World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:4, No:6, 2010

An augmented beam-search based algorithm for the strip packing problem

Hakim Akeb and Mhand Hifi

Abstract—In this paper, the use of beam search and look-ahead strategies for solving the strip packing problem (SPP) is investigated. Given a strip of fixed width W, unlimited length L, and a set of n circular pieces of known radii, the objective is to determine the minimum length of the initial strip that packs all the pieces. An augmented algorithm which combines beam search and a look-ahead strategies is proposed. The look-ahead is used in order to evaluate the nodes at each level of the tree search. The best nodes are then retained for branching. The computational investigation showed that the proposed augmented algorithm is able to improve the best known solutions of the literature on most instances used.

Keywords—combinatorial optimization, cutting and packing, beam search, heuristic, look-ahead strategy.

I. Introduction

Cutting-and-packing problems can be encountered in several real applications: logistics (Cochran and Ramanujam [5]), manufacturing and production processes (Baltacioglu *et al.* [2], Burke *et al.* [4]), and industrial engineering (Menon and Schrage [13]). A cutting (or packing) problem consists to cut (or pack) a set of items of known dimensions from (or into) one or more larger objects (or containers) in order to minimize the unused part of the objects (or waste). The items and the containers may have rectangular, circular, or irregular forms. In this paper, the strip packing problem (SPP) is studied, where the items are circular and the container is represented by a strip (see Wäscher *et al.* [17]).

In SPP, an initial strip S of fixed width W and unlimited length L is given, as well as a finite set N of n small circular pieces C_i of known radius $r_i,\ i \in N = \{1, \ldots, n\}$. The problem is then to pack (or cut) all pieces such that:

(i) there is no overlapping between pieces, and between pieces and the edges of the strip and,
(ii) the aim is to minimize the length to which the initial strip S is filled, as well as to determine the coordinates (x_i, y_i) of the center of $C_i,\ i \in N$.

The SPP can be stated as follows:

$$\begin{aligned}
&\min && L && \\
&\text{subject to} && && \\
&(x_i - x_j)^2 + (y_i - y_j)^2 && \geq && (r_i + r_j)^2,\ j < i,\ (i,j) \in N^2 && (1)\\
&x_i - r_i && \geq && 0,\ \forall i \in N && (2)\\
&y_i - r_i && \geq && 0,\ \forall i \in N && (3)\\
&W - y_i - r_i && \geq && 0,\ \forall i \in N && (4)\\
&L - x_i - r_i && \geq && 0,\ \forall i \in N && (5)\\
&L && \geq && \underline{L} && (6)
\end{aligned}$$

H. Akeb is a Professor at ISC Paris, School of Management, 22 boulevard du Fort de Vaux, 75017 Paris, France. (hakeb@groupeisc.com)

M. Hifi is a Professor at Université de Picardie Jules Verne, Équipe ROAD, UR MIS, 33 rue Saint-Leu, 80039 Amiens, France. (mhand.hifi@u-picardie.fr)

where $\underline{L} = \left(\pi \times \sum_{i \in N} r_i^2\right)/W$. Equation (1) denotes the non-overlap constraint of any pair of distinct pieces (C_i, C_j). It means that the distance between the center of both pieces must be greater than or equal to the sum of the radii of C_i and C_j (there are $n(n-1)/2$ of these non-overlap constraints). Equations (2), (3), (4) and (5) state that any piece $C_i,\ i \in N$, does not exceed the target-rectangle boundary of dimensions (L, W). Equation (6) means that the optimal solution is bounded by the simple lower bound $\underline{L}$. Hence, SPP is characterized by a linear function, by four linear constraint types and by a nonlinear constraint type.

The remainder of the paper is organized as follows. Section II provides a review of SPP related literature. Section III presents the main principle of the beam search, a truncated branch and bound. Section IV summarizes the principle of the minimum local-distance position (Section IV-B) and the adaptive beam search algorithm (Section IV-D) already used in Akeb and Hifi [1]. Section V details the main steps of the augmented algorithm that combines the beam search with a look-ahead-based heuristic and a series of target values. In Section VI, the performances of the proposed algorithm are evaluated on a set of benchmark instances. Finally, Section VII summarizes the contributions of this paper.

II. Related literature

To our knowledge, most published papers for the circular cutting/packing problem can be categorized into two categories: (i) cutting/packing (non)identical circles into a large circle and, (ii) cutting/packing (non)identical circles into a rectangle or a strip.

For the first category, the problem of packing non-identical circles into a containing circle of known radius was studied by several authors (see for instance, Hifi and M'Hallah [7], Huang *et al.* [11] and Sugihara *et al.* [16]). For the second category, most of the proposed papers solve variants of the packing problem, in particular, packing circles into a rectangle or a strip. George *et al.* [6] designed an approach based upon several building rules simulating the packing operation; the best rules use a quasi-random approach and a genetic algorithm. Birgin *et al.* [3] tackled the same problem using a non-linear approach-based algorithm. Stoyan and Yaskov ([15]) solved the SPP using a mathematical model that searches for feasible local optima by combining a tree-search procedure and a reduced-gradient method. Hifi and M'Hallah [8] proposed a constructive procedure and a genetic algorithm to pack circles into a strip. Huang *et al.*[10] proposed two solution procedures for the SPP, called B1.0 and B1.5. Finally, Akeb and Hifi [1] proposed three algorithms for the SPP: (i) an

open strip generation solution procedure which is based on an optimization problem, (ii) a local beam-search solution procedure that combines beam search and the open strip generation procedure and (iii) a hybrid algorithm that combines beam search, dichotomous interval search and the open strip generation procedure.

In this paper, the SPP is solved using an augmented algorithm which combines a constructive solution procedure, beam search, dichotomous search and a look-ahead strategy.

III. A STANDARD BEAM SEARCH

Beam Search (BS) is a classical tree-search method that was introduced in the context of scheduling (Ow and Morton [14]), but has since then been successfully applied to many other combinatorial optimization problems (Kim and Kim [12], Akeb and Hifi [1], and Hifi *et al.* [9]). It is based on a truncated branch and bound, where at each level of the search tree only a subset of *promising* nodes are selected for further branching. The other nodes are ignored, and no backtracking is performed. The number ω of promising nodes to be investigated at each level is called the *beam width*. The selected nodes are those having a high potential to lead to the optimal solution. The potential of a node is assessed via an *evaluation operator* whose role is to provide a good separation mechanism of the nodes at each level of the search tree.

`Initialization Phase`

1) Let ω be the beam width.
2) If an initial feasible solution is available, set z^* to its objective function value; otherwise, set $z^* = -\infty$.
3) Set $B = \{B_0\}$ and $B_\omega = \emptyset$, where B is the set of nodes to be investigated, and B_ω the set of nodes branched out of the nodes in B.

`Iterative Phase`
`Repeat`

4) Choose a node $\mu \in B$; branch out μ; remove μ from B and insert the created nodes (i.e., the offsprings of μ) into B_ω.
5) If a node μ of B_ω is a leaf, then
   a) compute its objective function value z_μ;
   b) if $z_\mu > z^*$, update z^* and the incumbent solution;
   c) remove μ from B_ω.
6) Assess the potential of each node of B_ω using an evaluation operator.
7) Rank the nodes of B_ω in a non-increasing order of their values.
8) Insert the $\min\{\omega, |B_\omega|\}$ best nodes of B_ω into B; and set $B_\omega = \emptyset$.

`Until` $B = \emptyset$.

Fig. 1. A standard beam search solution procedure.

BS is characterized by the beam width ω, which is a constant used for filtering the set of offspring nodes B_ω (step 1 in the `Initialization phase`). As with branch and bound, an upper bound can be used to fathom nodes. If a starting feasible solution is available, then it is set as the incumbent solution and its value is assigned to z^* (step 2). On the other hand, if no initial feasible solution is available, z^* is set to $-\infty$ (assuming that the problem at hand is a maximization problem). A node corresponds to a partial feasible solution. The set B of the current nodes is initialized to the root node B_0 whereas the set B_ω of the offspring nodes is initialized to the empty set (step 3 of the `Initialization Phase`). A current node of B generates a set of offspring nodes, and adds them to B_ω (step 4 in the `Iterative Phase`). If a node μ of B_ω is a leaf (i.e, no further branching is possible out of μ), then its objective function value z_μ is computed and compared to z^* (Step 5 of the `Iterative Phase`). If $z_\mu > z^*$, then the incumbent solution is set to the leaf node; z^* is then updated: $z^* = z_\mu$; and μ is removed from B_ω. The nodes of B_ω are after that assessed using an evaluation operator (step 6), and ranked in a non-ascending order of their values (step 7). The first ω nodes of B_ω are then chosen as the *elite nodes* and are appended to B, the set of nodes to be further investigated, whereas the remaining nodes of B_ω are fathomed and B_ω is reset to the empty set (step 8). This process is repeated until no further branching is possible, i.e. $B = \emptyset$.

IV. ADAPTING THE BEAM SEARCH FOR THE SPP

This section explains how to adapt the beam search strategy fo the strip packing problem. Section IV-B describes the principle of the Minimum Local-Distance Position (MLDP) used by Akeb and Hifi [1] as a constructive strategy for the algorithm. Section IV-D summarizes Akeb and Hifi's dichotomous beam search adapted for approximately solving the SPP.

A. Notations

In the rest of the paper, the following notations are used:

(i) $N = 1, ..., n$ denotes the set of circles to pack,
(ii) the strip S is placed with its bottom left corner at $(0, 0)$,
(iii) the four edges of S are denoted by S_{left}, S_{top}, S_{right} and S_{bottom},
(iv) each circular piece C_i of radius r_i is placed with its centre at (x_i, y_i),
(v) I_i represents the set of the i circles already packed inside the strip,
(vi) $\overline{I}_i$ contains the circles outside the strip (not yet placed),
(vii) P_{I_i} denotes the set of distinct corner positions of C_{i+1} given the set I_i,
(viii) a corner position $p_{i+1} \in P_{I_i}$ is determined by using two elements a and b. An element is either a piece already placed 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 a and b.

B. The MLDP strategy

MLDP can be viewed as a greedy strategy used for selecting a corner position among a set of feasible positions. The principle of the selection strategy follows. Having positioned a set $I_i \subseteq N$ of circular pieces, the process tries to position the next piece, namely $C_{i+1}, i \in N \backslash I_i$, into a corner position among all its eligible positions in the strip S −i.e., without overlapping any of the pieces already placed.

Figure 2 illustrates such corner positions −dotted-line circular pieces− for the piece C_3, where $I_2 = \{C_1, C_2\}$ and

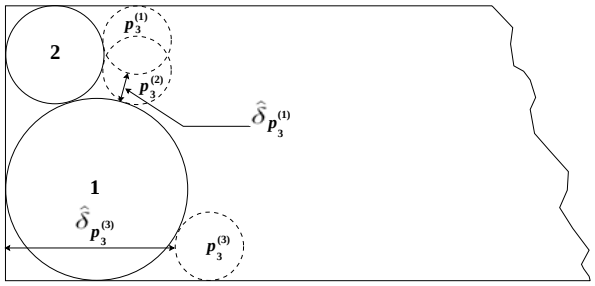

Fig. 2. Feasible distinct corner positions of C_3 in the strip S.

$P_{I_2} = \{p_3^{(1)}, p_3^{(2)}, p_3^{(3)}\}$. The notation $p_3^{(k)}$, for $k = 1, \ldots, 3$, denotes the k^{th} corner position such that $p_3^{(k)} \in I_2$. Here, the corner position $p_3^{(1)}$ is generated by using the piece C_2 and the top-edge of the strip, $p_3^{(2)}$ is provided by using both pieces C_1 and C_2 and $p_3^{(3)}$ is obtained by using C_1 and the bottom-edge of the strip. It follows that $T_{p_3^{(1)}} = \{C_2, S_{\rm top}\}$, $T_{p_3^{(2)}} = \{C_1, C_2\}$ and $T_{p_3^{(3)}} = \{C_1, S_{\rm bottom}\}$.

Let C_{i+1} be the selected circular piece to pack at position p_{i+1} and $\delta_{i+1}(edge)$, $edge \in E_{\rm edge} = \{S_{\rm left}, S_{\rm bottom}, S_{\rm top}\}$, be the three distances defined as follows: $\delta_{i+1}(S_{\rm left}) = x_i - r_i$, $\delta_{i+1}(S_{\rm bottom}) = y_i - r_i$,and $\delta_{i+1}(S_{\rm top}) = W - y_i - r_i$.

Consider also the distance $\delta_{i+1}(j)$, which separates circles C_{i+1} (when positioned at p_{i+1}) and C_j, defined as follows:

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

Then, the MLDP of a piece C_{i+1} when positioned at $p_{i+1} \in P_{I_i}$ is:

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

Note that, on the one hand, the computation of the MLDP of the piece C_{i+1} needs to compute all the distances from each position $p_{i+1}^{(k)}$ of P_{I_i} to the pieces already placed I_i augmented by the three edges of the strip S, and by excluding the elements of $T_{p_{i+1}^{(k)}}$ (since the last distance is always reduced to zero). On the other hand, when the piece C_{i+1} touches more than two elements, then the MLDP is reduced to zero;

Figure 2 indicates that the minimum local distance between C_3 and the already-packed circular pieces, when positioned in $p_3^{(1)}$ and $p_3^{(3)}$, are $\hat{\delta}_{p_3^{(1)}}$ and $\hat{\delta}_{p_3^{(3)}}$ respectively.

Specifically, for a pre-determined ordering of the pieces, the solution procedure starts by positioning the first circular piece C_1 at the bottom-left corner *i.e.*, at the position (r_1, r_1) whereas the remaining $n-1$ pieces are successively positioned using the MLDP rule.

C. Open strip generation procedure

In order to define an upper limit for the target-rectangle length, an open strip generation procedure (OSGSP) was proposed in Akeb and Hifi [1]. The procedure considers the strip as a rectangle with an unlimited length (an open strip) bounded by the left, top and bottom edges. OSGSP places the first circle of the instance C_1 in the bottom-left corner of the open strip and uses after that the MLDP procedure in order to place the remaining $n-1$ circles. OSGSP exits with the upper limit of the strip length $\overline{L}$ which can be used as un upper bound in a dichotomous search algorithm like BSBIS (Figure 3).

D. An adaptive dichotomous beam search for the SPP

In Akeb and Hifi [1], combining the dichotomous search and beam search yielded a Dichotomous width-beam search algorithm denoted by BSBIS (Beam-Search Binary-Interval Search). BSBIS uses two solution procedures: (i) a procedure that uses an interval search $[\underline{L}, \overline{L}]$, and (ii) a procedure Beam-Search (denoted by BS) which tries to pack all the pieces in the successive target-rectangles, where each target-rectangle (L^*, W) is such that $L^* \in [\underline{L}, \overline{L}]$.

Figure 3 describes the main steps of BSBIS and its principle can be summarized by the following points (for more details, the reader can refer to Akeb and Hifi [1]):

(i) The starting interval search is initialized with an upper limit $\overline{L}$ (obtained by applying the Open Strip Generation Solution Procedure (OSGSP) summarized above) and a natural lower bound $\underline{L} = \left(\pi \times \sum_{i \in N} r_i^2\right)/W$.
(ii) The lower limit $\underline{L}$ is updated every time the packing procedure BS (detailed in Figure 1) yields an infeasible solution for the current target-rectangle (L^*, W), where $L^* \in [\underline{L}, \overline{L}]$.
(iii) The upper limit $\overline{L}$ is set equal to the current length of the rectangle every time the aforementioned packing procedure BS yields a feasible solution.
(iv) The dichotomous search is stopped when the width of the interval search becomes smaller than a given tolerance gap δ; i.e., when $\overline{L} - \underline{L} \leq \delta$.

The packing procedure BS (detailed in Figure 1) builds a set of favorite nodes following MLDP rule. It receives two parameters: η_ℓ, the node at level ℓ, and $\omega \in \mathbb{N}^+$, the beam width. Depending on the considered node, the procedure considers what follows.

Input. A beam width ω.
Output. A feasible packing containing the n circles and the best length L^* for the strip.

`Initialization Phase.`
1) Assume that the pieces of N are ranked following the decreasing value of their radii.
2) Let $[\underline{L}, \overline{L}]$ be the starting interval and δ be the tolerance gap of the dichotomous search.
3) Set $L^* = \overline{L}$, where L^* denotes the best solution found so far.

`Iterative Phase.`
`while` $(\overline{L} - \underline{L} > \delta)$ `do`
4) Set $L^* = (\overline{L} + \underline{L})/2$.
5) Generate the node η_1 characterized by $I_1 = \{C_1\}$, $\overline{I}_1 = N \setminus \{C_1, C_k\}, k \in \{2, \ldots, n\}$, and P_{I_1}.
6) Set `Feasible` = BS(η_1, ω)
7) `if Feasible=true` {*the n pieces have been packed*}
`then` set $\overline{L} = L^*$, `otherwise` set $\underline{L} = L^*$.

`enddo`

Fig. 3. Dichotomous Beam Search algorithm (BSBIS)

The first node η_1, corresponding to the first level, is composed of (**i**) $I_1 = \{C_1\}$, where C_1 is packed at the

bottom-left corner of the current rectangle (L^*, W); that is tangent to both edges S_{left} and S_{bottom} at position (r_1, r_1), **(ii)** $\overline{I}_1 = N \setminus \{C_1, C_2\}$ the set of $n-2$ circles that have not yet been considered for packing, and **(iii)** P_{I_1}, the set of feasible distinct corner positions of C_2 given the position of C_1.

An internal node, corresponding to the node at level ℓ, is characterized by **(i)** I_ℓ, the set of already-positioned circles, **(ii)** P_{I_ℓ}, the set of feasible eligible corner positions of the circle $C_{\ell+1}$ to be packed, and **(iii)** $\overline{I}_\ell$, the set of circles that remain to be packed; *i.e.*, $\overline{I}_\ell = N \setminus I_\ell \setminus \{C_{\ell+1}\}$.

At a level $\ell \geq 1$, branching out of a node becomes equivalent to choosing a position in P_{I_ℓ} for $C_{\ell+1}$. There are as many possible branches out of a node as there are elements in P_{I_ℓ}. Moreover, out of all the nodes at level ℓ, the ω positions having the smallest MLDPs are chosen as branches to be further investigated. All other nodes of level ℓ are fathomed. In addition, a node whose $P_{I_\ell} = \emptyset$ is fathomed since it does not lead to a feasible packing of the n circles into (L^*, W). Also, a node whose $\overline{I}_\ell = \emptyset$ is a leaf; that is, the n circles have been packed into (L^*, W), and a feasible solution is at hand. Such a node has a level $\ell = n$.

Input. A node η_ℓ and the beam width ω.
Output. `Feasible=true` if a feasible packing into the target-rectangle (L^*, W) is obtained; and `Feasible=false` otherwise.

`Initialization Step.`
1) Let B (resp. B_ω) be the set of current (resp. offspring) nodes.
2) Set $B = \{\eta_\ell\}$, where η_ℓ is a node of level ℓ characterized by I_ℓ, $\overline{I}_\ell$, and P_{I_ℓ}.
3) Set `Feasible = false`.

`Recursive Step.`
`while` $B \neq \emptyset$ `do`
4) Set $\ell = \ell + 1$.
5) Set B equal to the $\min\{|B_\omega|, \omega\}$ nodes of B_ω having the best MDLPs and corresponding to distinct feasible corner positions, and reset $B_\omega = \emptyset$.
6) `for each` selected node of level ℓ, `do`
   a) Pack $C_{\ell+1}$ into its selected position, and remove the node from B.
   b) `if` $\ell + 1 = n$, `then` a feasible solution is at hand; `Feasible=true`; `exit`.
   c) Determine $P_{I_{\ell+1}}$, the set of potential distinct corner positions for $C_{\ell+2}$ given the positions of the already-packed circles of $I_{\ell+1}$.
   d) `if` $P_{I_{\ell+1}} = \emptyset$, `then` the node does not lead to a feasible solution for the target-rectangle; fathom the node.
   e) `else`
      i) Create $|P_{I_{\ell+1}}|$ branches out of the node; each corresponding to a node characterized by the potential position of $C_{\ell+2}$ given the positions of the already-packed circles of $I_{\ell+1}$.
      ii) Append the created offspring nodes to B_ω not allowing the duplication of existing corner positions.

`enddo`
`enddo`

Fig. 4. The beam search as a packing procedure (BS)

V. An augmented algorithm for SPP

In this section, the principle of the look-ahead, which leads to a filling procedure for the SPP, is described in Section V-A. In Section V-B, the augmented algorithm, denoted by ADBS (Augmented Dichotomous Beam Search), is exposed. ADBS combines the filling procedure, beam search, and a dichotomous interval search.

A. A filling procedure

Observe that the MLDP strategy can be used as a greedy procedure for searching for a quick solution. In this section, a *filling procedure* (FP), which combines MLDP and the look-ahead strategy, is exposed.

A look-ahead strategy, that may be more time-consuming but may produce better results, can be viewed as a particular case of the forward phase in branch and bound solution procedures. The forward strategy tries to explore all directions according to the positions associated to each decision variable (circular piece) whereas the look-ahead selects the best position according to the complementary lower bound used.

Input. A list $B = \{\eta_\ell^1, ..., \eta_\ell^\omega\}$ of ω nodes and a boolean variable `feasible`.
Output. Either a feasible solution corresponding to `feasible=true` or a set B_ω of ω nodes (those leading to the highest densities through the MLDP packing procedure).

`Starting Phase.`
Let P_{ℓ_i} be the set of corner positions of node $\eta_\ell^i \in B$
Set $B_\omega = \emptyset$
Set `feasible = false`.
`Iterative Phase.`
`for each` node η_ℓ^i of B `do`
`for each` corner position $p_i \in P_{\ell_i}$ `do`
1) Pack C_{i+1} in p_i and insert the resulting node $\eta_{\ell+1}$ into B_ω.
2) Evaluate the new inserted node $\eta_{\ell+1}$ by placing the remaining circles using the MLDP packing procedure.
3) `if` all circles are placed `then` set `feasible = true`, `exit`.
`else` assign to $\eta_{\ell+1}$ the density obtained by MLDP.
`enddo`
`enddo`
`Terminal Phase.`
4) Reduce B_ω to the ω nodes that led to the highest densities using MLDP.
5) Exit with B_ω.

Fig. 5. The filling procedure (FP)

The objective of the filling procedure FP, described in Figure 5, is to assess the potential of the nodes of the current level of the search tree. It is called at each iteration in the `while` loop of algorithm ADBS described in Figure 6. This strategy can be viewed as a *global search strategy* since it allows the choice of the expanding paths according to the best global solutions reached by applying the look-ahead.

The main steps of FP, are described in Figure 5. FP receives two parameters. The first one ($B = \{\eta_\ell^1, ..., \eta_\ell^\omega\}$) corresponds to the nodes of the current level of the beam-search tree, and the second one (`feasible`) is a boolean value. In the

`Iterative Phase`, the procedure creates as many nodes as there are positions in B by positioning the next circle at each position. This creates a list of offspring nodes B_ω. The MLDP packing procedure evaluates each node of B_ω by applying the MLDP rule. Here, two cases can be distinguished. First, FP obtains a feasible solution (Figure 5, step 3 of the `Iterative Phase`), i.e. all remaining pieces are packed into the target-rectangle (L^*, W); thus, `feasible` is set to `true` and FP exits with a solution. Second, FP does not obtain a solution after running MLDP on all nodes of B_ω, then the best nodes leading to the highest densities obtained by using the MLDP rule are returned, and these are used by algorithm ADBS (Figure 6) for branching. Note that the density of a node is equal to the sum of the surfaces of the circles placed divided by the surface of the target rectangle (L^*, W).

B. An augmented beam search algorithm

The large number of positions that can be produced by the search process excludes an exhaustive search. So in the augmented algorithm, the beam width can constrain the number of branches to explore, and the look-ahead-based filling procedure attempts to compensate for this restriction imposed on the search.

Indeed, the proposed augmented algorithm ADBS combines the dichotomous beam-search algorithm (BSBIS, Section IV), the filling procedure (FP, Section V-A) and an incremental upper bound limit for curtailing the search process.

Input. A node η_ℓ and a beam width ω.
Output. A feasible packing and the best corresponding value for the rectangle length (L_{best}).

`Initialization Phase.`
- Let $[\underline{L},\ \overline{L}]$ be the starting interval and δ be the tolerance gap of the dichotomous search.
- Let B and B_ω be the sets of nodes to be considered and the offspring nodes of the node currently being considered.
- Let L_{best} be the best length found so far.
- Let `feasible` be a boolean variable.

`Iterative Phase.`

`while` $(\overline{L} - \underline{L} > \delta)$ `do`
1) 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}.
2) Set $L^* = (\overline{L} + \underline{L})/2$.
3) Set `feasible = false`
   `while` $(B \neq \emptyset)$ and (`feasible=false`) `do`
   a) Set $\ell = \ell + 1$.
   b) Set B_ω = FP(B, `feasible`).
   c) `if feasible=true then` set $\overline{L} = L^*$ and $L_{\text{best}} = L^*$ `else` set $B = B_\omega$ and $B_\omega = \emptyset$.

   `enddo`
4) `if feasible=false then` set $\underline{L} = L^*$.
`enddo`

Fig. 6. Augmented Dichotomous Beam Search algorithm (ADBS)

As shown in Section IV, a node is defined by the pair of subsets I_i and $\overline{I}_i$, where C_{i+1} is the current circular piece to pack. The first set I_i can be viewed as *a local partial subset* and $\overline{I}_i$ the *complementary subset*. The pieces of I_i have already been positioned whereas the pieces of $\bar{I}_i$ remains to be positioned. Positioning all the circular pieces is then equivalent to branching out of the node $\left(I_i, \bar{I}_i\right)$. Of course, in some cases only a subset of the remaining pieces can be positioned into the target-rectangle (L^*, W). Algorithm ADBS, described in Figure 6, combines the dichotomous beam search (algorithm BSBIS of Figure 3) with the filling procedure (FP, described in Figure 5) in order to try to achieve better solutions.

Figure 6 describes the main steps of algorithm ADBS applied to the SPP. It is composed of two main phases. The `Initialization Phase` in which the starting interval search is initialized with its lower and upper limits $\underline{L}$ and $\overline{L}$. The best length L_{best} is equal to the best length found so far (if no solution is known, then $L_{\text{best}} = \overline{L}$).

The second phase of ADBS is the `Iterative Phase`. It contains two nested `while` loops. The first `while` loop corresponds to steps 1, 2, 3, and 4. Step 1 initializes the set B to the current node η_ℓ. Step 2 computes the current target lenght of the rectangle (L^*, W). Step 3 initializes the value of the boolean variable `feasible` to false (`feasible` indicates if a feasible packing into the target rectangle (L^*, W) is obtained). After step 3, the second `while` loop takes place. In step (a) the level ℓ of the searh tree is incremented. ADBS calls after that the filling procedure FP at step (b) in order to evaluate each position of each node of the current level. In step (c), two cases may be destinguished. If parameter `feasible` has the value `true`, then the procedure FP reaches a feasible solution for the current target rectangle (L^*, W). In this case, the upper bound $\overline{L}$ is set equal to L^* and L_{best} is updated. If the value of `feasible` is equal to `false` then the procedure FP does not reach a feasible solution by using the look-ahead strategy, in this case the list B_ω returned by this procedure contains the best ω expanded nodes. The nodes of B_ω then replace the nodes of B.

Finally in step 4, if the value of parameter `feasible` if `false` then no feasible packing is obtained by the second `while` loop for the current value L^* (the target rectangle is not large enough). In this case, the lower bound $\underline{L}$ is updated and set to L^*.

Algorithm ADBS stops when $(\overline{L} - \underline{L} \leq \delta)$ and the algorithm'output corresponds to a feasible packing containing the n circles placed inside the rectangle (L_{best}, W).

VI. Computational results

The objective of the computational investigation is to assess the performance of the proposed algorithm. The different tested algorithms are coded in C language and run on a 3-GHz Intel Celeron, with 256 MB of RAM.

Note that, 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. Of course, a different adjustment of the method's parameters would lead to a higher percentage of good solutions. But this better adjustment would sometimes lead to onerous execution-time requirements. The various algorithms use two decision parameters: (i) the value ω associated to

the beam width and (ii) the width of the interval search associated to the limits of the target-rectangle length L. On the one hand, as shown in [1] for BSBIS algorithm, a higher value associated to ω does not necessarily result in better solutions but it considerably influences the degree. In order to maintain the same degree of comparison, the same starting limits corresponding to $\underline{L}$ and $\overline{L}$, as discussed in Section IV, are considered.

A. Sets of instances used

All algorithms were tested on two sets of instances. The first set contains the six instances taken from Stoyan and Yaskov [15] and identified as SY1, SY2, SY3, SY4, SY5, and SY6. These instances contain between 20 and 100 circles. The other twelve problems, forming the second set, are taken from Akeb and Hifi [1], and are obtained by concatenating the six original problems of the first set. The problems of the second set are identified as SY12, SY13, SY14, SY23, SY24, SY34, SY56, SY123, SY124, SY134, SY234, and SY1234 and contain between 45 and 200 circles. In fact, the instances of the second set include relatively large and hard problems; thus, reflect the behavior of the proposed augmented algorithm ADBS when the problem size increases.

TABLE I
DESCRIPTION OF THE INSTANCES USED

Instance	n	m	$r_{\min}$	$r_{\max}$	W	$\underline{L}$	L_{OSGSP}
SY1	30	30	0.527	2.0500	9.5	14.5500	18.4136
SY2	20	20	0.566	2.1710	8.5	12.1600	16.8254
SY3	25	25	0.509	2.1470	9	12.2336	15.2054
SY4	35	35	0.606	2.1780	11	19.9073	24.9282
SY5	100	99	0.533	2.1864	15	31.2822	38.4229
SY6	100	98	0.509	2.1847	19	31.7844	39.7836
SY12	50	48	0.527	2.1710	9.5	25.4301	32.3336
SY13	55	54	0.509	2.1470	9.5	26.1398	33.3387
SY14	65	65	0.527	2.1780	11	32.4732	39.9354
SY23	45	45	0.509	2.1710	9	23.7181	30.2653
SY24	55	54	0.566	2.1780	11	29.3037	37.3802
SY34	60	59	0.509	2.1780	11	29.9166	36.4357
SY56	200	193	0.509	2.1864	19	56.4809	68.8995
SY123	75	72	0.509	2.1710	9.5	37.0198	45.8464
SY124	85	82	0.527	2.1780	11	41.8696	50.9251
SY134	90	88	0.509	2.1780	11	42.4826	52.3415
SY234	80	78	0.509	2.1780	11	39.3130	47.7496
SY1234	110	105	0.509	2.1780	11	51.8790	62.1011

Table I describes the caracteristics of the eighteen instances used. Column 1 idicates the instance's name. Column 2 contains the instance's size (n) and column 3 corresponds to the number of circle types (m) which is the number of inferent radii in the instance. Columns 4 and 5 indicate the smallest radius ($r_{\min}$) and the greatest radius ($r_{\max}$) in the instance. Column 6 displays the width of the strip (W). Finally, Columns 7 and 8 contain respectively the trivial lower bound of the target-rectangle length $\underline{L} = \frac{\pi}{W} \times \sum_{i=1}^{n}(r_i^2)$ and the upper limit of the length L_{OSGSP} computed by the OSGSP packing procedure as explained in Section IV-C.

B. Solution quality

TABLE II
SOLUTION QUALITY OF THE ALGORITHMS

Instance	n	Best	B1.0	B1.5	BSBIS	ADBS
SY1	30	17.2315	17.561	17.291	**17.2315**	17.3269
SY2	20	14.5350	14.735	**14.535**	14.6277	14.5837
SY3	25	14.4700	14.660	**14.470**	14.5310	14.5073
SY4	35	23.5550	23.915	**23.555**	23.6719	23.5794
SY5	100	36.0796	36.547	36.327	**36.0796**	36.1357
SY6	100	36.8456	36.997	36.857	36.8456	**36.7839**
SY12	50	29.7011	30.067	31.203	**29.7011**	30.0010
SY13	55	30.6371	30.891	32.217	30.6371	**30.5829**
SY14	65	38.0922	38.265	40.628	38.0922	**37.8367**
SY23	45	27.8708	28.270	28.664	27.8708	**27.7962**
SY24	55	34.5476	34.605	36.373	34.5476	**34.3350**
SY34	60	34.9011	34.901	37.296	34.9354	**34.8059**
SY56	200	64.7246	69.979	69.979	**64.7246**	65.0187
SY123	75	43.2558	43.626	45.457	43.2558	**43.2260**
SY124	85	48.8927	49.335	52.477	**48.8927**	48.9442
SY134	90	49.3954	49.721	53.399	49.3954	**49.2606**
SY234	80	45.8880	45.888	49.145	45.9526	**45.6405**
SY1234	110	60.2613	61.906	65.248	60.2613	**60.1844**

Four versions of the BSBIS algorithm (described in Figure 3) were used in [1]. These versions are denoted by $\mathrm{BSBIS_a}$, $\mathrm{BSBIS_b}$, $\mathrm{BSBIS_c}$, and $\mathrm{BSBIS_d}$, corresponding to a variation of the beam-width ω in the integer interval $[1, \ldots, 25]$ for the first version, $[1, \ldots, 50]$ for the second version, $[1, \ldots, 75]$ for the third version and $[1, \ldots, 100]$ for the fourth version. Obviously, the $\mathrm{BSBIS_d}$ version get the best results but with a larger average runtime. Note that, the results reported in [1] were compared to those obtained by B1.0 and B1.5; both algorithms are based upon the Maximum Hole Degree strategy (MHD) [10] for which only the results for the first set of instances (SY1,. . .,SY6) (see Table I) are available. Furthermore, in order to compare the performance of the proposed algorithms, both B1.0 and B1.5 are run on the second set containing twelve instances (described in Table I) by fixing the runtime to thirty hours.

Herein, in order to maintain the same degree of comparison, the beam width ω is taken in the interval $[1, \ldots, 30]$ for ADBS and each run is fixed to one hour. Moreover, the cumulative runtime for each algorithm does not exceed thirty hours for the thirty trials.

Table II displays the results reached by the proposed algorithm (ADBS) on all instances. The solution quality is compared to the best results of the literature that are extracted from [1]. These best results are obtained either by BSBIS [1] or by B1.5 [10] for the first set of instances SY1, . . ., SY6. Columns 1 and 2 contain the instance's name and the number of circles (n). Column 3 indicates the best known result in the literature (the best length of the target-rectangle). Columns 4 and 5 contain the results obtained by B1.0 and B1.5 respectively by fixing the runtime at thirty hours. Column 6 contains the best result obtained by the BSBIS algorithm (more precisely by $\mathrm{BSBIS_d}$) extracted from

[1]. Finally, Column 7 displays the results reached by the augmented algorithm ADBS (described in Figure 6).

Table II indicates that the BSBIS algorithm remains competitive, by providing better solutions, for the instances SY1, SY5, SY12, SY56, and SY124. The Maximum Hole Degree heuristic (algorithm B1.5) remains better on some instances (SY2, SY3, and SY4). But the proposed algorithm (ADBS) improves the best known results (column 3) in 10 occasions out of 18 (it improves BSBIS in 13 occasions and B15 in 14 occasions), and is thus the best one.

The study of the solution quality (Table II), shows that look-ahead-based algorithm (ADBS) is adapted for medium and large instances ($n > 35$) since it obtained the best results on the majority of these instances.

C. Computation time

TABLE III
COMPUTATION TIME OF THE VARIOUS ALGORITHMS

Instance	n	B1.0	B1.5	BSBIS	ADBS
SY1	30	292	–	1010	7836
SY2	20	27	22397	320	1351
SY3	25	108	–	545	3368
SY4	35	875	–	1482	13095
SY5	100	–	–	27607	93453
SY6	100	–	–	28299	93415
SY12	50	6970	30h	3779	48599
SY13	55	6883	30h	4734	60000
SY14	65	36858	30h	7657	73900
SY23	45	3164	30h	2576	34151
SY24	55	12185	30h	4913	60238
SY34	60	16735	30h	5698	65409
SY56	200	30h	30h	30h	101137
SY123	75	67298	30h	10480	83106
SY124	85	30h	30h	15237	88905
SY134	90	30h	30h	17549	88437
SY234	80	103492	30h	12791	85213

Table III reports a comparative study of the cumulative computation time (in seconds) realized by the considered algorithms. Columns 1 and 2 indicate the instance's name and its size (n). Columns 3 and 4 display the cumulative runtime taken by B1.0 and B1.5, respectively. The symbol "–" means that the algorithm needed more than thirty hours for attaining the solution reported in [10] (which is also displayed in Table II). Note that only the results for the first set of instances SY1, ..., SY6 are reported in [10]. The results obtained by B1.5 indicate that this algorithm is not efficient when the computation time is bounded; this phenomenon can be explained by the fact that the algorithm uses a heavier look-ahead strategy coupled with a time-consuming restarting strategy (see [10]). Column 5 of Table III indicates the cumulative computation time of BSBIS taken from [1]. Herein, BSBIS attains the limit of thirty hours when considering the largest instance SY56 that contains 200 circles. Finally, Column 6 corresponds to the augmented algorithm ADBS (that combines BSBIS and the look-ahead strategy). Of course, increasing the search space implies more computation time, as it is confirmed in Table III, but ADBS is able to improve the solution quality, as shown in Table II.

VII. CONCLUSION

This paper solves the strip packing problem using a beam search-based heuristic which is combined with a dichotomous interval search and a complementary partial solution, leading to an augmented algorithm. The computational investigation shows that the proposed algorithm is able to reach interesting solutions for all considered instances, varying from small to large-sized instances. The provided results highlight the importance of the choice of the evaluation operator that assesses the potential of each node in order to lead to good solutions.

REFERENCES

[1] H. Akeb, and M. Hifi, "Algorithms for the circular two-dimensional open dimension problem," *International Transactions in Operational Research*, vol. 15, pp. 685–704, 2008.
[2] E. Baltacioglu, J.T. Moore, and R. R. Hill, "The distributor's three-dimensional pallet-packing problem: a human intelligence-based heuristic approach," *International Journal of Operational Research*, vol. 1, pp. 249–266, 2006.
[3] E. G. Birgin, J. M. Martinez, and D. P. Ronconi 2005, "Optimizing the packing of cylinders into a rectangular container: A nonlinear approach," *European Journal of Operational Research*, vol. 160, pp. 19–33, 2005.
[4] E. Burke, R. Hellier, G. Kendall, and G. Whitwell, "A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem," *Operations Research*, vol. 54, pp. 587–601, 2006.
[5] J. K. Cochran, and B. Ramanujam, "Carrier-mode logistics optimization of inbound supply chains for electronics manufacturing," *International Journal of Production Economics*, vol. 103, pp. 826–840, 2006.
[6] J. A. George, J. M. George, and B. W. Lamar, "Packing different-sized circles into a rectangular container," *European Journal of Operational Research*, vol. 84, pp. 693–712, 1995.
[7] M. Hifi, and R. M'Hallah, "A dynamic adaptive local search based algorithm for the circular packing problem," *European Journal of Operational Research*, vol. 183, pp. 1280–1294, 2007.
[8] M. Hifi, and R. M'Hallah, "Approximate algorithms for constrained circular cutting problems," *Computers and Operations Research*, vol. 31, pp. 675–694, 2004.
[9] M. Hifi, R. M'Hallah, and T. Saadi, "Beam search algorithms for constrained two-staged two-dimensional cutting problems," *INFORMS Journal on Computing*, vol. 20, pp. 212–221, 2008.
[10] W. Q. Huang, Y. Li, H. Akeb, and C. M. Li, "Greedy algorithms for packing unequal circles into a rectangular container," *Journal of the Operational Research Society*, vol. 56, pp. 539–548, 2005.
[11] W. Q. Huang, Y. Li, C. M. Li, and R. C. Xu, "New heuristics for packing unequal circles into a circular container," *Computer and Operations Research*, vol. 33, pp. 2125–2142, 2006.
[12] K. H. Kim, and J. B. Kim, . "Determining load patterns for the delivery of assembly components under JIT systems," *International Journal of Production Economics*, vol. 77, pp. 25–38, 2002.
[13] S. Menon, and L. Schrage, "Order allocation for stock cutting in the paper industry," *Operations Research*, vol. 50, pp. 324–332, 2002.
[14] P. S. Ow, and T. E. Morton, "Filtered beam search in scheduling," *International Journal of Production Research*, vol. 26, pp. 35–62, 1988.
[15] 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," *International Transactions in Operational Research*, vol. 5, pp. 45–57, 1998.
[16] K. Sugihara, M. Sawai, H. Sano, D. S. Kim, and D. Kim, "Disk packing for the estimation of the size of wire bundle," *Japan Journal on Industrial and Applied Mathematics*, vol. 21, pp. 259–278, 2004.
[17] G. Wäscher, H. Haussner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, vol. 183, pp. 1109–1130, 2007.