7e Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 – Paris- France
« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination.».

SOLVING THE CIRCULAR PACKING PROBLEM USING A BEAM SEARCH APPROACH-BASED ALGORITHM

Hakim AKEB

Institut Supérieur du Commerce
22, Bvd. du Fort de Vaux
75017 Paris
LaRIA, CNRS FRE 2733
Université de Picardie Jules Verne
33, rue Saint-Leu
80039 Amiens, France
hakim.akeb@u-picardie.fr

Mhand HIFI

LaRIA, CNRS FRE 2733
Université de Picardie Jules Verne
33, rue Saint-Leu
80039 Amiens, France
mhand.hifi@u-picardie.fr

Rym M'HALLAH

Kuwait University
Stat & OR Dept.
PO Box 5969, 16080 Safat
Kuwait
mhallah@kuc01.kuniv.edu.kw

ABSTRACT : *This paper studies the circular packing problem (CPP) whose objective is to pack n non-identical circles C_i of known radius r_i, $i \in N = \{1, \ldots, n\}$, into the smallest containing circle C. The problem determines the radius r of C as well as the coordinates (x_i, y_i) of the center of C_i, $i \in N$. This problem, which is a variant of the two dimensional open dimension problem, is solved using a beam search with decisions at each level of the tree being based on the so-called maximum hole degree. The computational results show the effectiveness of the proposed approach.*

KEY-WORDS : *circular packing, beam search, maximum hole degree*

1. INTRODUCTION

Cutting and Packing (C&P) problems are scientifically challenging problems with a wide spectrum of applications. They are very interesting NP-hard combinatorial optimization problems; i.e., no procedure is able to exactly solve C&P problems in polynomial time. They are encountered in a variety of real world applications including production and packing for the textile, apparel, naval, automobile, aerospace, and food industries. They generally consist of packing a set of items of known dimensions into one or more large objects or containers as to minimize the unused part of the objects or waste. The items and objects can be rectangular, circular or irregular.

This paper addresses the problem of packing not necessarily identical circles into the smallest containing circle. This problem is known as CPP, the circular packing problem. CPP is of particular interest in bundling wires that connect a car's sensors to the display board (Sugihara *et al.*, 2004), (Hifi et M'Hallah, 2007b). The wires have to pass through a hole that will be drilled on the body of the car. The hole has to be large enough to allow all wires to pass, but as small as possible to avoid unnecessarily weakening the body. This problem is similarly of interest to the telecommunication / electrical / oil companies and refineries which have to pass bundles of different types of cables through cylindrical shapes over thousands of kilometers. The smaller the diameters of the cylinders, the cheaper the cost is.

CPP is a variant of the two dimensional open dimension problem. It consists of packing n non-identical circles C_i of known radius r_i, $i \in N = \{1, \ldots, n\}$, into the smallest containing circle C. The objective is to determine the radius r of C as well as coordinates (x_i, y_i) of the center of C_i, $i \in N$.

Formally, CPP can be stated as:

$$
(\text{CPP}) \begin{cases} \text{Min.} & r & & & \\ \text{S.T.} & (x_i - x)^2 + (y_i - y)^2 & \leq & (r - r_i)^2 & (1) \\ & i \in N & & & \\ & (x_i - x_j)^2 + (y_i - y_j)^2 & \geq & (r_i + r_j)^2 & (2) \\ & (i, j) \in N \times N,\ j < i & & & \\ & r & \geq & LB_r & (3) \end{cases}
$$

where $LB_r = \sqrt{\sum_{i \in N} r_i^2}$. Equation (1) states that any C_i, $i \in N$, is totally contained within C. There are n of these constraints, one for each circle C_i. Equation (2) reinforces the no overlap constraint of any pair of distinct circles (C_i, C_j); that is, the distance between the centers of C_i and C_j 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. Equation (3) provides a positive

lower bound for the radius of the containing circle. It substitutes the non-negativity constraint whose elimination from the model makes CPP unbounded. CPP has a linear objective function but non linear constraints. Even though its set of feasible solutions has piecewise smooth surface frontiers, CPP is a difficult problem to solve.

This paper solves CPP using a width first beam search algorithm with decisions at each level of the tree being based on the so-called maximum hole degree (MHD). Beam search is a heuristic approach for solving complex combinatorial optimization problems. It is a classical tree search method that was introduced in the context of scheduling (Ow and Morton, 1988), but has been successfully applied to many other combinatorial optimization problems. Even though beam search has been used to solve a wide variety of optimization problems, its performance is not known for cutting and packing problems. To our knowledge, the only applications of beam search to cutting and packing problems are those proposed in (Hifi and M'Hallah, 2007a) for constrained strip packing problems and in (Bennell and Song, 2006) for nesting problems (for which no experimental results were provided).

The paper is organized as follows. Section 2 provides a detailed review of CPP related literature. Section 3 explains the notion of maximum hole degree. Section 4 elucidates the mechanisms of Beam Search. Section 5 describes the proposed adaptive beam search approach. Section 6 presents the results and assesses their performance in terms of solution quality and run time. Finally, Section 7 summarizes the contributions of this paper and gives possible extensions.

2. LITERATURE REVIEW

CPP has received very little attention. Most published papers focus on packing identical circles (Graham and Lubachevsky, 1996), (Lubachevsky and Graham, 1997). The most recent paper (Mladenovic *et al.*, 2005) proposed a general reformulation descent heuristic, which switches from solving the problem in Cartesian coordinates to solving it in polar coordinates and *vice versa*.

Very few papers addressing the problem of packing non identical circles are available. Most of these papers solve variants of the packing problem, in particular, packing circles into a rectangle. (George *et al.*, 1995) proposed a non-linear mixed integer program for packing circles of different diameters into a rectangle of fixed known dimensions. However, the model was not of practical use. They, therefore, solved the problem using constructive heuristics. (Stoyan and Yaskov, 1998) addressed the problem of packing non-identical circles into a fixed width strip (i.e., an Open Dimension Problem (Wascher *et al.*, 2007)), using a mathematical model that searches for feasible local optima using branch and bound and the reduced gradient method. (Correia *et al.*, 2001) proposed two new upper bounds for the problem. (Huang *et al.*, 2005) proposed two greedy algorithms which use a maximum hole degree rule and a look ahead search strategy. (Hifi *et al.*, 2004) adapted simulated annealing to pack non-identical circles into a strip of fixed dimensions (i.e., a single knapsack problem (Wascher *et al.*, 2007)). (Hifi and M'Hallah, 2004) designed a constructive heuristic and a genetic algorithm to pack circles into a rectangle of fixed width but of variable length (i.e., a single large object placement problem (Wascher *et al.*, 2007)). Both algorithms searched for a good order of the pieces and used an adaptation of the best local position procedure (Hifi and M'Hallah, 2002), (Hifi and M'Hallah, 2003) to identify the layout that minimized the length of the rectangle. (Birgin *et al.*, 2005) addressed the same problem using a non-linear approach.

Packing non-identical circles into a containing circle of known radius has been addressed in (Wang *et al.*, 2002), (Zhang and Deng, 2005), (Sugihara *et al.*, 2004), (Huang *et al.*, 2006), (Hifi and M'Hallah, 2007a) and (Hifi and M'Hallah, 2007b). (Wang *et al.*, 2002) randomly generated an initial pattern, measure its infeasibility, and translated circles whose positions are infeasible till restoring feasibility. (Zhang and Deng, 2005) combined (Wang *et al.*, 2002)'s approach with simulated annealing to explore the neighbourhood of the current solution, and with tabu search to implement jumps. (Sugihara *et al.*, 2004) constructed a sufficiently large circle that contained all circles, then reduced the radius of the containing circle using a shrink and shake iterative strategy.

(Huang *et al.*, 2006) solved CPP using A1.0 and A1.5, which are extensions of (Huang *et al.*, 2005). A1.0 and A1.5 use the notion of hole degree of a position of a circle to be packed given a fixed value of r and the positions of already packed circles. A1.0 chooses for the circle to be packed the feasible position having the highest hole degree. It is run $n(n\text{-}1)/2$ times; each time starting with a different pair of the n circles. A1.5 is a modified version of A1.0 applying a look ahead search for every feasible corner position of the circle to be packed. Given the set of i packed circles, A1.5 packs C_{i+1} in every feasible corner position and uses A1.0 to pack all remaining items. If it successfully packs all items, A1.5 stops; else, it chooses for C_{i+1} the feasible corner position yielding the infeasible solution having the highest density and pursues packing the remaining circles.

(Hifi and M'Hallah, 2007a) designed a dynamic adaptive iterative local search procedure. At each iteration, the algorithm identifies the set of potential best local positions of a circle C_i, $i \in N$, given the positions of the previously packed circles, and determines for each of these positions the coordinates and radius of the smallest containing circle. The circles are considered in a on-increasing order of their radii.

Finally, (Hifi and M'Hallah, 2007b) proposed a three-phase approximate algorithm. During its first phase, the algorithm successively packs the ordered set of circles. It

searches for each circle's "best" position given the positions of the already packed circles where the best position minimizes the radius of the current containing circle. During its second phase, the algorithm reduces the radius of the containing circle by applying (i) an intensified search -based on a reduction search interval- and (ii) a diversified search -based on the application of a number of layout techniques-. During its third and last phase, the algorithm introduces a restarting procedure that explores the neighbourhood of the current solution in search for a better ordering of the circles.

3. THE MHD HEURISTIC

MHD is a greedy heuristic (Huang *et al.*, 2006) which uses the notion of maximum hole degree. Having positioned a set $I^i \subseteq N$ of circles, MHD positions C_{i+1}, $i \in N \setminus I^i$, into the maximum hole degree corner position among all its feasible positions in C -i.e., without overlapping any of the already packed circles-. A corner position $p^{i+1} \in P_{I^i}$, the set of corner positions of C_{i+1} given I^i, is a feasible position of C_{i+1} in C such that C_{i+1} is either tangent to two circles of I^i or tangent to a circle of I^i and to the perimeter of C. The hole degree of C_{i+1}, when positioned in $p^{i+1} \in P_{I^i}$ is

$$\lambda_{p^{i+1}} = 1 - \frac{\underline{d}_{p^{i+1}}}{r_{i+1}},$$

where

$$\underline{d}_{p^{i+1}} = \min_{j \in I^i \setminus T_{p^{i+1}}} \{d_{i+1,j}\}$$

is the minimum distance between C_{i+1} and already packed circles in I^i excepting the two circles of $T_{p^{i+1}}$. The set $T_{p^{i+1}}$, $p^{i+1} \in P_{I^i}$ denotes the set of circles in I^i used to determine p^{i+1} for C_{i+1}. In addition, $d_{i+1,j}$ is the distance that separates C_{i+1} and C_j, $(i+1, j) \in N \times N$:

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

Figure 1 illustrates $P_{I^4} = \{p_1^5, p_2^5, p_3^5\}$, the set of all feasible corner positions of C_5 given r and the positions of the packed circles of $I^4 = \{C_1, \ldots, C_4\}$. The feasible corner position p_1^5 is tangent to C_2 and C, p_2^5 to C_1 and C_2, and p_3^5 to C_3 and C. It follows that $T_{p_1^5} = \{C_2\}$, $T_{p_3^5} = \{C_3\}$ and $T_{p_2^5} = \{C_1, C_2\}$. The minimum distance between C_5 and already packed circles when positioned in p_1^5 and p_3^5 are $d_{1,5}$ and $d_{4,5}$. The minimum distance between C_5 and already packed circles when positioned in p_2^5 equals 0 since C_5 happens to be also tangent to C_3.

Specifically, for a fixed radius r and a pre-determined ordering of the circles, MHD proceeds as follows. It starts by positioning the first two circles tangent to each other and to the containing circle. The first circle C_1 is set at $(0, r_1\text{-}r)$ whereas the second circle C_2 is positioned tangent to C_1 and to C. The remaining n-2 circles are then successively positioned using the MHD rule.

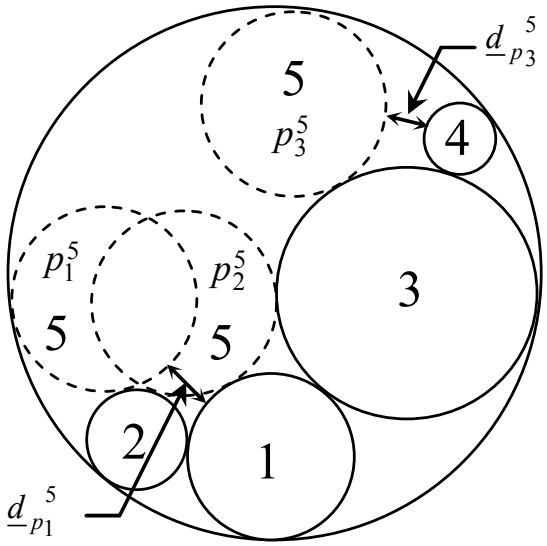

Figure 1. Corner positions

4. BEAM SEARCH

Beam-Search (BS) avoids exhaustive enumeration by performing a partial search of the solution space. It is a truncated branch and bound where at each level of the search tree only a subset of nodes, called the *set of elite nodes*, are selected for further branching. The other nodes are permanently discarded, and no backtracking is performed. The number ω of elite 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 of each level of the developed tree.

Figure 2 provides a pseudo code of a standard beam search. As with Branch and Bound, a lower bound can be used to fathom nodes. Indeed, if an initial feasible solution is available, then it is set as the incumbent solution and its value is assigned to z^*. On the other hand, if no initial feasible solution is available, z^* is set to 0 (assuming that the problem at hand is a maximization problem with a non-negative objective function).

1. *Initialization*

- Let ω be the beam width.
- Set $B = \{B_0\}$ and $B_\omega = \varnothing$, where B is the set of nodes to be investigated, and B_ω the set of nodes branched out of the nodes in B.
- If an initial feasible solution is available, set z^* to its objective function value; otherwise, set z^*=0.

2. *Iterative step*

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

3. *Stopping condition*

If $B = \varnothing$, stop; otherwise, goto the *iterative step*.

Figure 2. A generic beam search algorithm

Beam Search is characterized by the beam width ω, which is a constant used for filtering the set of offspring nodes B_ω. A node corresponds to a potential (partial) feasible solution. The set B of current nodes is initialized to the root node B_0 whereas the set B_ω of offspring nodes is initialized to the empty set. A current node of B generates a set of offspring nodes, and adds them to B_ω. 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^*. 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 assessed using an evaluation operator, and ranked in a non-ascending order of their values. The first ω nodes of B_ω are then chosen as the elite nodes; that is, they 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. This process is reiterated until no further branching is possible; that is, till B=∅.

5. AN ADAPTIVE BS FOR CPP

This paper solves CPP using an Adaptive Beam Search (ABS) algorithm. ABS uses a dichotomous search for the radius of the containing circle within an interval $[\underline{r}, \bar{r}]$, whose bounds $\underline{r}$ and $\bar{r}$ are dynamically updated. The lower bound $\underline{r}$ is updated every time the packing procedure yields an infeasible solution whereas the upper bound $\bar{r}$ is set to the current radius and updated every time a feasible layout is obtained. The dichotomous search is stopped when the width δ of the search interval becomes too small; i.e., when $\delta = \bar{r} - \underline{r} \leq 10^{-4}$. Initially, the lower bound $\underline{r}$ is set to $\sqrt{\sum_{i=1}^{n} (r_i)^2}$, whereas the upper bound $\bar{r}$ is set to $2 * \underline{r}$.

For $r \in [\underline{r}, \bar{r}]$, the packing procedure searches for a feasible layout of the ordered n circles using a width first beam search. Each node of the tree represents a partial solution or packing. The node at level 1, labelled η^1, is characterized by

- $I^1 = \{C_1\}$, where C_1 has been packed tangent to C at position $(0, r_1\text{-}r)$,
- $\bar{I}^1 = N \setminus \{C_1, C_2\}$ the set of n-2 circles that have not yet been considered for packing, and
- P_I^1, the set of feasible corner positions for C_2 given the position of C_1.

In general, a node at level l is characterized by the set I^l of already positioned circles, the set P_I^l of feasible corner positions of the circle C_{l+1} to be packed, and the set $\bar{I}^l$ of circles that remain to be packed; i.e., $\bar{I}^l = N \setminus I^l \setminus \{C_{l+1}\}$. Branching out of a node at level l is equivalent to choosing a position of P_I^l for C_{l+1}. There are as many possible branches out of a node as there are elements in P_I^l. Out of all the nodes at level l, the ω positions having the largest hole degrees are chosen as branches to be further investigated whereas all other nodes of level l are fathomed. In addition, a node whose $P_I^l = \varnothing$ is fathomed since it yields no feasible packing of the n circles into C. Finally, a node whose $\bar{I}^l = \varnothing$ is a leaf; that is, the n circles have been packed into C, and a feasible solution is at hand. Such a node has level $l = n$.

The adaptive beam search algorithm is detailed in Figures 3 and 4. Figure 3 provides the pseudo code of the dichotomous search undertaken by ABS whereas Figure 4 explains the mechanisms of the beam search undertaken by ABS given the position of the first two circles.

Input. An instance of CPP.
Output. An approximative solution C with radius $\underline{r}$.

Initialization Step.

Set $\underline{r} = \sqrt{\sum_{i=1}^{n} (r_i)^2}$ and $\bar{r} = 2 * \underline{r}$ and let $\delta = \bar{r} - \underline{r}$ be search interval width or gap between $\underline{r}$ and $\bar{r}$.

Iterative Step.

while $(\bar{r} - \underline{r}) > \delta$ **do**
Set $r = (\bar{r} + \underline{r}) / 2$
Generate the initial node η^1 characterized by $I^1 = \{C_1\}$, $\bar{I}^1 = N \setminus \{C_1, C_2\}$ and P_I^1.
Call *Beamsearch*(η^1)
If all circles have been packed **then** set $\bar{r} = r$
Else set $\underline{r} = r$
endif;
enddo

Figure 3. The Dichotomous Search of ABS

Procedure BeamSearch(η^1)

Initialization Step.

- Let B (resp. B_ω) be the set of current (resp. offspring) nodes.
- Set B to $\{\eta^1\}$, whose characteristics are $I^1 = \{C_1\}$, $\bar{I}^1 = N \setminus \{C_1, C_2\}$ and P_I^1. Set the level of the tree to $l = 1$.

Iterative Step.

while $B_\omega \neq \varnothing$ **do**
1. Set $l = l + 1$.
2. Set B equal to the $\min\{|B_\omega|, \omega\}$ nodes of B_ω having the best hole degrees, and reset $B_\omega = \varnothing$.
3. **For each** selected node of level l **do**
   a. Pack C_{l+1} into its selected position, and remove the node from B.
   b. **If** $l+1 = n$ **then** a feasible solution is at hand; stop the algorithm.
   c. Determine P_I^{l+1}, the set of potential corner positions for C_{l+2} given the positions of the already packed circles of I^{l+1}.
   d. **If** $P_I^{l+1} = \varnothing$ **then** the node leads to an infeasible solution; fathom the node.
   e. Create $\left|P_I^{l+1}\right|$ branches out of the node; each corresponding to a node characterized by the potential position of C_{l+2} given the positions of the already packed circles of I^{l+1}. Update B_ω by appending the created offspring nodes.

enddo
enddo

Figure 4. The Beam Search Iterative Steps of ABS

6. COMPUTATIONAL RESULTS

The objective of the computational investigation is to assess the performance of the beam search algorithm by comparing its results to the best known solutions in the literature, and to study the effect of the beam width ω on the solution quality and run time. ABS is run with the circles considered in a non-increasing order of their radii. ABS is run for all values of $\omega = 1, \ldots, 100$, for $n < 100$, and for all values of $\omega = 1, \ldots, 50$, for $n \geq 100$. ABS is coded in `C` and run on an `AMD Athlon XP2000+`, 1.67 Ghz and 512 Mb of RAM.

The algorithm is first tested on a set of 12 instances consisting of the six problems of Stoyan and Yaskov (Stoyan and Yaskov, 1998), and six additional problems obtained by duplicating or concatenating the six original problems. These 12 instances were used as benchmark problems in (Hifi and M'Hallah, 2007a) and (Hifi and M'Hallah, 2007b). These instances are interesting as they have not been specifically constructed for packing circles into a circle but for packing circles into a rectangle. Moreover, these problems include relatively large problems; thus, reflect the behaviour of the proposed algorithms when the problem size increases.

The best recorded results as well as radii obtained by A1.0, A1.5 and the adaptive approach HM of (Hifi and M'Hallah, 2007a) are given in Table 1. Columns 1 and 2 report the instance and the number n of circles. Columns 3-5 indicate $r_{\text{A1.0}}$, $r_{\text{A1.5}}$ and r_{HM}, the smallest radius obtained by applying A1.0, A1.5, and the adaptive approach HM. Columns 6-8 tally results obtained by ABS. Column 6 indicates r_{ABS}, the smallest radius obtained when ABS is applied. Column 7 shows ω_{ABS}, the beam width that yielded the smallest radius. Finally, Column 8 gives t_{ABS}, the run time in seconds when ABS is applied with beam width ω_{ABS}.

ABS improves all of the results obtained by the adaptive approach HM, but at the cost of a higher computational time. ABS improves some of the results obtained by A1.0 (exactly 3 out of 6 instances) but can not compete with A1.5 for small problems (SY1-SY4). For larger problems (n=100), ABS improves one result obtained by A1.5. It is noteworthy that A1.0 and A1.5 can not solve the larger sized problems with $n \geq 110$.

The algorithms are further tested on a second set of 24 instances consisting of the problems proposed in (Huang *et al.*, 2006). Table 2 reports the best results obtained by A1.0, A1.5 and ABS. Columns 1 and 2 indicates the instance and the problem size n, whereas Columns 3-4 indicate the smallest radius obtained by applying A1.0 and A1.5. Columns 5-7 tally results relative to the application of ABS: Column 5 indicates the smallest radius obtained when ABS is applied; Column 6 shows the beam width ω that yielded the smallest radius, and Column 7 gives the run time in seconds when ABS is applied with the beam width ω of Column 6.

Instance	n	$r_{A1.0}$	$r_{A1.5}$	r_{HM}	r_{ABS}	ω_{ABS}	t_{ABS}
SY1	30	7.30	7.21	7.2100	7.2732	13	3
SY2	20	6.34	6.27	6.3393	6.3125	45	3
SY3	25	6.45	6.42	6.4842	6.4629	63	9
SY4	35	9.16	9.05	9.2745	9.1633	12	4
SY5	100	13.20	13.17	13.2742	13.1523	87	900
SY6	100	14.95	14.94	15.1476	14.9832	45	427
SY1-4	110			14.8351	14.4780	39	498
SY5-6	200			20.2022	19.7817	95	8856
SY10	300			22.9070	22.6133	23	7600
SY20	200			19.8020	19.5531	22	1853
SY30	250			20.3060	20.0645	18	2907
SY40	350			28.8889	28.4238	26	15900

Table 1. Performance of the Beam Search Heuristics on the instances of Stoyan and Yaskov

The analysis of the results of Table 2 shows that ABS improves the results obtained by A1.0 in 16 out of 24 instances. However, it does not improve any of the results obtained by A1.5. Subsequently, to improve the performance of ABS, a diversification strategy is implemented. The beam search is restarted $n(n\text{-}1)$ times; once with every possible ordered pair of the n circles. The Beam Search with diversification, labeled ABSD and described in Figure 5, starts with the pair of circles (C_1, C_2) and $r = (\underline{r} + \bar{r}) / 2$. If it finds a feasible solution, the upper bound of r is updated. Otherwise, another pair of circles is considered. Every time a feasible solution is obtained, the upper bound is updated. The lower bound is updated only if the search has been applied to all $n(n\text{-}1)$ possible pairs without identifying a feasible solution.

The results of Table 3 show that ABS improves all the results obtained by A1.0 and improves in 14 out of 24 instances the results obtained by A1.5.

Instance	n	$r_{A1.0}$	$r_{A1.5}$	r_{ABS}	ω_{ABS}	t_{ABS}
NR10-1	10	102.29	99.89	100.2096	16	0.19
NR11-1	11	61.58	60.71	60.7100	29	0.39
NR12-1	12	66.08	65.30	66.1770	11	0.19
NR14-1	14	114.83	113.84	114.9881	20	0.58
NR15-1	15	38.99	38.97	39.1332	20	0.77
NR15-2	15	39.15	38.85	39.1465	83	3.86
NR16-1	16	144.51	143.44	144.5201	8	0.29
NR16-2	16	129.26	128.29	129.3405	94	6.51
NR17-1	17	49.60	49.25	49.3799	28	2.31
NR18-1	18	198.88	197.40	198.8163	27	1.64
NR20-1	20	126.69	125.53	126.4728	41	3.66
NR20-2	20	123.55	122.21	122.5748	9	0.58
NR21-1	21	150.70	148.82	149.9975	69	7.81
NR23-1	23	176.65	175.47	176.4390	59	8.68
NR24-1	24	140.17	138.38	139.5833	86	14.46
NR25-1	25	192.68	190.47	192.0539	21	3.47
NR26-1	26	249.93	246.75	249.3432	26	5.59
NR26-2	26	306.25	303.38	307.1960	30	6.56
NR27-1	27	225.46	222.58	224.5264	37	8.10
NR30-1	30	178.90	178.66	179.6901	15	3.57
NR30-2	30	174.93	173.70	175.0935	99	37.00
NR40-1	40	360.75	357.00	359.5155	31	23.14
NR50-1	50	381.05	380.00	380.0330	35	49.00
NR60-1	60	525.57	522.93	524.5508	57	151.00

Table 2. Adaptive Beam Search Results

Initalization Step.

Set $\underline{r} = \sqrt{\sum_{i=1}^{n} (r_i)^2}$ and $\bar{r} = 2 * \underline{r}$.

Set $(k_1, k_2) = (1,2)$.

Iterative Step.

while $(\bar{r} - \underline{r}) > \delta$ **do**

1. Set $r = (\bar{r} + \underline{r}) / 2$.
2. Choose the pair of circles (C_{k_1}, C_{k_2}) and generate the initial node η^2 characterized by $I^2 = \{C_{k_1}, C_{k_2}\}$.
3. Call *BeamSearch*(η^2).
4. **If** all n circles have been packed **then** $\bar{r} = r$, and repeat iterative step
   Else
   if ($k_2 < n$) then increment k_2 by 1 and goto Step 1 of Iterative Step.
   else if $k_1 < n$-1 increment k_1 by 1, reset k_2 to 1 and goto Step 1 of Iterative Step;
   else set $\underline{r} = r$, and repeat iterative step.
   endif

enddo

Figure 5. Adaptive Beam Search with Diversification

Figure 6, which displays the run times displayed in Table 2, shows that ABS's runtime increases linearly as the problem size increases but grows exponentially when the problem size becomes very large. These observations are further confirmed by Figure 7 which displays ABS's runtime for the instances of Table 1. Finally, the runtime of ABSD is a linear function of $n(n\text{-}1)$.

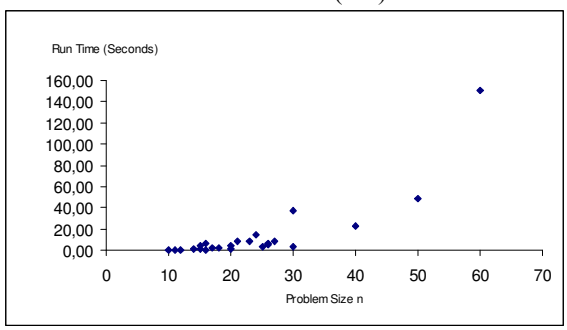

Figure 6. Variation of ABS's run time as problem size increases

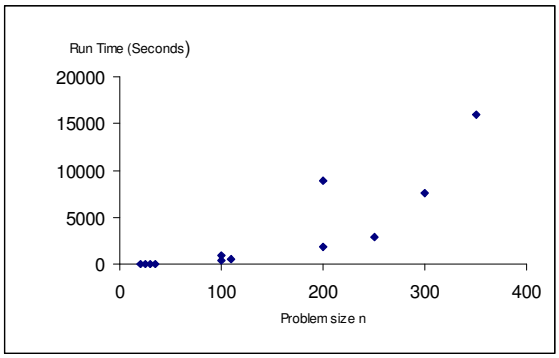

Figure 7. Variation of ABS's Run Time as the Problem Size Increases for Large Instances

Instance	$r_{A1.0}$	$r_{A1.5}$	r_{ABS}	r_{ABSD}	ω_{ABSD}	t_{ABSD}
NR10-1	102.29	99.89	100.2096	99.8851	39	19.13
NR11-1	61.58	60.71	60.7100	60.7100	38	28.25
NR12-1	66.08	65.30	66.1770	65.4752	3	4.70
NR14-1	114.83	113.84	114.9881	114.2919	36	151.00
NR15-1	38.99	38.97	39.1332	38.9441	16	59.46
NR15-2	39.15	38.85	39.1465	38.8380	91	1179.00
NR16-1	144.51	143.44	144.5201	143.7176	26	139.00
NR16-2	129.26	128.29	129.3405	128.5826	6	43.50
NR17-1	49.60	49.25	49.3799	49.2069	91	234.00
NR18-1	198.88	197.40	198.8163	198.2850	3	29.00
NR20-1	126.69	125.53	126.4728	125.6316	30	764.00
NR20-2	123.55	122.21	122.5748	122.2192	21	351.00
NR21-1	150.70	148.82	149.9975	149.1351	23	638.00
NR23-1	176.65	175.47	176.4390	175.4058	67	3072.00
NR24-1	140.17	138.38	139.5833	138.2778	17	510.00
NR25-1	192.68	190.47	192.0539	190.1855	36	1493.00
NR26-1	249.93	246.75	249.3432	247.5464	8	583.00
NR26-2	306.25	303.38	307.1960	303.2102	74	11240.00
NR27-1	225.46	222.58	224.5264	222.4896	53	3750.00
NR30-1	178.90	178.66	179.6901	178.0102	43	5045.00
NR30-2	174.93	173.70	175.0935	173.4359	47	9217.00
NR40-1	360.75	357.00	359.5155	357.0695	21	22140.00
NR50-1	381.05	380.00	380.0330	378.5854	19	21400.00
NR60-1	525.57	522.93	524.5508	521.2739	9	13975.00

Table 3. Adaptive Beam Search with Diversification

7. CONCLUSION

This paper solves the circular packing problem using an adaptive algorithm. The algorithm combines dichotomous search with beam search and uses the notion of maximum hole degree to position a circle at each node of the beam search tree. The search space of the algorithm is further diversified by restarting it with all possible ordered pairs of circles. The algorithm improves existing solutions in many instances. It is very fast for small to medium sized problems. Its run time remains reasonable for large sized problems. The performance of the algorithm can be further improved by hybridizing it with different diversification and intensification search strategies.

REFERENCES

Birgin E. G., 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*, 160, p.19-33.

Bennell J. A., X. Song, 2006. A beam search implementation for nesting problems, *3rd ESICUP meeting*, Portugal.

Correia M. H., J. F. Oliveira and J. S. Ferreira, 2001. A new upper bound for the cylinder packing problem, *International Transactions in Operational Research*, p. 571-583.

George J. A., J. M. George and B. W. Lamar, 1995. Packing different--sized circles into a rectangular container, *European Journal of Operational Research*, 84, p. 693-712.

Graham R. L. and B. D. Lubachevsky, 1996. Repeated patterns of dense packings of equal disks in a square, *The Electronic Journal of Combinatorics*, 3, 16 pages.

Hifi M. and R. M'Hallah, 2002. A best-local position procedure-based heuristic for the two-dimensional layout problem, *Studia Informatica Universalis, International Journal on Informatics, Special Issue on Cutting, Packing and Knapsacking*, 2, p. 33-56.

Hifi M. and R. M'Hallah, 2003. A hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes, *International Transactions in Operational Research*, 3, p. 195-216.

Hifi M. and R. M'Hallah, 2004. Approximate algorithms for constrained circular cutting problems, *Computers & Operations Research*, 31, p. 675-694.

Hifi M. and R. M'Hallah, 2007a. A dynamic adaptive local search based algorithm for the circular packing problem, *European Journal of Operational Research*, 183(3), p. 1280-1294.

Hifi M. and R. M'Hallah, 2007b. Adaptive and restarting techniques-based algorithms for circular packing problems, Computational Optimization and Applications, DOI 10.1007/s10589-007-9049-5.

Hifi M., V. Th. Paschos and V. Zissimopoulos, 2004. A simulated annealing approach for the circular cutting problem, *European Journal of Operational Research*, 159, p. 430-448.

Hifi M., R. M'Hallah and T. Saadi, 2006. Beam Search Algorithms for Constrained Two-staged Two-Dimensional Cutting Problems, *INFORMS Journal on Computing*, to appear.

Huang W. Q., Y. Li, H. Akeb and C. M. Li, 2005. Greedy algorithms for packing unequal circles into a rectangular container, *Journal of the Operational Research Society*, 56, p. 539-548.

Huang W. Q., Y. Li, C. M. Li and R. C. Xu, 2006. New heuristics for packing unequal circles into a circular container, *Computer & Operations Research*, 33, p. 2125-2142.

Lubachevsky D. and R. L. Graham, 1997. Curved hexagonal packing of equal circles in a circle, *Discrete and Computational Geometry*, 18, p. 179-194.

Mladenovic N., F. Plastria and D. Urosevic, 2005. Reformulation descent applied to circle packing problems, *Computers & Operations Research*, 32, p. 2419-2434.

Ow P. S. and T. E. Morton, 1988. Filtered beam search in scheduling, *International Journal of Production Research*, 26, p. 35-62.

Stoyan Y. G. and G. N. Yaskov, 1998. Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints, *International Transactions in Operational Research*, 5, p. 45-57.

Sugihara K., M. Sawai, H. Sano, D. S. Kim and D. Kim, 2004. Disk packing for the estimation of the size of wire bundle, *Japan Journal on Industrial and Applied Mathematics*, 21, p. 259-278.

Wang H., W. Huang, Q. Zhang and D. Xua, 2002. An improved algorithm for the packing of unequal circles within a larger containing circle, *European Journal of Operational Research*, 141, p. 440-453.

Wascher G, H. Haussner and H. Schumann, 2007. An improved typology of cutting and packing problems, *European Journal of Operational Research*, 183(3), p. 1109-1130.

Zhang D. and A. Deng, 2005. An effective hybrid algorithm for the problem of packing circles into a larger containing circle, *Computers & Operations Research*, 32, p. 1941-1951.