Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 33 (2006) 2125–2142
computers & operations research
www.elsevier.com/locate/cor

New heuristics for packing unequal circles into a circular container☆

Wen Qi Huang[a], Yu Li[b,*], Chu Min Li[b], Ru Chu Xu[a]

[a]*College of Computer Science, HuaZhong University of Science and Technology, Wuhan 430074, China*
[b]*LaRIA, Université de Picardie Jules Verne, 33 Rue Saint Leu, 80039 Amiens cedex 1, France*

Available online 12 February 2005

Abstract

We propose two new heuristics to pack unequal circles into a two-dimensional circular container. The first one, denoted by A1.0, is a basic heuristic which selects the next circle to place according to the *maximal hole degree rule*. The second one, denoted by A1.5, uses a *self look-ahead strategy* to improve A1.0. We evaluate A1.0 and A1.5 on a series of instances up to 100 circles from the literature and compare them with existing approaches. We also study the behaviour of our approach for packing equal circles comparing with a specified approach in the literature. Experimental results show that our approach has a good performance in terms of solution quality and computational time for packing unequal circles.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Circle packing problem; Combinatorial optimization; Heuristic

1. Introduction

The two-dimensional (2D) circle packing problem is a famous packing problem [1], encountered in some industries (textile, glass, wood, paper, etc.). It consists in placing a set of circles in a container without overlap. The usual objective is to maximize the material utilization and hence to minimize the wasted area. There are various options in the problem, e.g. the container can be circle, rectangle or polygon, the

☆ This work is supported by "The National Natural Science Foundation of China under grant No. 10471051" and "Programme de Recherches Avancées de Coopérations Franco-Chinoises (PRA SI02-04)".
* Corresponding author.
E-mail addresses: wqhuang@mail.hust.edu.cn (W.Q. Huang), yu.li@u-picardie.fr (Y. Li), cli@laria.u-picardie.fr (C.M. Li).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.01.003

radii of circles to be placed can be equal or unequal. In this paper, we investigate the problem of packing unequal circles into a circular container. The problem is known to be NP-hard [2].

Most published research on circle packing focused on packing equal circles into a container [3–6]. The proposed approaches were heavily influenced by the congruence of the circles.

Several authors addressed the problem of packing unequal circles into a rectangular or a strip container. George et al. [7] proposed a set of heuristic rules and different combinations of these rules to pack unequal circles into a rectangle and the best ones were a quasi-random algorithm and a genetic algorithm. Stoyan and Yaskov [8,9] proposed a mathematical model and a solution method based on a combination of the branch-and-bound algorithm and the reduced gradient method to pack unequal circles into a strip. Hifi and M'Hallah [10] presented a Bottom-Left (BL) based genetic algorithm to solve a dual of circle packing problem: cutting unequal circles on a rectangular plate.

But to our knowledge, no significant published research addresses the problem of packing unequal circles into a circular container, except a simulation approach based on an elasticity physics model, proposed by Huang et al. [11,12] and Wang et al. [13]. This approach can obtain good enough solutions for packing unequal circles and very good solutions for packing equal circles, but the computational time greatly increases for large instances.

In this paper, we propose two new heuristics to solve the problem. A preliminary version of this work appeared in [14,15], and then extended to solve the problem of packing unequal circles into a rectangular container [16]. The basic idea of our approach is a quantified measure, called *hole degree*, to evaluate the benefit of placing a circle in the container, which gives the first algorithm, denoted by A1.0. Another feature of our approach is the use of a *self look-ahead strategy* to improve A1.0 and gives the second algorithm, denoted by A1.5. We evaluate our approach on a series of instances up to 100 circles from the literature [8,9,12–15,17] and compare it with existing approaches. We also study the behaviour of our approach when it is applied to pack equal circles. Experimental results show that our approach has a good performance in terms of solution quality and computational time for packing unequal circles.

The paper is organized as follows. In Section 2, we give a formal definition of the circle packing problem. In Section 3, we present the two heuristics A1.0 and A1.5. In Section 4, we present and analyze the experimental results. Section 5 concludes the paper.

2. Circle packing problem

We consider the following circle packing optimization problem: given a set of unequal circles, find the minimal radius of a circular container so that all the circles can be packed into the container without overlap. The associated decision problem is formally stated as follows.

Given a circular container of radius r_0 and a set of n circles $c_1, \ldots, c_n$ of radii $r_1, \ldots, r_n$. There may be several circles with the same radius. Let (0,0) be the coordinate of the container centre, and (x_i, y_i) the coordinate of the centre of the circle c_i, the problem is to determine if there exist $2n$ real numbers $x_1, y_1, \ldots, x_n, y_n$ such that

$$r_0 - r_i - \sqrt{x_i^2 + y_i^2} \geqslant 0, \quad i \in \{1, \ldots, n\}, \tag{1}$$

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - r_i - r_j \geqslant 0, \quad i \neq j \in \{1, \ldots, n\}. \tag{2}$$

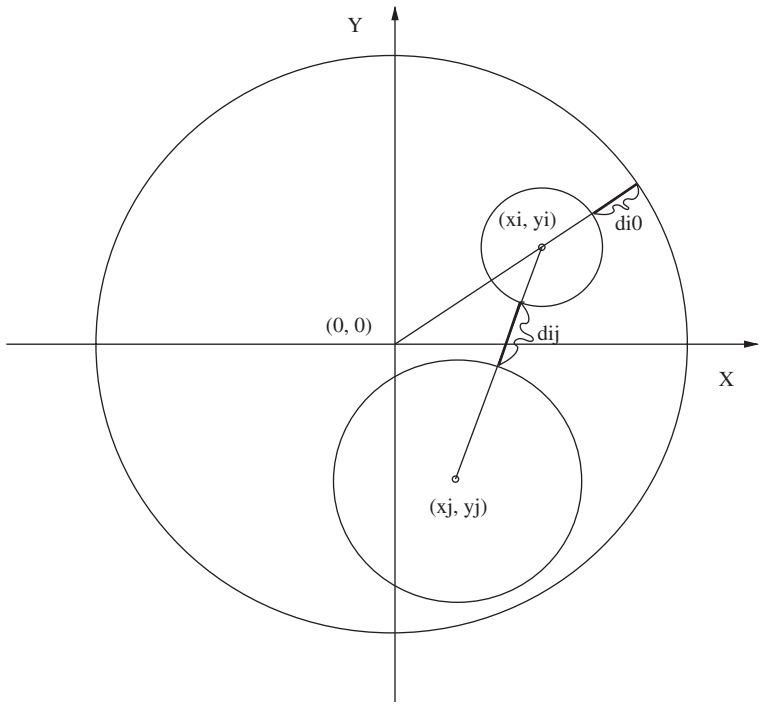

Fig. 1. Circle packing problem.

Constraint (1) states that circle c_i placed in the container should not extend outside the container, i.e., the distance from c_i to the container, denoted by d_{i0}, should not be negative:

$$d_{i0} = r_0 - r_i - \sqrt{x_i^2 + y_i^2}.$$

Constraint (2) requires that circles placed in the container cannot overlap each other. For such two circles c_i and c_j, the distance between them, denoted by d_{ij}, should not be negative:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - r_i - r_j.$$

Fig. 1 illustrates these constraints.

If we find an efficient algorithm to solve this decision problem, we can solve the original optimization problem by using some search strategies, for example dichotomous search [18], to reach the minimal radius of the container within a fixed precision.

Let ε be a precision, I be a decision packing problem instance with the container radius r_0, A be an algorithm solving I. The procedure in Fig. 2 computes the minimal $r_{\min}$ of the container using A, so that the n circles in I can be placed into the container without overlap.

Consequently, we will concentrate our discussion on the decision problem and propose two heuristics to solve it in the following section.

```
Procedure DichotomousSearch(I,ε,A)
Begin
    r0 = ∑ri;
    while (A(I,r0/2) return "success") do
        r0 = r0/2;
    r0' = r0/2;
    repeat
        if (A(I,(r0+r0')/2) return "success")
        then r0 = (r0 + r0')/2;
        else r0' = (r0 + r0')/2;
    until r0−r0' < ε;
    rmin = r0;
End.
```

Fig. 2. Dichotomous search.

3. Algorithms A1.0 and A1.5

Inspired from human experiences in packing, we propose a quantified measure, called *hole degree*, to evaluate the benefit of placing a circle in the container. The selection of the next circle to pack is guided by the *maximal hole degree* (*MHD*) rule, which gives our basic packing algorithm A1.0.

Following the same line, we propose a *self look-ahead strategy* to improve A1.0, which consists in using A1.0 itself to evaluate the benefit of placing a circle in the container, which gives our second packing algorithm A1.5.

First, we explain the basic notions, *corner placement* and *hole degree*. Then, we present the MHD rule and A1.0, and the self look-ahead strategy and A1.5.

3.1. Corner placement and hole degree

We give some preliminaries before presenting the corner placement and the hole degree.

Definition (*Configuration*). A configuration C is a partial pattern (layout) where $m \geqslant 2$ circles have been already placed inside the container without overlap, and $n - m$ circles remain to be packed into the container.

A configuration is said to be *successful* if $m = n$, i.e., all circles have been placed inside the container without overlap.

A configuration is said to be *failure* if $m < n$ and none of the circles outside the container can be placed into the container to satisfy constraints (1) and (2).

Let M be the set of circles already placed in the container. Assume in addition that the container is itself in M.

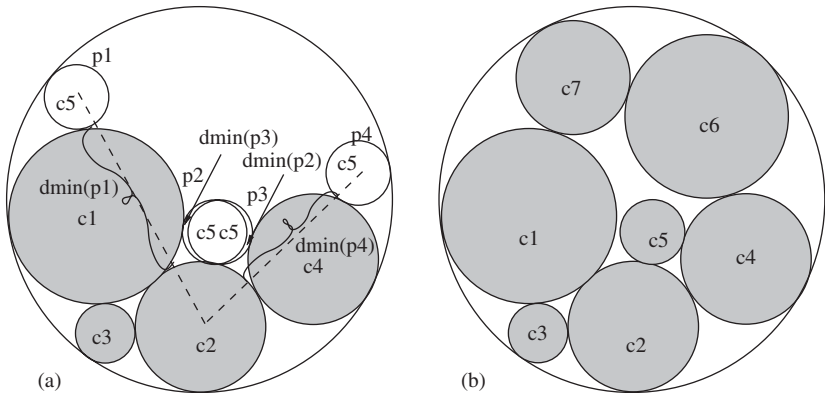

Fig. 3. An instance of 7 circles: (a) corner placements to put circle c_5; (b) a successful configuration obtained by A1.0.

3.1.1. Corner placement

Human experiences in packing show that the placements of a circle at some particular positions, like corner regions formed by the container and circles already placed, are worth to be examined to generate a successful configuration [16]. The notion *corner placement* in our approach is inspired from these experiences, which refers to placing a circle at a corner region formed by the container and circles already placed, so that the circle touches at least two circles.

Definition (*Corner placement*). Given a configuration C, a corner placement is the placement of a circle c_i inside the container, so that c_i does not overlap any other circle in M and is tangent at least with two circles in M (note that the two circles are not necessarily tangent each other, and one of them may be the container itself). A corner placement is represented by a triplet $p(c_i, x, y)$, meaning that c_i is to be placed at position (x, y).

Fig. 3 illustrates an instance of 7 circles, $c_1, c_2, c_3, c_4, c_5, c_6$ and c_7, to be placed into a circular container with radius r_0, where c_2 and c_4 are of the same size, i.e., $r_2 = r_4$.

Note that,

1. Let $p(c_i, x, y)$ be a corner placement, the two circles tangent with c_i if c_i is placed at (x, y) are uniquely determined. Reciprocally, if c_i is placed in the container and tangent with two circles c_u and c_v, then the centre of c_i is also uniquely determined. So a corner placement might also be denoted by $p(c_i, c_u, c_v)$.
2. There may be several corner placements to put a circle in the container. Fig. 3(a) illustrates four possible corner placements to put circle c_5: p_1, p_2, p_3 and p_4, where circles c_1, c_2, c_3 and c_4 are already placed in the container (in this order), p_1 places c_5 at the position tangent with c_1 and the container, p_2 places c_5 at the position tangent with c_1 and c_2, p_3 places c_5 at the position tangent with c_2 and c_4, p_4 places c_5 at the position tangent with c_4 and the container.
3. For circles of the same radius, the corresponding corner placements are identical. For example, the placements to put circles c_2 and c_4 are the same.

3.1.2. Hole degree

Human experiences also tell us that a good placement may be a region of a good "hole" form so that the wasted area after placing the circle is as small as possible. To evaluate the benefit of a corner placement, we propose a quantified measure λ, called *hole degree*, to describe how the circle to be placed at a corner placement is close to the nearest circle already inside the container.

Definition (*Hole degree*). Let $p(c_i, x, y)$ be a corner placement, c_u and c_v be the two circles in M and tangent with circle c_i if c_i is placed at (x, y). The degree λ of the corner placement $p(c_i, x, y)$ is defined as

$$\lambda = \left(1 - \frac{d_{\min}}{r_i}\right),$$

where r_i is the radius of c_i, and $d_{\min}$ is the minimal distance from c_i to other circles in M (excluding c_u and c_v), i.e., the distance between c_i *and the nearest circle to it*, $d_{\min} = \min_{j \in M, j \neq u,v}(d_{ij})$.

Note that, if c_i can be placed by a corner placement into the container and tangent with more than two circles, then $d_{\min} = 0$ and $\lambda = 1$; otherwise $\lambda < 1$. If λ is equal to 1, it means that the placed circle occupies a complete "hole", and the wasted area after placing this circle is the minimal. For example, the degree λ of the corner placement putting c_3 at the position tangent with c_1, c_2 and the container in Fig. 3(a) is equal to 1, since the corresponding $d_{\min} = 0$. Fig. 3(a) also indicates the minimal distance $d_{\min}$ of the four corner placements to put circle c_5.

3.2. Maximal hole degree (MHD) rule and Algorithm A1.0

We use the hole degree λ as the benefit of a corner placement. Intuitively, since one should place a circle as close as possible to the circles already inside the container, a packing procedure should select a corner placement having the maximal hole degree to put the next circle into the container.

Definition (*MHD rule*). By the MHD rule, the corner placement $p(c_i, x, y)$ having the maximal hole degree λ is selected to place c_i at position (x, y).

The MHD rule directly results in our basic packing Algorithm A1.0.

Given a circle packing instance I, A1.0 proceeds as follows. It starts by placing a pair of unequal circles into the empty container, so that the two circles are tangent each other and also to the container, generating an initial configuration. Under the current configuration, for each circle outside the container, A1.0 examines every corner placement by calculating its degree. Among all possible placements, A1.0 selects the placement $p(c_i, x, y)$ with the maximal degree λ, and places c_i at (x, y).

If all circles are placed in the container so that constraints (1) and (2) are satisfied, A1.0 stops with success. If a failure configuration is reached, A1.0 tries the next pair of circles. If all pairs of circles have been tried but no successful configuration is found, A1.0 stops with failure.

Fig. 3(b) illustrates a successful configuration obtained by A1.0 to pack seven circles into the container, where circles c_1 and c_2 are first placed in the container to generate an initial configuration, then c_3, c_4, c_5, c_6 and c_7 are successively placed according to the MHD rule. Fig. 4(a) shows a successful configuration obtained by A1.0 for packing 26 circles into a container (instance NR26-1 in Table 1).

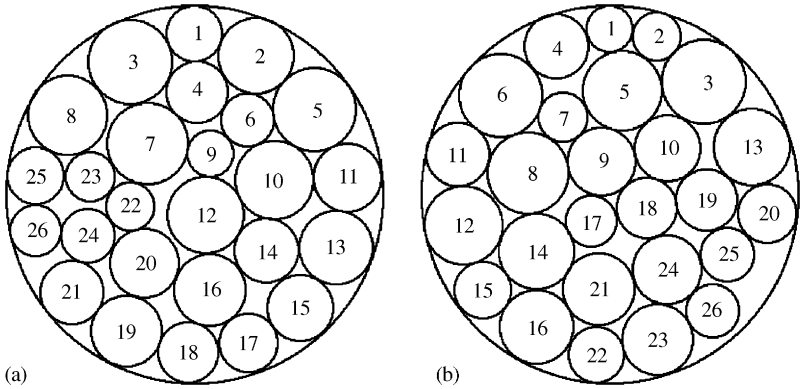

Fig. 4. Two successful configurations for instance NR26-1 obtained by A1.0 and A1.5, where the number of each circle is the order to place. The same configurations with the order of circles in Table 1 are shown in Fig. 8(a) and (b). (a) Solution by A1.0, $r_0 = 249.93$; (b) Solution by A1.5, $r_0 = 249.93$.

Algorithm A1.0 is described in Fig. 5.

Given a configuration C, we always associate to C the list L of all possible corner placements. Note that L is empty for a successful configuration or a failure configuration. After placing circle c_i, the list L of corner placements is modified as follows:

- Remove all invalid corner placements. A corner placement becomes invalid because the involved circle would overlap c_i if it was placed.
- Re-calculate the degree λ of the remaining corner placements.
- If a circle outside the container can be placed inside the container without overlap so that it is tangent with c_i and another circle inside the container (including the container), create a new corner placement and put it into L, and compute its degree.

Algorithm A1.0 was tested on the instances presented in Section 4. Experimental results show that A1.0 is effective enough to solve some instances.

3.3. Self look-ahead strategy and Algorithm A1.5

Given a configuration, A1.0 only looks at the relation between the circles already inside the container and the circle to be packed. It does not examine the relation between the circles outside the container.

In order to more globally evaluate the benefit of a corner placement and to improve A1.0, we propose a *self look-ahead strategy* which consists in using the result obtained by A1.0 (A1.0Core more precisely) to measure the benefit of a corner placement, which gives our second packing algorithm A1.5 which is presented below.

Given a configuration C, A1.5 examines every corner placement using procedure A1.0Core on a copy of C. It places the circle of the examined placement in the container and then applies A1.0Core to reach a final configuration. If the final configuration is successful, A1.5 stops; otherwise, the density of the final configuration is used to define the benefit of the examined placement, where the configuration density is the ratio of the total surfaces of the circles inside the container to the surface of the container. After

Table 1
24 irregular instances of the first group

Inst	*n*	r_i	r_0^*
NR10-1*	10	10, 12, 15, 20, 21, 30, 30, 30, 40, 50	99.89
NR11-1*	11	8.4, 11, 10, 10.5, 12, 14, 15, 20, 20, 25, 25	60.71
NR12-1	12	11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22	65.30
NR14-1*	14	11, 14, 15, 16, 17, 19, 23, 27, 31, 35, 36, 37, 38, 40	113.84
NR15-1	15	3, 3, 4, 4, 4.5, 6, 7.5, 8, 9, 10, 11, 12, 13, 14, 15	38.97
NR15-2*	15	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15	38.85
NR16-1	16	13, 14, 15, 15, 17, 19, 23, 26, 27, 27, 32, 37, 38, 47, 57, 63	143.44
NR16-2	16	21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36	128.29
NR17-1*	18	5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 10, 10, 10, 15, 15, 20, 25	49.25
NR18-1	18	12, 14, 16, 23, 25, 26, 27, 28, 33, 35, 47, 49, 53, 53, 55, 60, 67, 71	197.40
NR20-1	20	4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42	125.53
NR20-2	20	6, 8, 9, 12, 12, 15, 16, 18, 20, 21, 24, 24, 27, 28, 30, 32, 33, 36, 40, 44	122.21
NR21-1	21	10, 15, 16, 17, 17, 18, 21, 22, 23, 25, 26, 31, 33, 34, 37, 37, 38, 39, 40, 42, 45	148.82
NR23-1	23	14, 14, 16, 18, 18, 21, 22, 23, 26, 28, 28, 32, 34, 34, 36, 37, 39, 41, 45, 48, 49, 49, 51	175.47
NR24-1	24	9, 10, 11, 13, 13, 16, 17, 17, 18, 19, 19, 20, 20, 20, 21, 22, 23, 23, 24, 25, 30, 31, 32, 82	138.38
NR25-1	25	14, 17, 22, 26, 26, 27, 28, 29, 29, 30, 31, 32, 33, 34, 34, 34, 34, 35, 37, 37, 37, 47, 52, 53, 55	190.47
NR26-1	26	31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	246.75
NR26-2	26	41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66	303.38
NR27-1	27	17, 21, 25, 26, 26, 27, 27, 28, 29, 33, 33, 34, 35, 35, 35, 37, 40, 42, 43, 44, 45, 49, 53, 55, 55, 55, 63	222.58
NR30-1	30	5, 8, 10, 10, 12, 14, 15, 16, 18, 20, 20, 20, 20, 20, 22, 24, 25, 26, 30, 30, 30, 30, 35, 40, 40, 45, 48, 50, 55, 60	178.66 .
NR30-2	30	6, 8, 8, 10, 12, 13, 14, 16, 18, 18, 20, 22, 23, 24, 25, 27, 28, 29, 31, 33, 33, 35, 37, 38, 39, 41, 43, 43, 48, 53	173.70 .
NR40-1	40	31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70	357.00
NR50-1	50	18, 18, 19, 19, 19, 19, 21, 21, 24, 25, 25, 30, 31, 31, 33, 33, 36, 36, 40, 42, 43, 46, 46, 47, 49, 49, 49, 50, 50, 54, 56, 56, 57, 57, 58, 58, 59, 59, 59, 61, 62, 63, 63, 64, 65, 68, 78, 79, 80, 86	380.00
NR60-1	60	35, 35, 35, 36, 37, 37, 38, 38, 39, 39, 40, 41, 41, 42, 42, 42, 42, 42, 44, 44, 45, 45, 46, 46, 47, 48, 48, 49, 50, 50, 54, 54, 57, 57, 59, 60, 60, 71, 71, 71, 72, 72, 74, 74, 76, 77, 77, 79, 79, 80, 82, 82, 85, 86, 89, 90, 94, 95, 96, 100	522.93

examining all placements, A1.5 takes the corner placement with the maximal benefit to really change the configuration C. Like A1.0, A1.5 tries all pairs of unequal circles before stopping with failure. A1.5 is described in Fig. 6.

Obviously, A1.5 looks further forward and evaluates the benefit of a corner placement by examining the relation between all circles (inside and outside the container).

Fig. 4(b) shows a successful configuration obtained by A1.5 to solve NR26-1 in Table 1.

3.4. Complexity of A1.0 and A1.5

The analysis of the real computational time of A1.0 and A1.5 is difficult, because it not only depends on the number of circles and the container radius, but also the diversity of circle radii, as well as the implementation. Here, we analyze the upper bound of the complexity of A1.0 and A1.5 in the worse case,

```
Procedure A1.0(I)
Begin
   for k := 1 to n − 1 do
      for l := k + 1 to n do
         generate an initial configuration C using circles k and l;
         generate the corner placement list L;
         if (A1.0Core(C, L) returns a successful configuration)
         then stop with success;
   stop with failure;
End.
Procedure A1.0Core(C, L)
Begin
   while (there are corner placements in L) do
      for every corner placement p(c_i, x, y) do
         benefit(p(c_i, x, y)) := λ ;
      select the corner placement p(c_i, x, y) with the maximum benefit;
      modify C by placing c_i at (x, y);
      modify L;
   return C;
End.
```

Fig. 5. Algorithm A1.0.

```
Procedure A1.5(I)
Begin
   for k := 1 to n − 1 do
      for l := k + 1 to n do
         generate an initial configuration C using circles k and l;
         generate the corner placement list L;
         if (A1.5Core(C, L) returns a successful configuration)
         then stop with success;
   stop with failure;
End.
Procedure A1.5Core(C, L)
Begin
   while (there are corner placements in L) do
      for (every corner placement p(c_i, x, y) do
         let C′ and L′ be copies of C and L;
         modify C′ by placing circle c_i at (x, y); modify L′;
         C′ =A1.0Core(C′, L′);
         if (C′ is successful) then return C′;
         else benefit(p(c_i, x, y)) := density(C′);
      select the corner placement p(c_i, x, y) with the maximum benefit;
      modify C by placing circle c_i at (x, y);
      modify L;
   return C;
End.
```

Fig. 6. Algorithm A1.5.

i.e., when they do not find a successful configuration, and discuss the real computational time to find a successful configuration.

3.4.1. Upper bound of A1.0's complexity

Given a configuration *C* where *m* circles are already placed in the container and it remains $n-m$ circles outside the container. Let $|L|$ be the number of corner placements in the list *L* of corner placements. After

placing a new circle c_i into the container, the modification of L is done by removing all invalid corner placements, re-calculating the degree λ for remaining ones and computing new corner placements. To remove invalid placements, we must check each placement in L to verify if it overlaps the new placed circle c_i, which is done in $O(|L|)$. To re-calculate the degree λ for remaining placements, we must compute the distance between c_i and the circles involved in the remaining placements, which is done in $O(|L|)$. New corner placements for $n-m$ circles outside the container are generated by using c_i and m circles in the current configuration, which is done in $O(m(n-m))$. So the complexity for placing a circle is $O(|L|+m(n-m))$. Then the complexity for placing n circles, i.e., the complexity of A1.0Core, is $O(n(|L|+m(n-m)))=O(n|L|+n^3)$. A1.0 calls A1.0Core for every pair of circles, its complexity is bounded by $O(n^3|L|+n^5)$.

3.4.2. Upper bound of A1.5's complexity

A1.5Core uses a powerful self look-ahead strategy in which the consequences of each possible placement in L is evaluated by applying A1.0Core in full, which allows to examine the relation between all circles (inside and outside the container). So, the complexity of A1.5Core for placing a circle is $O(|L|(n|L|+n^3))=O(n|L|^2+n^3|L|)$. The complexity of A1.5Core for placing n circles is bounded by $O(n(n|L|^2+n^3|L|))=O(n^2|L|^2+n^4|L|)$. A1.5 calls A1.5Core for every pair of circles, its complexity is bounded by $O(n^4|L|^2+n^6|L|)$.

In theory, since every pair (there are $m(m-1)/2$ pairs) of circles (including the container) can give two possible corner placements for $n-m$ circles outside the container, $|L|$ is bounded by $O(n^3)$. But in practice, most corner placements are infeasible and the real number of corner placements in L is much smaller than the theoretical upper bound $O(n^3)$. In Tables 4 and 5, the, the rows max($|L|$) record the maximal number of placements to be examined for placing a circle by A1.0 and A1.5 for an instance. These numbers vary from 32 to 360 for A1.0, and from 52 to 571 for A1.5, for solving 24 instances of up to 60 circles in Table 4; and from 70 to 750 for A1.0, and from 135 to 792 for A1.5, for solving 6 instances of up to 100 circles in Table 5.

The real computational time of A1.0 and A1.5 to find a successful configuration is still much smaller than the above upper bounds. When a successful configuration is found, A1.0 does not continue to try other pairs of circles, nor A1.5 to exhaust the search space. In fact, every call to A1.0Core in A1.5Core may lead to a successful configuration and stop the execution. Then, the real computational cost of A1.0 and A1.5 essentially depends on the real number of corner placements in L and the distribution of successful configurations.

4. Experimental results and analysis

In this section, we evaluate our approach on a series of instances up to 100 circles from the literature [8,9,12–15,17], and compare our approach with the simulation approach proposed by Huang et al. and Wang et al. [11–13] in terms of solution quality and running time. We also study the behaviours of our approach when it is applied to pack equal circles.

The simulation approach in [11–13] is practically a Local Search based on an elasticity physics model, denoted by LS. We briefly present it below: the n circles are considered as elastic disks and squeezed inside the container. Due to elastic forces between the disks, the n disks will move to restore their original shapes. The objective function to minimize is thus defined as the sum of the potential energies of the

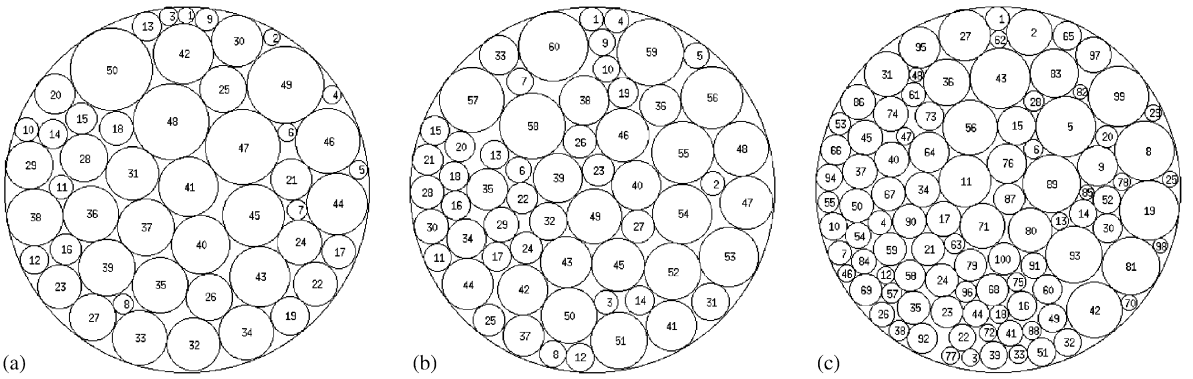

Fig. 7. Three successful configurations obtained by A1.5, where the number of circles for NR50-1 and NR60-1 is the order in Table 1, and for SY5 the order in Stoyan and Yaskov's files [19]: (a) Solution for NR50-1, $r_0 = 380.00$; (b) Solution for NR60-1, $r_0 = 522.93$; (c) Solution for SY5, $r_0 = 13.17$.

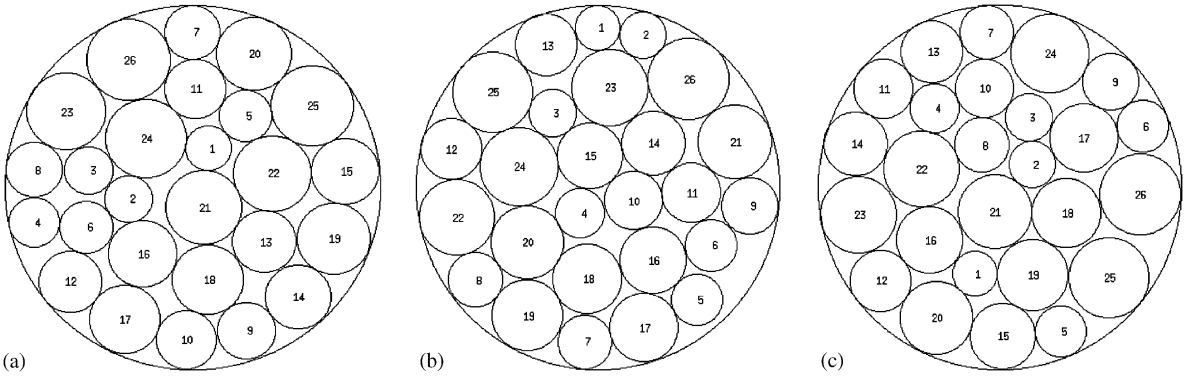

Fig. 8. Three successful configurations for NR26-1 obtained by A1.0, A1.5 and LS with $r_0 = 249.93$, where the number of circles is the order in Table 1: (a) Solution by A1.0 for NR26-1; (b) Solution by A1.5 for NR26-1; (c) Solution by LS for NR26-1.

disks in the system. Such movements continue until the potential energy of the system reaches a local minimum. Then a strategy inspired from human behaviour is used to make the system escape from local minima. A more detailed description of the approach can be found in [12,13,17]. Though LS can obtain enough good solutions for packing unequal circles and very good solutions for packing equal circles, the computational time greatly increases for large instances, as shown by the following experimental results.

A1.0 , A1.5 and LS were all implemented in C language, and executed on a PC with an Athlon XP2000+ processor and 256 Mo of RAM.

4.1. Test instances

Unequal circle packing instances can be classified into two categories: *irregular* instances and *regular* instances. For an irregular instance, the successful configurations have an irregular form (see Figs. 7 and 8) and the optimal container radius is unknown; while for a regular instance, the successful configurations are of regular form and the optimal container radius can be obtained by geometrical calculation.

Regular instances are in general easier to solve than irregular ones [14]. We concentrate thus the following discussion on irregular instances.

We evaluate our approach on two groups of irregular instances which are intended to represent a large spectrum of the difference between the radii of circles to be placed. The first group includes 24 irregular instances from 10 to 60 circles listed in Table 1. The radii of the circles to be packed are of discrete values and some of them are identical. The column *Inst* gives the identifier of an instance, the columns n and r_i respectively provide the number and the radii of the circles to be placed, and the column r_0^* gives the minimal container radii obtained by A1.5. The 5 instances marked by "*" were used to test LS: NR10-1*, NR11-1*, NR14-1* and NR17-1* used in [12]; NR15-2* in [13]; NR15-2* and NR17-1* also in [17]. The 19 other instances are taken from a preliminary version of the present work in [14,15].

We also use the 6 irregular instances of 20 to 100 circles proposed by Stoyan and Yaskov [8,9], in such a way that we place the circles into a circular container instead of a rectangular one. The 6 instances, denoted by SY1, ..., SY6, constitute the second group. The radii of the circles to be packed are random real numbers and all different, available from [19].

In addition, we made these instances and their solutions publicly available from [19], including the successful configurations and the corresponding circle centre coordinates, hoping to help further study on this problem.

4.2. Results and analysis

We apply A1.0 and A1.5 to solve the above instances and analyze the obtained results in terms of solution quality and running time, i.e., the obtained minimal container radius and the running time to get a successful configuration with this minimal container radius. We also evaluate LS in the same way for the comparison purpose. The results obtained by A1.0, A1.5 and LS for the first group of the instances are reported in Tables 2 and 4, and for the second group in Tables 3 and 5.

Tables 2 and 3 provide the solution quality results of A1.0, A1.5 and LS. The columns r_0 give the obtained minimal container radii. The columns *density* (%) are the corresponding successful configuration densities. We summarize, in the columns *gain* (%), the density gains made by A1.5 over A1.0, A1.5 over LS, and A1.0 over LS respectively, which are calculated as $100 \times (density_1 - density_2)/density_1$, where $density_1$ is the density obtained by the first algorithm and $density_2$ is that of the second algorithm. Just as a reference to the solution quality for the 6 instances in the second group, we report also the best known density obtained by Stoyan and Yaskov's approach to pack circles into a rectangular container [8,9], in the column *density* (%) leaded by "S.Y." of Table 3.

Tables 4 and 5 report the running time results of A1.0, A1.5 and LS. For each algorithm, r_0 indicates the obtained minimal container radius; *time* (s) is the running time (in second) to find a successful configuration under r_0. For r_0 of A1.0 and LS, A1.5 is also applied to find a successful configuration and the corresponding running time is given in the column leaded by "A1.5" for comparison. The last two columns $\max(|L|)$ give the maximal number of corner placements to be examined for placing a circle by A1.0 and A1.5 for an instance (see the Section 3.4).

4.2.1. Performance of A1.0 and A1.5

Concerning the solution quality, we observed from Tables 2 and 3 that:

- A1.0 produces good enough solutions within reasonable time with the average density of 81.977% for the first group and of 83.910% for the second one.

Table 2
Solution quality results obtained by A1.0, A1.5 and LS for the first group of instances

Inst	r_0	A1.0 *density* (%)	r_0	A1.5 *density* (%)	r_0	LS *density* (%)	A1.5/A1.0 *gain* (%)	A1.5/LS *gain* (%)	A1.0/LS *gain* (%)
NR10-1*	102.29	77.509	99.89	81.279	99.92	81.230	4.64	0.06	−4.80
NR11-1*	61.58	79.555	60.71	81.852	60.79	81.636	2.81	0.26	−2.62
NR12-1	66.08	78.093	65.30	79.970	65.60	79.240	2.35	0.91	−1.47
NR14-1*	114.83	80.396	113.84	81.801	114.39	81.016	1.72	0.96	−0.77
NR15-1	38.99	83.047	38.97	83.132	39.14	82.412	0.10	0.87	0.76
NR15-2*	39.15	80.902	38.85	82.156	39.06	81.275	1.53	1.07	−0.46
NR16-1	144.51	83.282	143.44	84.530	144.89	82.846	1.48	1.99	0.52
NR16-2	129.26	79.817	128.29	81.029	129.35	79.706	1.50	1.63	0.14
NR17-1*	49.60	82.312	49.25	83.486	49.30	83.317	1.41	0.20	−1.22
NR18-1	198.88	82.512	197.40	83.753	199.67	81.860	1.48	2.26	0.79
NR20-1	126.69	82.490	125.53	84.022	126.62	82.582	1.82	1.71	−0.11
NR20-2	123.55	82.708	122.21	84.531	123.67	82.547	2.16	2.35	0.19
NR21-1	150.70	81.619	148.82	83.694	150.99	81.305	2.48	2.85	0.38
NR23-1	176.65	83.092	175.47	84.213	177.85	81.974	1.33	2.66	1.35
NR24-1	140.17	84.046	138.38	86.234	140.02	84.226	2.54	2.33	−0.21
NR25-1	192.68	81.327	190.47	83.225	192.33	81.623	2.28	1.92	−0.36
NR26-1	249.93	81.103	246.75	83.207	249.93	81.103	2.53	2.53	0.00
NR26-2	306.25	80.906	303.38	82.444	306.00	81.038	1.87	1.71	−0.16
NR27-1	225.46	81.830	222.58	83.961	225.76	81.613	2.54	2.80	0.27
NR30-1	178.90	85.217	178.66	85.446	180.90	83.344	0.27	2.46	2.20
NR30-2	174.93	84.427	173.70	85.627	176.13	83.280	1.40	2.74	1.36
NR40-1	360.75	82.480	357.00	84.222	361.75	82.025	2.07	2.61	0.55
NR50-1	381.05	85.176	380.00	85.648	385.25	83.329	0.55	2.71	2.17
NR60-1	525.57	83.602	522.93	84.449	528.97	82.531	1.00	2.27	1.28
Average		81.977		83.496		81.961	1.83	1.83	0.02

For each algorithm, r_0 indicates the obtained minimal container radius, *density* (%) the density of the corresponding successful configuration, *gain* (%) the density gain obtained by one algorithm over another one.

- A1.5 improves the solution quality of A1.0 with the average density of 83.496% for the first group and of 85.117% for the second one, corresponding to 1.83% and 1.43% average density gains respectively. The improvement brought by A1.5 is significant, considering that the average density given by A1.0 is already important. Note that, for the second group, the best known density of packing circles into a rectangular container, obtained by Stoyan and Yaskov's approach [8,9], is 82.161% in the average.

About the computational cost, we remarked from Tables 4 and 5 that:

- The running time of A1.0 to get a successful configuration under its minimal container radius for an instance is reasonable, but that of A1.5 is considerable for large instances.

On the one hand, this fact can be explained by the computational complexity analysis in the Section 3.4, and on the other hand we can also give some practical analyses as follows:

- The real computational time of A1.0 and A1.5 essentially depends on the number $|L|$ of corner placements for placing a circle and the distribution of successful configurations. $|L|$ is related to the

Table 3
Solution quality results obtained by A1.0, A1.5 and LS for the second group of instances

Inst	*n*	r_0	A1.0 *density* (%)	r_0	A1.5 *density* (%)	r_0	LS *density* (%)	A1.5/A1.0 *gain* (%)	A1.5/LS *gain* (%)	A1.0/LS *gain* (%)	S.Y. *density* (%)
SY1	30	7.30	82.564	7.21	84.638	7.32	82.114	2.45	2.98	0.55	83.186
SY2	20	6.34	81.851	6.27	83.689	6.34	81.851	2.20	2.20	0.00	81.638
SY3	25	6.45	84.242	6.42	85.031	6.49	83.207	0.93	2.15	1.23	81.940
SY4	35	9.16	83.074	9.05	85.106	9.22	81.996	2.39	3.65	1.30	81.738
SY5	100	13.20	85.722	13.17	86.113	13.38	83.431	0.45	3.11	2.67	82.220
SY6	100	14.95	86.007	14.94	86.123	15.20	83.202	0.13	3.39	3.26	82.243
Average			83.910		85.117		82.634	1.43	2.91	1.50	82.161

For each algorithm, r_0 indicates the obtained minimal container radius, *density* (%) the density of the corresponding successful configuration, *gain* (%) is the density gain obtained by one algorithm over another one. Column S.Y. gives the density of a successful configuration obtained by Stoyan and Yaskov's approach [8,9] to pack these circles into a rectangular container.

size of the instance and the structure of the instance (radii of the circles). In general, the larger the instance, the more the number of corner placements (see the column $\max(|L|)$ in Tables 4 and 5). Concerning the distribution of successful configurations, when the container radius is not close to the optimal one, there exist many successful configurations, A1.0 and A1.5 can quickly find such one (see Tables 4 and 5). However, when the container radius is very close to the optimal one, few successful configurations exist in the search space, A1.0 and A1.5 may need more time to find a successful configuration in this case. The minimal container radii found by A1.5 are guessed to be very close to the optimal values, even though the optimal ones are not known.

- Our main objective in this paper is to study the behaviours of the proposed algorithms A1.0 and A1.5, so we did not make a big effort to optimize our implementation. The further improvement to the implementation might considerably reduce the real run time of A1.0 and A1.5.

Fig. 7 gives three successful configurations for NR50-1, NR60-1 and SY5 obtained by A1.5 with its minimal container radius r_0. Fig. 8(a) and (b) give two successful configurations for NR26-1 obtained by A1.0 and A1.5 with $r_0 = 249.93$. The corresponding circle centre coordinates for these configurations are available from [19].

4.2.2. *Comparison of A1.0 and A1.5 with LS*

We observe that:

- For the first group of instances, A1.0 yields better solutions than LS for larger instances, but not for some small instances. For the second group of instances, A1.0 works systematically better than LS in terms of solution quality and running time, with the average density gain of 1.50%.
- A1.5 works better than LS in terms of solution quality and running time, with the average density gain of 1.83% for the first group and 2.91% for the second one. A1.5 is much faster than LS to find a successful configuration for all instances under the minimal container radius obtained by LS. This difference is specially huge for large instances, for example for NR40-1, NR50-1, NR60-1, SY5 and

Table 4
Running time results of A1.0, A1.5 and LS for the first group of instances

Inst	r_0	A1.0 *time* (*s*)	A1.5 *time* (*s*)	r_0	A1.5 *time* (*s*)	r_0	LS *time* (*s*)	A1.5 *time* (*s*)	A1.0 max($\|L\|$)	A1.5 max($\|L\|$)
NR10-1*	102.29	<1	<1	99.89	<1	99.92	16	<1	24	34
NR11-1*	61.58	<1	<1	60.71	<1	60.79	48	<1	28	36
NR12-1	66.08	<1	<1	65.30	6	65.60	477	<1	32	52
NR14-1*	114.83	<1	3	113.84	2	114.39	1150	3	40	50
NR15-1	38.99	<1	4	38.97	25	39.14	769	<1	74	75
NR15-2*	39.15	<1	<1	38.85	6	39.06	527	3	73	75
NR16-1	144.51	<1	<1	143.44	71	144.89	1841	1	69	99
NR16-2	129.26	<1	10	128.29	44	129.35	1213	8	52	84
NR17-1*	49.60	<1	<1	49.25	30	49.30	103	1	126	180
NR18-1	198.88	<1	6	197.40	88	199.67	1080	3	78	112
NR20-1	126.69	1	7	125.53	39	126.62	1436	3	99	109
NR20-2	123.55	<1	<1	122.21	318	123.67	3029	2	80	118
NR21-1	150.70	<1	19	148.82	683	150.99	1023	<1	70	140
NR23-1	176.65	<1	5	175.47	1229	177.85	384	2	78	175
NR24-1	140.17	3	1	138.38	2339	140.02	7846	4	90	187
NR25-1	192.68	3	36	190.47	4614	192.33	10776	13	95	158
NR26-1	249.93	3	2	246.75	1019	249.93	14265	2	105	191
NR26-2	306.25	5	185	303.38	5164	306.00	9981	557	108	193
NR27-1	225.46	4	39	222.58	4436	225.76	9984	12	105	204
NR30-1	178.90	6	200	178.66	1365	180.90	2621	9	154	229
NR30-2	174.93	12	47	173.70	1078	176.13	3287	1	132	189
NR40-1	360.75	10	58	357.00	12109	361.75	41523	14	192	324
NR50-1	381.05	346	534	380.00	9717	385.25	11977	7	270	346
NR60-1	525.57	103	1219	522.93	13256	528.97	74574	272	360	571

For each algorithm, r_0 indicates the obtained minimal container radius; *time* (*s*) is the running time (in second) to find a successful configuration under r_0. The last two columns max($|L|$) give the maximum number of corner placements to be examined for placing a circle by A1.0 and A1.5 for an instance.

Table 5
Running time results of A1.0, A1.5 and LS for the second group of instances

Inst	*n*	r_0	A1.0 *time* (*s*)	A1.5 *time* (*s*)	r_0	A1.5 *time* (*s*)	r_0	LS *time* (*s*)	A1.5 *time* (*s*)	A1.0 max($\|L\|$)	A1.5 max($\|L\|$)
SY1	30	7.30	1	5	7.21	100	7.32	9733	2	126	183
SY2	20	6.34	<1	<1	6.27	604	6.34	4671	<1	70	135
SY3	25	6.45	3	19	6.42	4884	6.49	16119	<1	112	222
SY4	35	9.16	4	2	9.05	13924	9.22	4749	<1	156	323
SY5	100	13.20	364	2616	13.17	3630	13.38	32756	3	710	741
SY6	100	14.95	8445	6014	14.94	28752	15.20	41513	3	750	792

For each algorithm, r_0 indicates the obtained minimal container radius; *time*(*s*) is the running time (in second) to find a successful configuration under r_0. The last two columns max($|L|$) give the maximum number of corner placements to be examined for placing a circle by A1.0 and A1.5 for an instance.

Table 6
Results of A1.0, A1.5 and LS for 10 instances of equal circle packing problem compared to the best known results

n	Graham's *density* * (%)	LS *density* (%)	*Gap* (%)	A1.0 *density* (%)	*Gap* (%)	A1.5 *density* (%)	*Gap* (%)	A1.5 *time* (*s*)	LS *time* (*s*)
56	79.667	79.667	0.00	75.566	5.15	77.611	2.58	<1	<1
57	79.882	79.869	0.12	75.181	5.88	77.681	2.76	<1	<1
58	79.815	79.814	0.00	75.640	5.23	78.571	1.56	<1	3
59	79.912	79.912	0.00	75.608	5.39	78.905	1.26	<1	19
60	80.260	80.258	0.00	74.020	7.77	79.920	0.42	<1	4
61	81.314	81.314	0.00	75.010	7.75	78.695	3.22	<1	15
62	79.523	79.522	0.00	73.532	7.53	78.733	0.99	<1	7
63	79.672	79.671	0.00	75.236	5.57	78.788	1.11	<1	3
64	79.684	79.684	0.00	75.166	5.67	78.854	1.04	<1	86
65	79.938	79.938	0.00	74.227	7.14	78.246	2.12	<1	44
Average	79.967	79.965	0.00	74.919	6.31	78.600	1.71		

SY6, A1.5 is respectively about 3000, 1800, 270, 10000 and 14000 times faster than LS.

Fig. 8 gives three successful configurations for NR26-1 obtained by A1.0, A1.5 and LS with the same container radius $r_0 = 249.93$, and the corresponding circle centre coordinates for these configurations are available from [19].

The above experimental results show that A1.5 and A1.0 work well for packing unequal cirlces in terms of solution quality and running time. However, we notice the difference of the solution quality between the two groups of instances: the average solution quality obtained by A1.0 and A1.5 for the second group where circles are all different is higher than the first group where some circles are identical. LS remains relatively stable for the two groups. This phenomenon and the referees's suggestions lead us to study an extreme situation: equal circle packing.

4.3. Equal circle packing by A1.0 and A1.5

In order to further study the behaviours of A1.0 and A1.5, we apply A1.0 and A1.5 as well as LS to pack equal circles and compare the obtained results with the best known results for packing up to 65 circles, given by a simulation approach developed by Graham et al. [5].

Graham et al.'s simulation approach, referred as to "*Billiards Simulation* (*BS*)", consists in distributing randomly *n* points in the container and gradually blowing them as increasing circles in an uniform manner. While the circles are growing, change the positions of some circles if collisions occur between them. The process continues until the growing of circles is too small or the number of iterations is larger than a given number. Its running time greatly increases for larger instances. Like other equal circle packing algorithms, Graham et al.'s approach is also based on heavy exploitation of the congruence of circles and cannot be applied for unequal circle packing.

Table 6 gives the results obtained by A1.0, A1.5 and LS compared to Graham et al.'s ones. The column *n* is the number of equal circles, from 56 to 65. The column *density** (%) is the maximal density obtained by Graham et al.'s approach. The column *density* (%) gives the maximal density obtained by A1.0, A1.5

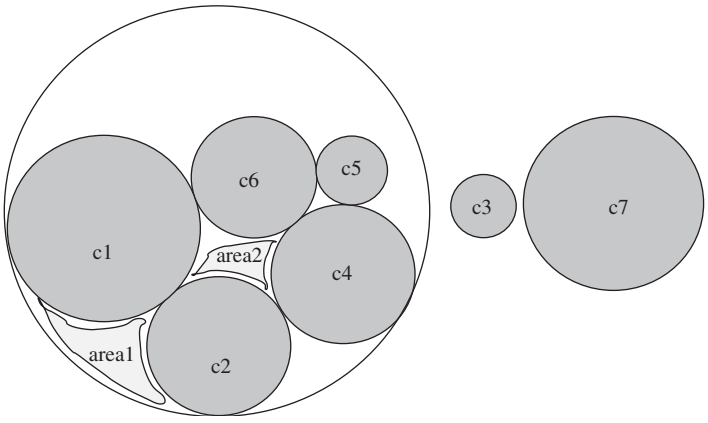

Fig. 9. Illustration of the MHD rule.

and LS. The column *gap* (%) displays the gap between the obtained density and the best known one, which is equal to $100 \times (density^* - density)/density^*$. The columns *time* ($s$) under A1.5 and LS are the running times of A1.5 and LS to get successful configurations under the minimal container radii obtained by A1.5. Graham et al.'s times to get their best results were not reported. From Table 6, we observe that A1.0 and A1.5 work worse than LS with the average gap of 6.31% and 1.71% respectively from the best known results, while LS obtains nearly the same results as the best known ones.

The main reason is that the degree λ used in A1.0 and A1.5 is a good measure to evaluate a corner placement when the circle radii are quite different, but not good enough in the case where most or all circles are identical. We give an intuitive explanation in Fig. 9, using the same example as in Fig. 3. When the circle radii are quite different, there may be some corner placements having degrees λ equal or close to 1 which completely fill "holes" formed by the already placed circles, like a placement of degree $\lambda = 1$ to put c_3 in *area*1 and a placement of degree λ close to 1 to put c_5 in *area*2 in Fig. 9. The MHD rule ensures to occupy such areas as early as possible so that the wasted spaces after placing the corresponding circles are as small as possible. If c_3 is placed elsewhere than *area*1, *area*1 will be wasted, because no other circle can fill it. Similarly, if c_5 is not placed in *area*2, but c_6 occupies *area2*, c_5 has to be placed somewhere else in the future as indicated in Fig. 9, then larger space in *area*2 will be wasted.

However, when most or all circles are identical, there are less corner placements having degree λ equal or close to 1. In this case, the earlier occupation of a corner placement having the maximal degree by the MHD rule is less interesting. We should search for other measures more suitable for this case. This issue will be studied in detail in a future paper.

On the other hand, the running time of A1.5 for packing equal circles can be greatly reduced, since there is only one circle to be examined at each iteration. Therefore, A1.5 can be used to obtain quickly a reasonably good solution even when n is very large.

5. Conclusion

We have presented two new heuristics A1.0 and A1.5 for packing unequal circles into a 2D circular container. The main features of our approach are the use of the MHD rule to select a next circle to place and obtain A1.0, and the use of a self look-ahead strategy to improve A1.0 and obtain A1.5.

We evaluate our approach on a series of instances from the literature and compare with existing approaches. Experimental results show that our approach works a good performance for packing unequal circles in term of solution quality and computational time.

In the future, we hope to study further the MHD rule in order to improve A1.0 and A1.5. We will also study other circle packing problems.

Acknowledgements

The authors thank anonymous referees for their helpful comments and suggestions which improved the quality of the paper.

References

[1] Lenstra JK, Rinnooy AHG. Complexity of packing, covering, and partitioning problems. In: Schrijver A, editor. Packing and covering in combinatorics. Amsterdam: Mathematisch Centrum; 1979. p. 275–97.
[2] Hochbaum DS, Maass W. Approximation schemes for covering and packing problems in image processing and VLSI. Journal of the Association for Computing Machinery 1985;32:130–6.
[3] Dowsland KA. Palletisation of cylinders in cases. OR Spektrum 1991;13:171–2.
[4] Fraser HJ, George JA. Integrated container loading software for pulp and paper industry. European Journal of Operational Research 1994;77/3:466–74.
[5] Graham RL, Lubachevsky BD, Nurmela KJ. Östergard, Dense packings of congruent circles in a circle. Discrete Mathematics 1998;181:139–54.
[6] Reis GE. Dense packing of equal circles within a circle. Mathematics Magazine 1975;48:33–7.
[7] George JA, George JM, Lamer BW. Packing different-sized circles into a rectangular container. European Journal of Operational Research 1995;84:693–712.
[8] Stoyan YuG, Yaskov GN. Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints. International Transactions on Operational Research 1998;5/1:45–57.
[9] Stoyan YuG, Yoskov G. A mathematical model and a solution method for the problem of placing various-sized circles into a strip. European Journal of Operational Research 2004;156/3:590–600.
[10] Hifi M, M'Hallah R. Approximate algorithms for constrained circular cutting problems. Computers and Operations Research 2004;31/5:675–94.
[11] Huang WQ, Zhan SH. A quasi-physical method of solving packing problems. Mathematical reviews. Providence, RI: American Mathematical Society 1982;82h:52002.
[12] Huang WQ, Li Y, Xu RC. Local search based on a physical model for solving a circle packing problem. Meta-Heuristic international conference 2001, Porto, Portugal; 2001.
[13] Wang HQ, Huang WQ, Zhang Q, Xu DM. An improved algorithm for the packing of unequal circles within a larger containing circle. European Journal of Operational Research 2002;141:440–53.
[14] Huang WQ, Li Y, Gérard S, Li CM, Xu RC, A "learning from human" heuristic for solving unequal circle packing problem. In: Hao JK, Liu BD, editors. Proceedings of the first international workshop on heuristics, Beijing, China; 2002. p. 39–45.
[15] Huang WQ, Li Y, Jurkowiak B, Li CM. A two-level search strategy for packing unequal circles into a circle container. In: Francesca R, editor. Proceedings of principles and practice of constraint programming—CP2003, Kinsale, Ireland, Lecture notes in computer science, vol. 2833. Berlin: Springer; 2003. p. 868–72.
[16] Huang WQ, Li Y, Akeb H, Li CM. Greedy algorithms for packing unequal circles into a rectangular container. Journal of Operational Research in Society, 2004; available online doi:10.1057/palgrave.jors.2601836.
[17] Zhang DF, Deng AS. An effective hybrid algorithm for the problem of packing circles into a larger containing circle. Computers and Operations Research; Available Online 7 February 2004.
[18] Cormen Thomas H, Leiserson Charles E, Rivest Ronald L. Introduction to algorithms. MA, London, UK: The MIT Press, Cambridge; 1989.
[19] http://www.laria.u-picardie.fr/∼yli