)
SY13	9.5	55	54	30.8906	–	30.6371(100)	**30.3705** (68)	30.3705	30.4231 (20)
SY14	11	65	65	38.2652	–	38.0922 (79)	37.8518 (63)	37.8518	**37.6187** (10)
SY23	9	45	45	28.2697	–	27.8708 (98)	**27.6351** (89)	27.6351	27.7148 (35)
SY24	11	55	54	34.6048	–	34.5476 (26)	34.1455 (49)	34.1455	**34.0970** (30)
SY34	11	60	59	34.9011	–	34.9354 (39)	34.6859 (43)	34.6859	**34.5983** (25)
SY56	19	200	193	69.9790	–	64.7246 (65)	65.2024 (06)	64.7246	**64.6904** (10)
SY123	9.5	75	72	43.6257	–	43.2558 (64)	**43.0306** (25)	43.0306	43.1709 (15)
SY124	11	85	82	49.3345	–	48.8927 (90)	48.8411 (35)	48.8411	**48.6432** (10)
SY134	11	90	88	49.7214	–	49.3954(100)	49.3362 (27)	49.3362	**49.2238** (10)
SY234	11	80	78	45.8880	–	45.9526 (83)	45.6115 (39)	45.6115	**45.4260** (10)
SY1234	11	110	105	61.9060	–	60.2613 (48)	60.0564 (25)	60.0564	**60.0036** (10)

TABLE III
SOLUTION QUALITY OF ALGORITHM IA WHEN FIXING THE TIME LIMIT AT 30 MINUTES

	BSBIS		SEP-MSBS		IA		%imp.	%imp.
Instance	L (ω^*)	$t^*(sec)$	L (ω^*)	$t^*(sec)$	L (ω^*)	$t^*(sec)$	BSBIS	SEP-MSBS
SY1	17.2315 (45)	166	17.2145 (34)	1463	**17.2029 (20)**	1790	0.17%	0.07%
SY2	14.6277 (86)	222	14.5287 (24)	155	**14.4548 (15)**	216	1.18%	0.51%
SY3	14.5310 (78)	308	14.4616 (44)	1253	**14.4106 (20)**	750	0.83%	0.35%
SY4	23.6719 (42)	211	23.5335 (27)	1662	**23.3538 (10)**	1007	1.34%	0.76%
SY5	36.4042 (14)	445	36.3362 (2)	1324	**36.1707 (10)**	1432	0.64%	0.46%
SY6	36.9387 (26)	1637	37.2555 (3)	669	**36.9232 (10)**	1135	0.04%	0.89%
SY12	**29.7011 (52)**	875	30.0447 (9)	650	29.9744 (10)	1800	-0.92%	0.23%
SY13	30.7415 (21)	165	30.7843 (13)	1800	**30.6149 (10)**	1710	0.41%	0.55%
SY14	38.3573 (37)	885	38.2962 (6)	851	**37.9690 (10)**	1501	1.01%	0.85%
SY23	27.9146 (70)	1116	28.0388 (13)	885	**27.8493 (10)**	1768	0.23%	0.68%
SY24	34.5476 (26)	266	34.6732 (8)	766	**34.3544 (10)**	675	0.56%	0.92%
SY34	34.9354 (39)	720	34.9614 (9)	1304	**34.7531 (10)**	914	0.52%	0.60%
SY56	65.5565 (8)	1022	65.7608 (1)	1800	**65.3079 (10)**	1800	0.38%	0.69%
SY123	43.4907 (44)	1745	43.5815 (6)	1412	**43.4793 (10)**	1511	0.03%	0.23%
SY124	49.3281 (19)	456	49.6348 (5)	1720	**49.1915 (10)**	1661	0.28%	0.89%
SY134	49.8705 (32)	1536	49.9136 (5)	1397	**49.8184 (10)**	1621	0.10%	0.19%
SY234	45.9913 (27)	775	46.1901 (4)	880	**45.9209 (10)**	1321	0.15%	0.58%
SY1234	60.9055 (15)	565	60.8783 (3)	1800	**60.5660 (10)**	1369	0.56%	0.51%

From Table III, we can see clearly that when using a relatively short time limit (which is more practical), the proposed algorithm (IA) is practically always the best one (in 17 cases out of 18), except for the instance SY12 where BSBIS remains better. The good results obtained by algorithm IA can be explained by the fact that the look-ahead strategy computes quickly feasible solutions, i.e., from level $\ell = 2$ in the search tree (see Fig. 2) when BSBIS and SEP-MSBS obtain feasible solutions at level $\ell = n$ only. So, even if algorithm IA is stopped after a short computation time, it will have calculated a lot of feasible solutions, increasing the probability to obtain good ones. Fig. 4 displays for example the solution obtained by the proposed algorithm (IA) on the largest instance that contains 200 circles. The new best length is $L = 64.6904$, the previous best known value in the literature was $L = 64.7246$.

VII. CONCLUSION

In this paper an improved algorithm, denoted by IA, was proposed in order to solve the strip packing problem. IA is a beam-search based algorithm that includes a look-ahead strategy in order to improve the selection mechanism in each

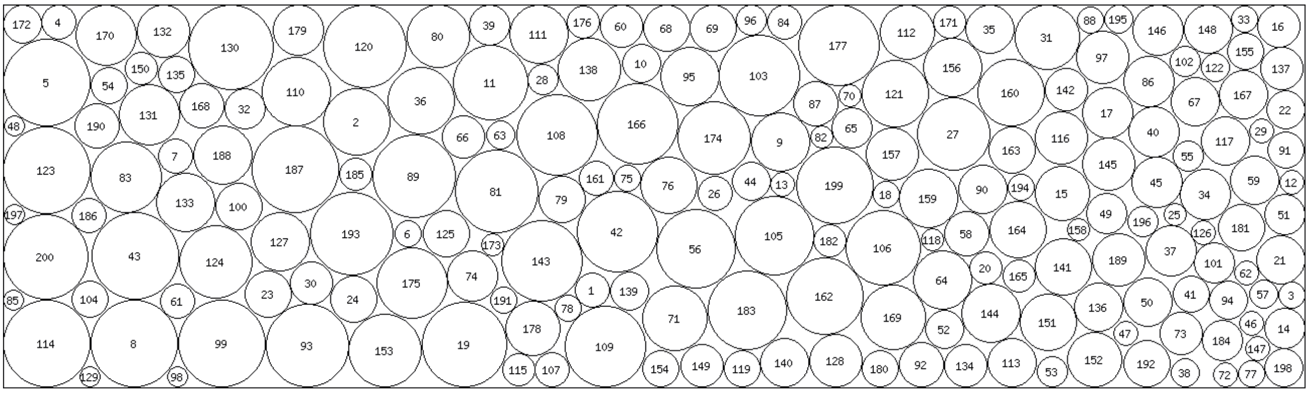

Fig. 4. Solution obtained by the proposed algorithm IA on the largest instance SY56 $(n = 200, m = 193, L = 64.6904)$

level of the tree. In addition, a restarting strategy was also used.

The computational investigation, conducted on a set of well-known instances in the literature, showed the effectiveness of the proposed algorithm since it has succeeded to improve 67% of the best known solutions in the literature. In addition, another experimentation has indicated that the look-ahead obtains good solutions more quickly, i.e., faster than the existing beam-search based algorithms.

As a future work, it would be interesting to use a parallel algorithm in order to reduce the computation time.

References

[1] E. Baltacioglu, J. T. Moore, and R. R. Hill, "The distributor's three-dimensional pallet-packing problem: a human intelligence-based heuristic approach," *Int. J. Oper. Res.*, vol. 1, 2006, pp. 249–266.

[2] J.H. Conway and N.J.A. Sloane, "Sphere packings, lattices and groups," *A Series of comprehensive studies in Mathematics*, Springer, vol. 290, 1999, 703 pages.

[3] R. Lewis, S. Song, K. Dowsland, and J. Thompson, "An investigation into two bin packing problems with ordering and orientation implications," *Eur. J. Oper. Res.*, vol. 213, 2011, pp. 52–65.

[4] S. Menon and L. Schrage, "Order allocation for stock cutting in the paper industry," *Oper. Res.*, vol. 50, 2002, pp. 324–332.

[5] G. Wäscher, H. Haussner, and H. Schumann, "An improved typology of cutting and packing problems," *Eur. J. Oper. Res.*, vol. 183, 2007, pp. 1109–1130.

[6] W. Q. Huang, Y. Li, C. M. Li, and R. C. Xu, "New heuristics for packing unequal circles into a circular container", *Comput. Oper. Res.*, vol. 33, 2006, pp. 2125–2142.

[7] M. Hifi, R. M'Hallah, "A dynamic adaptive local search algorithm for the circular packing problem," *European J. Oper. Res.*, vol. 183, 2007, pp. 1280–1294.

[8] H. Akeb, M. Hifi, and R. M'Hallah, "A beam search based algorithm for the circular packing problem," *Comput. Oper. Res.*, vol. 36, 2009, pp. 1513–1528.

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

[10] Y. G. Stoyan and G. N. Yaskov, "Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints," *Int. Trans. Oper. Res*, vol. 5, 1998, pp. 45–57.

[11] M. Hifi and R. M'Hallah, "Approximate algorithms for constrained circular cutting problems," *Comput. Oper. Res.*, vol. 31, 2004, pp. 675–694.

[12] 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.* vol. 56, 2005, pp. 539–548.

[13] E. G. Birgin, J. M. Martinez, and D. P. Ronconi, "Optimizing the packing of cylinders into a rectangular container: A nonlinear approach," *Eur. J. Oper. Res.*, vol. 160, 2005, pp. 19–33.

[14] T. Kubach, A. Bortfeldt, and H. Gehring, "Parallel greedy algorithms for packing unequal circles into a strip or a rectangle," *Cent. Eur. J. Oper. Res.*, vol. 17, 2009, pp. 461–477.

[15] H. Akeb, M. Hifi, and S. Negre, "An augmented beam search-based algorithm for the circular open dimension problem," *Comput. Ind. Eng.*, vol. 61, 2011, pp. 373–381.

[16] P. S. Ow, and T. E. Morton, "Filtered beam search in scheduling," *Int. J. Prod. Res*, vol. 26, 1988, pp. 35–62.

[17] H. Akeb and M. Hifi, "Algorithms for the circular two-dimensional open dimension problem," *Int. Trans. Oper. Res.*, vol. 15, 2008, pp. 685–704.