

Highlights

Adaptive Simulated Annealing with Greedy Search for the Circle Bin Packing Problem

Yong Yuan, Kevin Tole, Fei Ni, Kun He, Zhengda Xiong, Jinfu Liu

- First paper to introduce the circle bin packing problem with circular items (CBPP-CI).
- Define a tangent occupying action and propose a greedy constructive algorithm for CBPP-CI.
- Design two new operations, circle perturbation and sector perturbation, to generate neighbor solutions.
- Propose an adaptive simulated annealing algorithm with greedy search that obtains competitive results.
- Build two sets with a total of 52 new benchmark instances with 20 to 100 circular items.

Adaptive Simulated Annealing with Greedy Search for the Circle Bin Packing Problem

Yong Yuan^{a,1}, Kevin Tole^{a,b,1}, Fei Ni^a, Kun He^{a,*}, Zhengda Xiong^a and Jinfa Liu^c

^aSchool of Computer Science, Huazhong University of Science and Technology, Wuhan 430074, China.

^bInstitute of Computing and Informatics, Technical University of Mombasa, Mombasa 90420 - 80100, Kenya.

^cGuangzhou Key Laboratory of Multilingual Intelligent Processing, Guangdong University of Foreign Studies, Guangzhou 510006, China

ARTICLE INFO

Keywords:

Packing

Heuristics

Tangent occupying action

Adaptive simulated annealing

Greedy search

ABSTRACT

We introduce a new bin packing problem, termed the circle bin packing problem with circular items (CBPP-CI). The problem involves packing all the circular items into multiple identical circle bins as compact as possible with the objective of minimizing the number of used bins. We first define the tangent occupying action (TOA) and propose a constructive greedy algorithm that sequentially packs the items into places tangent to the packed items or the bin boundaries. Moreover, to avoid falling into a local minimum trap and efficiently judge whether an optimal solution has been established, we continue to present the adaptive simulated annealing with greedy search (ASA-GS) algorithm that explores and exploits the search space efficiently. Specifically, we offer two novel local perturbation strategies to jump out of the local optimum and incorporate the greedy search to achieve faster convergence. The parameters of ASA-GS are adaptive according to the number of items so that they can be size-agnostic across the problem scale. We design two sets of new benchmark instances, and the empirical results show that ASA-GS completely outperforms the constructive greedy algorithm. Moreover, the packing density of ASA-GS on the top few dense bins is much higher than that of the state-of-the-art algorithm for the single circle packing problem, inferring the high quality of the packing solutions for CBPP-CI.

1. Introduction

As a classic combinatorial optimization problem, the packing problems aim to pack a certain number of items into one or multiple containers without overlapping. Most researches are for single container packing. The shape of the container can be rectangular, square, or circular, and the items can be rectangles or circles. As an important branch of operational research, the packing problems have a wide variety of applications in the logistic industry, circular cutting, container loading, cylinder packing, etc. Meanwhile, it has been proved to be NP-hard by (Demaine, Fekete and Lang, 2010). Hence there is no deterministic algorithm to find the exact solutions in polynomial time unless $P = NP$.


The bin packing problem (BPP) has been well studied for multiple container packing since the 1970s (Johnson, 1973). There exist mainly two variants: the two-dimensional rectangular bin packing problem (2D-RBPP) and the two-dimensional square bin packing problem with circular items (SBPP-CI). The 2D-RBPP aims to pack a set of rectangular items into a minimum number of identical rectangular bins without overlapping (Chung, Garey and Johnson, 1982). The impact of these techniques on the practical solution of 2D-RBPP has been quite impressive (Christensen, Khan, Pokutta and Tetali, 2017). For example, Kang and Park (2003) propose two greedy algorithms: IFFD and IBFD. IFFD assigns the items sequentially by the first-fit decreasing manner, and a new bin will be initialized when there is

no more room for the packing; IBFD is a modification of IFFD, which assigns each item to the bin with the smallest remaining capacity. Other representative approaches include the tabu search (Lodi, Martello and Vigo, 1999), the guided local search (Faroe, Pisinger and Zachariasen, 2003), the hybrid GPASP/VND approach (Parreño, Alvarez-Valdés, Oliveira and Tamarit, 2010), and various heuristics based on greedy method (Lodi, Martello and Monaci, 2002; Monaci and Toth, 2006; Wei, Oon, Zhu and Lim, 2011). The SBPP-CI allocates all the circular items to a minimum number of square bins without overlap, which is first presented by He and Dosh (2017). They further propose a greedy algorithm with corner occupying action to improve the packing quality by introducing the adaptive large neighborhood search (He, Tole, Ni, Yuan and Liao, 2021).

To our knowledge, many studies have focused on multiple square or rectangular containers, while no significant published research addresses the problem of packing with multiple circular bins. Therefore, in this paper, we address a new variant termed the circle bin packing problem with circular items (CBPP-CI), which places a series of circular items inside multiple circular bins to minimize the number of bins used. It is an important extension of the two-dimensional circle packing problem (CPP), which is to pack all circular items into a single container of the circular or square shape to minimize the size of the container. Generally speaking, the approaches of CPP can be classified into two categories: constructive strategies and global optimization strategies.

Constructive strategies sequentially pack the items into the bin based on some rules, such as the best-local position (BLP) (Hifi and M'Hallah, 2002; Mhand and Rym,

*Corresponding author

 brook1et60@hust.edu.cn (K. He)

ORCID(s):

¹The first two authors contribute equally.

2004) and the maximal hole degree (MHD) (Huang, Li, Li and Xu, 2006), which are defined to evaluate the benefit of a partial solution. Representative heuristics include the prune-enriched Rosenbluth method (PERM) (Lü and Huang, 2008), the augment beam search (Akeb, Hifi and M'Hallah, 2009; Akeb, Hifi and Negre, 2011), the best-fit algorithm (BFA) (He, Huang and Jin, 2012), etc.

As the second category of approaches, global optimization strategies improve the solution iteratively based on the initial solution. It could be further subdivided into two categories: quasi-physical methods and meta-heuristic optimizations. The quasi-physical methods are based on a physical gradient or human-intuitive behavior to enhance the solutions obtained by problem-oriented heuristics (Wang, Huang, Zhang and Xu, 2002; Lubachevsky and Graham, 1997), while meta-heuristic optimizations usually have an evaluation function devised to employ a trade-off between randomization and local search, with the goal of directing and remodeling basic heuristics to generate feasible solutions. Typical algorithms include a simulated annealing approach (SA) (Hifi, Paschos and Zissimopoulos, 2004), monotonic basin hopping approach (MBH) (Grosso, Jamali, Locatelli and Schoen, 2010), iterated tabu search (ITS) (Fu, Huang and Lü, 2013), action-space-based global optimization algorithm (ASGO) (He, Huang and Yang, 2015), formulation space search (FSS) (López and Beasley, 2016), adaptive tabu search and variable neighborhood descent (ATS-VND) (Zhizhong, Xinguo, Kun and Zhanghua, 2018), etc.

Most of the constructive solutions focus on the traditional CPP and are designed on the specific characteristics of the problem. These methods are no longer applicable for CBPP-CI because of the characteristic gap between CPP and CBPP-CI. Moreover, although the global optimization technique can be used on CBPP-CI as a general search framework, it lacks adaptive adjustments, including the search strategy and evaluation function. Otherwise, the search efficiency is poor, and it is hard to find an iterative optimization method to make further improvements based on the current solution.

As the CBPP-CI is a new problem, there are no available benchmark instances. Following our previous works on the square bin packing problem with circular items (SBPP-CI) in (He et al., 2021; He and Dosh, 2017), we choose two categories of benchmarks for the single circle packing problem (SCPP) on the packomonia website² and build two sets of new benchmark instances based on them for the CBPP-CI. For the solving method, we first propose a greedy heuristic based on the designed tangent occupying action (TOA), which can quickly obtain a competitive packing result. TOA always places the current circular item tangent to any two packed items or the bin boundary. At the same time, we also need the packing item to have a minimum distance to the bin boundary. In this way, items are packed as compact as possible, and the remaining space can all gather in the center area of a bin. To judge whether an optimal solution has been

found, we continue to design adaptive simulated annealing with greedy search (ASA-GS) method inspired by related works (He et al., 2021; Hifi et al., 2004; Geng, Chen, Yang, Shi and Zhao, 2011). In contrast to the TOA algorithm, we apply a globalization approach that improves the packing pattern iteratively. We first present an energy function to be minimized and offer an initial packing solution. Then we try to seek more adaptive parameter control to improve the solution quality on large-scale instances. Besides, we utilize the greedy search strategy to achieve faster convergence. Finally, to avoid falling into local optimal solutions, we propose two novel perturbation strategies, and the experiments have verified their effectiveness. Moreover, the packing density of ASA-GS on the top few bins is much higher than the best results for the single circle packing problem on the packomonia website, which indicates the high quality of our solution.

The main contributions of this work are summarized as follows:

- We address a new and important variant of BPP termed CBPP-CI, which comprises packing circular items into multiple circle bins as compactly as possible to minimize the number of used bins. Moreover, we build two sets of new benchmark instances for CBPP-CI.
- We propose a constructive greedy algorithm based on the devised tangent occupying action that can quickly generate a competitive solution.
- We define an energy function for simulated annealing and present two novel perturbation methods (*sector perturbation* and *circle perturbation*) to generate neighbor solutions. Besides, we incorporate a greedy search to achieve faster convergence.
- The parameters are adaptive along with the number of items such that our algorithm can obtain the better solution for the CBPP-CI with a broad scale.

The rest of this paper is organized as follows: Section 2 presents a formal definition of the CBPP-CI and our alternate optimization function, which could help find denser packing so as to minimize the objective. Section 3 gives some definitions and proposes the constructive algorithm. Section 4 presents two perturbation operators and describes the ASA-GS algorithm in detail. Section 5 shows and analyzes the experimental results. Section 6 concludes the work with future work recommendations.

2. Preliminary

In the proposed circle bin packing problem with circular items (CBPP-CI), we are given $n(n \in N^+)$ circular items C_1, C_2, \dots, C_n with radius r_1, r_2, \dots, r_n , and a set of n identical circular bins with radius R (w.l.g. for any circular item $C_i, r_i \leq R$), we aim to determine the center coordinates of each item C_i in a bin such that all items are packed

²www.packomonia.com

Table 1
Variable definition.

Variable	Description
n	Number of circular items
C_i	The i -th circular item
r_i	Radius of C_i
(x_i, y_i)	Center coordinates of C_i
B_k	The k -th bin
R	Radius of the circular bins
I_{ik}	Indicator of whether C_i is in the k -th bin
Y_k	Indicator of whether the k -th bin is used
d_{ij}	Distance between points (x_i, y_i) and (x_j, y_j)

feasibly, i.e. with all circular items fitting completely inside the bins and no overlapping exists between any pair-wise items (i.e. $C_i \cap C_j = \emptyset$). The goal is to minimize the number of used bins, denoted as K ($1 \leq K \leq n$).

2.1. Problem Formulation

Assume that the center of each circular bin B_k is located at (R, R) in two-dimensional Cartesian coordinate system and denote the center of each circular item C_i as (x_i, y_i) . We can define a packing solution as $X = \{ \langle x_1, y_1, b_1 \rangle, \langle x_2, y_2, b_2 \rangle, \dots, \langle x_n, y_n, b_n \rangle \}$, where b_i is the indicator that the placement of item C_i in the b_i -th bin B_{b_i} ($b_i \in \{1, \dots, K\}$). In order to formulate the problem, a summary of necessary variables is listed in Table 1.

The CBPP-CI problem can be formalized as minimizing K while satisfying the following constraints:

$$\sum_{k=1}^n I_{ik} = 1, \quad (1)$$

where $I_{ik} \in \{0, 1\}$ and $i, k \in \{1, \dots, n\}$, implying that each circular item is packed exactly once. CBPP-CI also requires that any pair-wise items in the same bin (i.e. $I_{ik} = I_{jk} = 1$, $\forall i, j, k \in \{1, \dots, n\}$) must not overlap:

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

Third, to ensure that every circular item is placed entirely inside a bin, CBPP-CI requires:

$$\sqrt{(x_i - R)^2 + (y_i - R)^2} + r_i \leq R. \quad (3)$$

Finally, we use Y_k to indicate whether there exist circular items packed into a bin B_k :

$$Y_k = \begin{cases} 1, & \text{if } \sum_{i=1}^n I_{ik} > 0, i, k \in \{1, \dots, n\}, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

And the goal is to minimize the summation of Y_k :

$$\min K = \sum_{k=1}^n Y_k, \quad (5)$$

and clearly $1 \leq K \leq n$.

We could associate the items in bin B_k as an item set, denoted as S_k . So a solution can be obtained by two steps: we first partition the items into different sets $S = \langle S_1, S_2, \dots, S_K \rangle$ for the bins; then we try to pack the items of S_k into bin B_k without overlapping. An optimal packing is that the number of bins used can not be reduced any further.

2.2. Optimization Function

The overall goal of the CBPP-CI is to use as few bins as possible to pack the n circular items C_i , as shown in Eq. (5). However, to attain the global optimum, it is necessary to consider a more local objective function that focuses on packing as tightly as possible. In this regard, suppose that a packing solution X corresponds to a partition $S = S_1 \cup S_2 \cup \dots \cup S_K$ such that S_k is the set of circular items that are packed in bin B_k , and $k \in \{1, \dots, K\}$. Let A be the area of a bin (all bins are identical). Then, the density of packing S_k into a bin B_k is given by:

$$d_{B_k}(X) = \frac{1}{A} \sum_{C_i \in S_k} \pi r_i^2, \quad \text{where } A = \pi R^2. \quad (6)$$

Given a packing solution X and $k \in \{1, \dots, n\}$, let $d_{\min} = \min\{d_{B_k}(X) | 1 \leq k \leq K\}$ and $d_{\max} = \max\{d_{B_k}(X) | 1 \leq k \leq K\}$. A useful local optimization function is defined as follows:

$$v(X) = d_{\max} - d_{\min}. \quad (7)$$

The greater the value of $v(\cdot)$, the higher the quality of a feasible solution X . Since an increment in $v(\cdot)$ corresponds to a tighter packing as some items move from sparser bins to the denser bins.

Further, we need to minimize the value of K , i.e., to maximize the value of $-K$. So we define our optimization function as:

$$\max F(X) = -K + d_{\max} - d_{\min}. \quad (8)$$

The greater the value of $F(\cdot)$ is, the better and tighter the packing is.

Note that $0 \leq d_{\max} - d_{\min} \leq 1$, this term is used for regularization. It implies that the optimization function is more inclined to use fewer bins, and the difference in the number of bins is enough to weigh different solutions. When two feasible packings use the same number of bins, we will focus on each candidate solution's densest bin and the sparsest bin. The denser the densest bin is, the less the wasted space is. The more sparse the sparsest bin is, the more concentrated and complete the remaining still-reserved space is, making it easier to pack the following circular items. Therefore, we assume such a difference in density could determine the quality of candidate solutions.

3. Tangent Occupying Action Algorithm

This section introduces the concept of tangent occupying action and then proposes a constructive greedy algorithm based on this action. We want to pack circular items into the bins as compact as possible through the tangent occupying action to reduce the number of bins used.

3.1. Definitions

We first provide several essential definitions, especially the tangent occupying action.

Definition 1. (Tangent occupying action). A tangent occupying action (TOA) is a packing action that chooses an outside circular item to place to a position inside a bin such that the item is tangent to any two or more packed items (the circular bin can be regarded as a special hollow item).

Definition 2. (Quality of a feasible packing position). For an item, the quality of a feasible packing position is determined by the distance between the center of the packing item and the circular bin's boundary:

$$d(x, y) = R - \sqrt{(x - R)^2 + (y - R)^2} - r, \quad (9)$$

where (x, y) is the center of the circular item. The smaller the interval, the better the packing position.

All feasible positions are sorted in the ascending order of $d(x, y)$ for a circular item in the current bin. A smaller $d(x, y)$ is better, which allows more concentrated free space in favor of placing the remaining circular items. The idea is to pack circular items nearer to the bin's boundary.

3.2. TOA Algorithm

Algorithm 1: TOA Algorithm

Input: A vector of unassigned circle's ID:
 circle_ids, a vector of bin's ID: bin_ids,
 bin's radius: R ;
Result: For each circle C_i , find a bin B_k , and place
 the circle center at (x_i, y_i) ;

```

1 for  $i \in \text{circle\_ids}$  do
2      $\text{vector} < \text{TOA} > s = \emptyset$ ;
3      $\text{bin\_id\_idx} = 0$ ;
4     while true do
5         if  $\text{bin\_id\_idx} == \text{bin\_ids.size}()$  then
6             return false;
7         end
8          $s \leftarrow$  Compute feasible packing positions for
            $C_i$ ;
9         if  $s \neq \emptyset$  then
10            break;
11        end
12         $\text{bin\_id\_idx} \leftarrow \text{bin\_id\_idx} + 1$ ; // Turn to
           the next bin
13    end
14    TOA  $\text{best\_toa} =$  Select the best packing
           position from  $s$  with  $d(x, y)$ ;
15     $\text{circles}[i].x = \text{best\_toa}.p.x$ ;
16     $\text{circles}[i].y = \text{best\_toa}.p.y$ ;
17    Place the circles[i] into the bin_ids[  $\text{bin\_id\_idx}$  ]
           bin;
18 end
```

Details of the TOA algorithm are presented in Alg. 1. It works by packing circular items sequentially in a particular order of their radii (e.g., from large to small). To load the

current item, we first locate all the TOAs of the first bin that satisfies the problem constraints. If there is no available TOA, we seek the next bin to continue searching feasible TOAs until at least one available TOA occurs. Among all possible TOAs, we select the placement with the minimal distance $d(x, y)$ and place the item at (x, y) in the current bin. The TOA algorithm iterates the above procedure until all circular items have been loaded into the bins without overlapping. With this process, TOA prefers positions closer to the bin's boundary. Hence, it packs the circular items as compact as possible and utilizes the bin space greedily to minimize the number of bins used.

TOA is very fast in constructing a solution, but it could not obtain a solution with excellent quality. Therefore, we present two novel mutations and introduce a meta-heuristic global optimization approach called ASA-GS to improve the solution quality.

4. Adaptive Simulated Annealing with Greedy Search

Simulated annealing (SA) algorithm (Kirkpatrick, Gelatt and Vecchi, 1983) has been extensively developed and widely used in many optimization problems. It can avoid getting trapped in the local optimum and attain better solutions by accepting worse solutions with a certain probability. To strengthen the packing solution, we propose a boosted algorithm called the adaptive simulated annealing with greedy search (ASA-GS) for the CBPP-CI. Our method is inspired by the works (He et al., 2021; Hifi et al., 2004; Geng et al., 2011) that can guide the algorithm quickly converging to optimal solutions.

The ASA-GS algorithm (Geng et al., 2011) is described in Alg. 2. In ASA-GS, there are several decisions to be made: how to define the energy function $f(\cdot)$; how to attain an initial solution; how to generate a neighbor solution; how to determine the assignments of parameters such as the probability of accepting a new solution, and the current temperature.

In what follows, we show how one can use the principle of the ASA-GS algorithm to solve the CBPP-CI.

4.1. Energy Function

According to our defined optimization function of the packing problem, we define the energy function $f(\cdot)$ as $-F(\cdot)$ for the simulated annealing algorithm:

$$f(X) = -F(X) = K - d_{max} + d_{min}. \quad (10)$$

It can be seen from Eq. (10) that minimizing the energy function $f(\cdot)$ is equivalent to maximizing the optimization function $F(\cdot)$. Therefore, the smaller the value of $f(\cdot)$, the better a packing solution.

4.2. Initial Packing Solution

We can easily obtain an initial packing solution using n circular bins and assigning each circular item C_i in bin B_i as

Algorithm 2: ASA-GS Algorithm

Input : Bin radius R , a set of n circular items $\{C_i | 1 \leq i \leq n\}$ with radii r_1, \dots, r_n ($r_i \geq r_{i+1}$)

Output: A dense packing solution \mathbf{X} for CBPP-CI.

- 1 Initialize the annealing parameters $t_{start}, t_{cool}, N, t_{greedy}$, and set $t_{current} = t_{start}, G = 0$;
- 2 Initialize a packing solution \mathbf{X}_0 and let $\mathbf{X} = \mathbf{X}_0$;
- 3 **for** $i \leftarrow 1$ **to** N **do**
- 4 Select one perturbation method between sector perturbation and circle perturbation;
- 5 Compute $dE = f(X') - f(X)$;
 // See Subsection 4.3.3 and Algorithm 6 for details
- 6 $X' \leftarrow$ Generate a new packing solution(X, R);
- 7 **if** $dE \leq 0$ **then**
- 8 $X = X'$; // Accept the new solution
- 9 **else**
- 10 $G = G + 1$ and compute $f(X'_G)$;
- 11 **if** $G \geq t_{greedy}$ **then**
- 12 Select X'_{best} with condition $f(X'_{best}) = \min(f(X'_1), f(X'_2), \dots, f(X'_{t_{greedy}}))$;
- // Accept the best solution with probability p
- 13 **if** $e^{(-dE/t_{current}) \times \log(n/2)} \geq \text{rand}(0, 1)$ **then**
- 14 $X = X'_{best}$;
- 15 **end**
- 16 **else**
- 17 Continue to generate next neighbor solution;
- 18 **end**
- 19 **end**
- 20 $t_{current} = t_{current} \times t_{cool}$ and let $G = 0$;
- 21 **if** $t_{current} \leq t_{end}$ **then**
- 22 **break**;
- 23 **end**
- 24 **end**

shown in Fig. 1.

$$\text{Initialize_packing_solution}(P) = \{ \langle R, r_i, B_i \rangle | i \in \{1, \dots, n\} \} \quad (11)$$

4.3. Generate Neighbor Solutions

Generally, a new neighbor solution is obtained by conducting a local disturbance to the current solution. Here an effective perturbation strategy plays a significant role in heuristic algorithms in the local search process to solve the optimization problem. Besides, different perturbation methods usually have different impacts on the specific problem. To void falling into local optimum, we design two new perturbation strategies for the CBPP-CI, termed circle perturbation and sector perturbation.

4.3.1. Circle perturbation

As Alg. 3 shows, the circle perturbation strategy selects a circular item randomly in a circular bin B_k , then generates a circular area with the item's center as its center, the radius of the circular area is a random number in $[0, \frac{R}{2}]$. It guarantees that at least one item will intersect the generated circular area. In most cases, more than one item will cross this area and be reassigned at each iteration.

Algorithm 3: Pseudo-code of sampling a circle

Input : Bin B_k , bin radius R

Output: A circular area with $\langle x, y, r \rangle$

// Each circle is represented as $\langle x, y, r \rangle$

- 1 $r \leftarrow \text{random_real}(R/2)$; // The circle radius is r
- // Randomly select a circular item from B_k
- 2 **if** ($\neg B_k.\text{empty}()$) **then**
- 3 $i \leftarrow \text{random_ints}(1, \{i | C_i \in B_k\})$;
- 4 **end**
- 5 $x \leftarrow C_i.x$;
- 6 $y \leftarrow C_i.y$;
- 7 $\text{circle} = \text{Circle}(x, y, r)$; // Generate a circle area

Algorithm 4: Pseudo-code of sampling a sector

Input : Size of the central angle $\Delta\theta$, is_fixed

Output: A sector with (α, β)

// Each sector is represented as (α, β)

- 1 $\alpha \leftarrow \text{randInt}(0, 360)$;
- 2 **if** is_fixed **then**
- 3 $\Delta\theta \leftarrow \text{randInt}(20, 60)$;
- 4 **end**
- 5 $\beta \leftarrow (\alpha + \Delta\theta) \% 360$;
- 6 $\text{sector} = \text{Sector}(\alpha, \beta)$; // Generate a sector area

4.3.2. Sector perturbation

As Alg. 4 shows, the sector perturbation strategy randomly generates a sector area (α, β) in a circular bin. The larger the central angle, the larger the sector area. Therefore, the larger the disturbance, the more circular items intersecting the area will be reassigned at each iteration. Especially the circular items are taken out and unassigned from the border to the center of the circular bin.

Alg. 5 can determine whether a circular item C_i intersects the selected circular (or sector) area. If a circular item C_i intersects the chosen area, that is, the item with a red dotted border in Fig. 2 or Fig. 3, which will be taken out from the bin and added to the unassigned circular set circle_ids in Alg. 6 (line 5). However, as the sector area is not easy to express with mathematical formulas like the circle area, it is not intuitive to judge whether a circular item intersects with the sector area. Due to the sector is surrounded by two radii and the arc opposite the central angle. We turn it into two small subproblems: 1) whether the center of a circular item is in the sector area; 2) whether the circular item intersects the radii r_α or r_β of the central angle. The former is judged

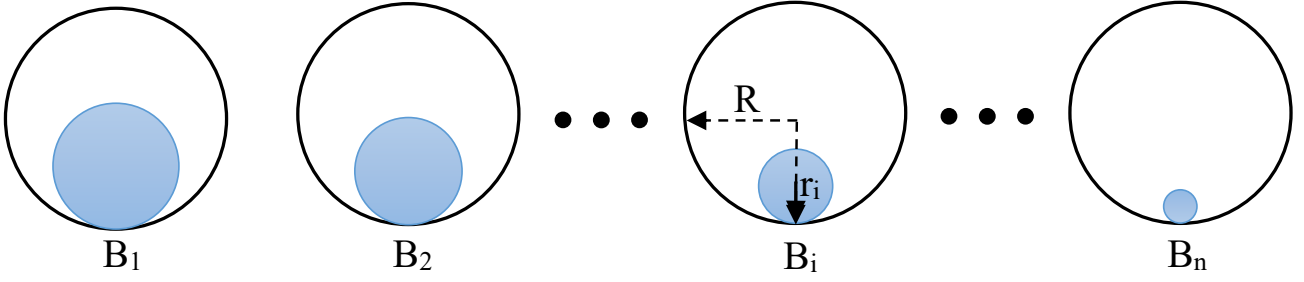


Fig. 1: Initialize a packing solution.

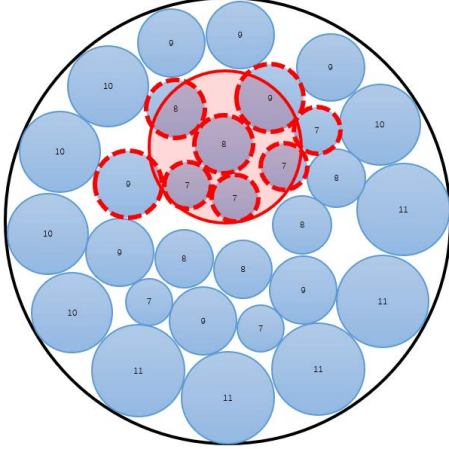


Fig. 2: An illustration of circle perturbation.

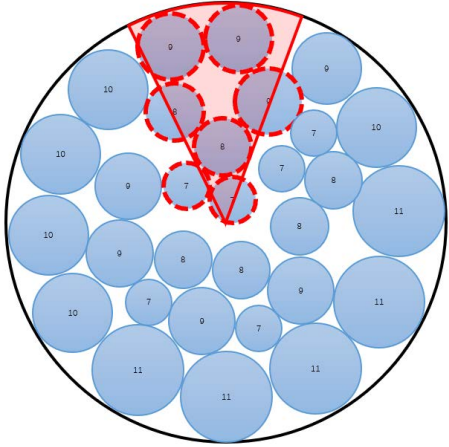


Fig. 3: An illustration of sector perturbation.

by line 4, Alg. 5, and the latter is implemented by line 7, Alg. 5.

4.3.3. Generate Neighbor Solution

As Alg. 6 shows, a new packing solution X' is generated from the old packing solution X by selecting two bins B_{k_1}, B_{k_2} randomly and performing *sector perturbation* or *circle perturbation*. We randomly choose a sector area with equal angle size in each bin, and all items that intersect the

Algorithm 5: Pseudo-code of intersecting with the sector or circle

Input : Circle C_i , sector S (or circle C), bin radius R
Output: True or false
 // Returns if C_i intersects the perturbation area S (or C)

```

1 if adopt the sector perturbation then
2    $\alpha = S.\alpha$ ;
3    $\beta = S.\beta$ ;
4   if the center of  $C_i$  is in sector area  $S$  then
5     return true; // Intersects with the sector area  $S$ 
6   end
7   if  $C_i$  intersects with radii  $r_\alpha$  or  $r_\beta$  of  $S$  then
8     return true; // Intersects with the sector area  $S$ 
9   end
10  return false; // No intersects with the sector area  $S$ 
11 else
12   // adopt the circle perturbation
13   if  $(C_i.x - C.x)^2 + (C_i.y - C.y)^2 \leq C_i.r + C.r$  then
14     return true; // Intersects with circle area  $C$ 
15   else
16     return false; // No intersects with circle area  $C$ 
17   end
18 end
    
```

sector area will be taken out, and their IDs will be added to set *circles_ids*. k_1, k_2 will be added to set *bin_ids*. The unassigned circular items will be reassigned with algorithm TOA. Then we will get a new neighbor packing solution X' .

At each iteration, two (or more) bins will be selected so that the unassigned circular items have more free space to be assigned. Even in the worst case, the algorithm will attempt to exchange the circular items in the two (or more) areas, ensuring that there will be some disturbance at each operation.

Algorithm 6: Pseudo-code of generating a new solution

Input : The old packing solution: X , bin radius: R
Output: A new neighbor packing solution: X'

- 1 $K \leftarrow X.bins.size()$;
 // Select two bins randomly
- 2 $(k_1, k_2) \leftarrow \text{random_ints}(2, \{1, \dots, K\})$;
 // Select a sector or circle area in the first bin using Algorithm 4 or Algorithm 3
- 3 $A_1 \leftarrow \text{Sample a sector}(\theta, is_fixed)$; // or Sample a circle(B_k, R);
 // Select an area in the second bin
- 4 $A_2 \leftarrow \text{Sample a sector}(\theta, is_fixed)$; // or Sample a circle(B_k, R);
- 5 $circle_ids \leftarrow \bigcup_{j \in \{1,2\}} \{i | \langle x_i, y_i, k_j \rangle \in X \wedge I_{ik_j} = 1 \wedge \text{intersects}(C_i, A_j, R) == True\}$; // See Algorithm 5
- 6 $bin_ids \leftarrow \{k_1, k_2\}$;
- 7 $X' \leftarrow TOA(circle_ids, bin_ids, R)$; // Generate a new solution

Besides, at the early stage, the sector area (i.e., $\Delta\theta$) can be set larger so that the new neighbor solution can be located far away from the current solution to speed up the search process and to avoid getting trapped at a local minimum solution. Once the temperature $f(X)$ gets low, the sector area will become smaller. The new solution will be generated nearby with the minor disturbance and focus on the local area.

4.4. The Assignments of Parameters

In the experiments, we find that different assignments of parameters are suitable for different problem scales. Therefore, to obtain a better solution in solving the packing problem in a broad scale, the parameter values should change along with the number of items, which can make the assignments of parameters dynamic and adaptive, such as the times of greedy search t_{greedy} :

$$t_{greedy} \leftarrow \beta \times n, \quad (12)$$

and the cool coefficient of the temperature t_{cool} :

$$t_{cool} \leftarrow \frac{\alpha \times \sqrt{n} - 1}{\alpha \times \sqrt{n}}. \quad (13)$$

In this way, our algorithm is adaptive for the number of items, and the parameter space can be sampled much more efficiently. For example, if n is small, we will get a quick cooling coefficient. As the number of items increases, the times of greedy search will become larger and get a slower cooling coefficient fit for big-scale packing instances. The difficulty of the problem becomes higher as the number of items becomes larger, indicating more solution space to be explored.

4.5. The Overall ASA-GS Algorithm

The workflow of ASA-GS is provided in Alg. 2. Firstly, it is necessary to initialize the annealing parameters and attain a feasible packing pattern with the initial solution as shown in subsection 4.2. Then, it will select one of the perturbation methods between sector perturbation and circle perturbation as well as generate a new neighbor packing solution by Alg. 6. After that, it will compute the energy function and utilize the greedy search technique based on the simulated annealing to decide whether accept the new solution. Finally, it updates the parameters such as the cooling coefficient of the temperature with $t_{current} = t_{current} \times t_{cool}$, and the acceptance probability by Eq. (14). The process will execute until the terminal criterion such as the current temperature $t_{current}$ is below the threshold t_{end} , or the number of iterations i exceeds the given value N .

The key concept of greedy search can be described as follows: take a new neighbor packing X' (i.e. X'_1) as the best packing X when $dE \leq 0$ (i.e. $f(X') \leq f(X)$), and go to the next step. Otherwise the algorithm continues to generate the next new neighbor packing X'_2 , and takes it as the best packing X when $f(X'_2) \leq f(X)$, then goes to the next step. Otherwise this step will continue to be executed until attaining a better packing solution or has generated $t_{greedy} - th$ new neighbor packing $X'_{t_{greedy}}$. The latter will generate t_{greedy} neighbor packing solutions $X'_1, X'_2, \dots, X'_{t_{greedy}}$ while they are all worse than the original packing solution X . In such case, it will accept the best new packing X'_{best} among the t_{greedy} neighbor packing solutions generated with probability p .

$$p \leftarrow e^{-(f(X')-f(X))/t_{current} \times \log(n/2)}. \quad (14)$$

Obviously, the quality of the best neighbor solution X'_{best} will vary from low to high with the times of greedy search increases so that the new solution can jump to a better solution space with high probability. $f(X'_{best})$ is defined by Eq. (15):

$$f(X'_{best}) = \min(f(X'_1), f(X'_2), \dots, f(X'_G), \dots, f(X'_{t_{greedy}})). \quad (15)$$

The ASA-GS algorithm can achieve faster convergence and improve the quality-time trade-off by utilizing the greedy search technique. As the experimental results show, the solutions produced by the ASA-GS algorithm are very competitive.

5. Experiments

For experiments, we evaluate and analyze the competency and performance of the proposed algorithms, TOA and ASA-GS. We implemented the algorithms using Visual C++ programming language. All results were generated by setting parameters as $N = 2 \times 10^6$, $\alpha = 0.9$, $\beta = 0.08$,

Table 2

 Experimental results on the fixed benchmarks with circular bins for $r_i = i$.

n_0	n	Alg.	Bin_0	bin 1	bin 2	bin 3	bin 4	bin 5	bin 6	F	$F_A - F_T$
8	40	ASA-GS	0.78	0.84	0.80	0.74	0.74	0.71	0.03	-5.19	0.19
		TOA		0.81	0.74	0.72	0.72	0.69	0.19	-5.38	
9	45	ASA-GS	0.79	0.83	0.80	0.77	0.76	0.70	-	-4.87	0.47
		TOA		0.81	0.75	0.74	0.71	0.69	0.15	-5.34	
10	50	ASA-GS	0.80	0.84	0.79	0.79	0.79	0.79	-	-4.95	0.45
		TOA		0.81	0.79	0.74	0.74	0.70	0.21	-5.40	
11	55	ASA-GS	0.80	0.84	0.81	0.81	0.77	0.77	0.07	-5.23	0.29
		TOA		0.83	0.74	0.72	0.71	0.70	0.35	-5.52	
12	60	ASA-GS	0.80	0.85	0.80	0.79	0.77	0.76	0.06	-5.21	0.21
		TOA		0.82	0.79	0.77	0.73	0.68	0.24	-5.42	
13	65	ASA-GS	0.81	0.84	0.82	0.81	0.78	0.76	0.10	-5.26	0.16
		TOA		0.84	0.81	0.75	0.75	0.72	0.26	-5.42	
14	70	ASA-GS	0.81	0.85	0.80	0.79	0.79	0.78	0.12	-5.27	0.11
		TOA		0.86	0.80	0.77	0.74	0.72	0.24	-5.38	
15	75	ASA-GS	0.82	0.85	0.82	0.81	0.79	0.77	0.08	-5.23	0.19
		TOA		0.84	0.79	0.75	0.74	0.72	0.26	-5.42	
16	80	ASA-GS	0.83	0.85	0.82	0.82	0.78	0.78	0.11	-5.26	0.13
		TOA		0.86	0.82	0.77	0.74	0.70	0.25	-5.39	
17	85	ASA-GS	0.83	0.86	0.83	0.81	0.79	0.76	0.11	-5.25	0.12
		TOA		0.86	0.84	0.75	0.74	0.74	0.23	-5.37	
18	90	ASA-GS	0.83	0.86	0.83	0.80	0.79	0.78	0.14	-5.28	0.11
		TOA		0.86	0.83	0.76	0.75	0.75	0.25	-5.39	
19	95	ASA-GS	0.84	0.86	0.83	0.80	0.79	0.77	0.15	-5.29	0.12
		TOA		0.86	0.82	0.77	0.75	0.73	0.27	-5.41	
20	100	ASA-GS	0.84	0.87	0.83	0.80	0.80	0.77	0.13	-5.26	0.12
		TOA		0.86	0.83	0.77	0.76	0.75	0.24	-5.38	

$t_{start} = 0.1$, $t_{end} = 10^{-4}$, and obtained using a computer equipped with an Intel(R) Core(TM) i7-10710U CPU @ 1.10GHz 1.61Hz.

As the CBPP-CI is a new problem, there are no available benchmark instances. Referring to the pioneering work of square bin packing problem with circular items (SBPP-CI) (He et al., 2021). We choose two categories of benchmarks for the single circle packing problem (SCPP) on the packomonia website and build two sets of new benchmark instances based on them for the CBPP-CI.

The generated instances consist of strong heterogeneous $r_i = i$ (i.e., the circle radii vary widely), and $r_i = \sqrt{i}$ for weakly heterogeneous instances. For each category, we produce fixed and random instances. We first choose instances from the packomonia website for SCPP to generate our instances. Each circular bin's best-known solution found in Eckardi (2018) ranges from 8 to 20 from the circular bin benchmarks. The fixed set of benchmarks contains exactly five copies of each circle instance, and for the random benchmarks instances, it contains a random copy of each circular item that ranges from 2 – 10 from the same benchmarks. We fix the circular bin size from the best solution found on the packomonia website.

In the computational tables, we list 52 generated instances from the two categories of benchmarks (fixed and rand). For each instance in the Tables (2, 3, 4 and 5), we have results for two algorithms: ASA-GS and TOA. Column n_0

represents the original index number of the circle set for each instance, column n represents the actual number of replicated circles in the CBPP-CI instance. The third column (i.e., Alg.) represents the two algorithms. Column Bin_0 is only for the fixed benchmarks representing the reference value indexed from Eckardi (2018) for the state-of-the-art results. Columns 5^{th} to 10^{th} denote the density (bin occupancy rate) for each bin. Lastly, the F and $F_A - F_T$ columns represent the actual measure value achieved for each algorithm and relative improvement of ASA-GS over TOA.

5.1. Comparison on $r = i$

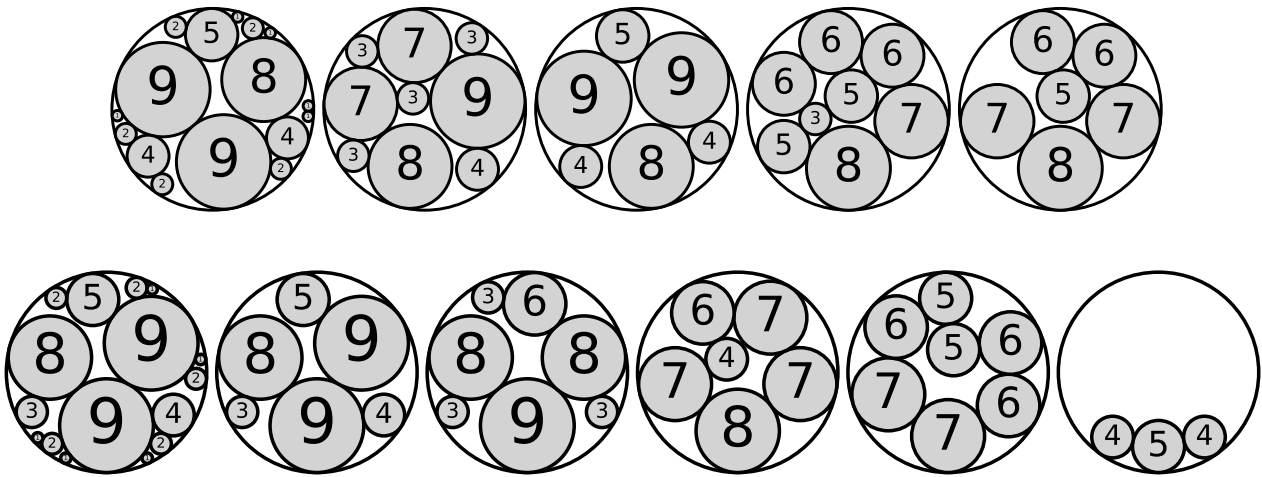
Here $r = i$ is a benchmark instance that has a wide variation of circle sizes. In this set of benchmarks, we execute ASA-GS and TOA algorithms for comparison. We select instances that range from 8 to 20 for both fixed and random setup from the benchmark. Table 2 displays the computational results of fixed benchmarks while Table 3 displays for random benchmarks.

In Table 2 we can notice that the objective value of ASA-GS is better than TOA on all the instances, and in addition, we can also observe one lesser bin occupancy rate. for instance, $n_0 = 9$ & $n = 45$ and $n_0 = 10$ & $n = 45$, i.e., ASA-GS uses five bins to pack 45 circles while TOA uses six bins to load the same set of circular items, for a diagrammatic representation of the packing layout when $n_0 = 9$ & $n = 45$ (See Fig. 4) and when $n_0 = 10$ & $n = 50$, we can also

Table 3

 Experimental results on the random benchmarks with circular bins for $r_i = i$.

n_0	n	Alg.	bin 1	bin 2	bin 3	bin 4	bin 5	bin 6	F	$F_A - F_T$
8	35	ASA-GS	0.84	0.81	0.71	-	-	-	-2.87	0.36
		TOA	0.84	0.77	0.68	0.07	-	-	-3.23	
9	44	ASA-GS	0.84	0.80	0.80	0.70	-	-	-3.86	0.41
		TOA	0.82	0.75	0.75	0.73	0.09	-	-4.27	
10	48	ASA-GS	0.84	0.84	0.79	0.70	-	-	-3.86	0.39
		TOA	0.83	0.80	0.74	0.72	0.08	-	-4.25	
11	52	ASA-GS	0.85	0.84	0.82	0.71	-	-	-3.86	0.39
		TOA	0.85	0.80	0.74	0.73	0.10	-	-4.25	
12	59	ASA-GS	0.85	0.82	0.81	0.72	-	-	-3.87	0.37
		TOA	0.85	0.80	0.74	0.72	0.09	-	-4.24	
13	64	ASA-GS	0.85	0.83	0.83	0.73	-	-	-3.88	0.35
		TOA	0.85	0.80	0.77	0.74	0.08	-	-4.23	
14	67	ASA-GS	0.85	0.82	0.79	0.79	0.16	-	-4.31	0.07
		TOA	0.87	0.81	0.77	0.72	0.25	-	-4.38	
15	73	ASA-GS	0.87	0.82	0.78	0.65	-	-	-3.78	0.10
		TOA	0.85	0.80	0.74	0.73	-	-	-3.88	
16	79	ASA-GS	0.86	0.83	0.81	0.80	0.73	-	-4.87	0.39
		TOA	0.86	0.82	0.77	0.76	0.70	0.12	-5.26	
17	84	ASA-GS	0.86	0.84	0.80	0.78	0.10	-	-4.24	0.08
		TOA	0.86	0.82	0.78	0.74	0.18	-	-4.32	
18	87	ASA-GS	0.87	0.83	0.80	0.80	0.71	-	-4.84	0.44
		TOA	0.85	0.82	0.75	0.74	0.72	0.13	-5.28	
19	92	ASA-GS	0.87	0.84	0.82	0.80	0.06	-	-4.19	0.09
		TOA	0.86	0.83	0.80	0.76	0.14	-	-4.28	
20	97	ASA-GS	0.87	0.84	0.81	0.74	-	-	-3.87	0.40
		TOA	0.86	0.82	0.74	0.71	0.13	-	-4.27	


Fig. 4: Packing layouts generated by ASA-GS (top) and TOA (bottom) for the fixed benchmark $r = i$ with 9×5 circles.

notice that ASA-GS packs 50 circles in 5 bins while TOA uses six bins for the same set of circular items. For the fixed benchmarks, ASA-GS has an average of 21% improvement.

For the random benchmarks, we can also observe that in all the instances, ASA-GS returns a feasible solution compared to TOA with an overall average improvement of 30% for $r = i$ benchmarks in Table 3. We show the packing

layout when $n_0 = 11$ & $n = 52$ in Fig. 5 for the random benchmarks.

5.2. Comparison on $r_i = \sqrt{i}$

Here $r_i = \sqrt{i}$ has a smaller variation of the circle's radii. Similarly, we test the two algorithms on this set of benchmarks ranging from 8 – 20 instances. Table 4 and 5 represent fixed and random benchmarks respectively.

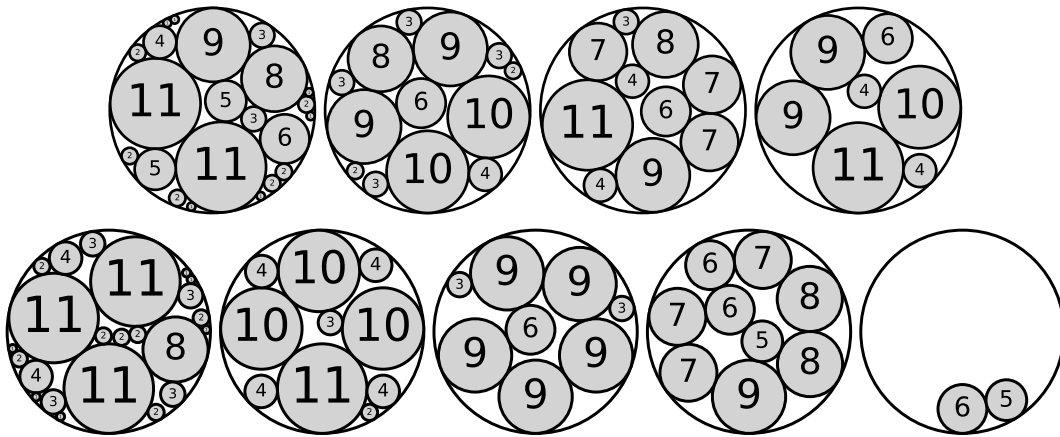


Fig. 5: Packing layouts generated by ASA-GS (top) and TOA (bottom) for the random benchmark $r_i = i$ with 11 – 52 circles.

Table 4

Experimental results on the fixed benchmarks with circular bins for $r_i = \sqrt{i}$.

n_0	n	Alg.	Bin_0	bin 1	bin 2	bin 3	bin 4	F	$F_A - F_T$
8	40	ASA-GS	0.76	0.84	0.78	0.76	0.39	-3.55	0.09
		TOA		0.81	0.76	0.76	0.45	-3.64	
9	45	ASA-GS	0.77	0.83	0.81	0.76	0.37	-3.54	0.12
		TOA		0.83	0.75	0.70	0.49	-3.66	
10	50	ASA-GS	0.80	0.83	0.80	0.78	0.39	-3.56	0.09
		TOA		0.83	0.75	0.74	0.48	-3.65	
11	55	ASA-GS	0.81	0.86	0.80	0.79	0.37	-3.51	0.06
		TOA		0.85	0.78	0.74	0.42	-3.57	
12	60	ASA-GS	0.81	0.85	0.81	0.80	0.32	-3.47	0.09
		TOA		0.86	0.76	0.75	0.42	-3.56	
13	65	ASA-GS	0.82	0.85	0.81	0.79	0.37	-3.52	0.07
		TOA		0.85	0.78	0.74	0.44	-3.59	
14	70	ASA-GS	0.82	0.86	0.80	0.78	0.37	-3.51	0.13
		TOA		0.83	0.76	0.75	0.47	-3.64	
15	75	ASA-GS	0.82	0.85	0.81	0.79	0.38	-3.53	0.08
		TOA		0.86	0.76	0.75	0.47	-3.61	
16	80	ASA-GS	0.83	0.86	0.80	0.80	0.39	-3.53	0.06
		TOA		0.85	0.79	0.76	0.44	-3.59	
17	85	ASA-GS	0.83	0.86	0.80	0.80	0.39	-3.53	0.07
		TOA		0.85	0.78	0.77	0.45	-3.60	
18	90	ASA-GS	0.84	0.87	0.81	0.80	0.38	-3.51	0.06
		TOA		0.87	0.79	0.77	0.44	-3.57	
19	95	ASA-GS	0.84	0.87	0.81	0.79	0.40	-3.53	0.07
		TOA		0.86	0.78	0.77	0.46	-3.60	
20	100	ASA-GS	0.84	0.86	0.82	0.80	0.40	-3.54	0.06
		TOA		0.87	0.78	0.75	0.47	-3.60	

Table 4 is for the fixed benchmarks, from which we can notice that ASA-GS outperforms TOA in all the instances with an average improvement of 8%. We show the significant improvements of the packing layout in Fig. 6.

We can notice that ASA-GS packs the items with lesser bins than TOA for the random benchmarks in Table 5. We can observe that when $n_0 = 8$ (or 9, 10, 12, 13, 15, 16, 18, 19), ASA-GS uses fewer bins than TOA. We use $n_0 = 16$ & $n = 81$ to demonstrate the packing layout for this

benchmark in Fig. 7. For these random benchmarks, we can notice an overall improvement of 26%.

5.3. Further Analysis

Since our solution methods are stochastic, we further analyze and assess the two proposed algorithms' significance comparison by using a T-tail statistical hypothesis test on $H_0: \mu_T = \mu_A$. H_0 denotes the null hypothesis, which equates to no difference between the results returned by TOA and ASA-GS. We apply the commonly used $\alpha = 0.05$ as our threshold value. For each table we generated the p-value

Table 5

 Experimental results on the random benchmarks with circular bins for $r_i = \sqrt{i}$.

n_0	n	Alg.	bin 1	bin 2	bin 3	bin 4	F	$F_A - F_T$
8	29	ASA-GS	0.83	0.79	-	-	-1.96	0.29
		TOA	0.82	0.72	0.07	-	-2.25	
9	36	ASA-GS	0.84	0.75	-	-	-1.91	0.34
		TOA	0.80	0.74	0.05	-	-2.25	
10	51	ASA-GS	0.86	0.81	0.73	-	-2.87	0.34
		TOA	0.84	0.76	0.76	0.05	-3.21	
11	56	ASA-GS	0.85	0.80	0.68	-	-2.83	0.06
		TOA	0.85	0.75	0.74	-	-2.89	
12	61	ASA-GS	0.86	0.82	0.73	-	-2.87	0.37
		TOA	0.83	0.76	0.75	0.07	-3.24	
13	63	ASA-GS	0.86	0.82	0.74	-	-2.88	0.33
		TOA	0.84	0.79	0.74	0.05	-3.21	
14	66	ASA-GS	0.86	0.81	0.65	-	-2.79	0.12
		TOA	0.83	0.75	0.74	-	-2.91	
15	77	ASA-GS	0.86	0.83	0.73	-	-2.87	0.38
		TOA	0.85	0.76	0.72	0.10	-3.25	
16	81	ASA-GS	0.86	0.81	0.78	-	-2.92	0.36
		TOA	0.86	0.76	0.7	0.14	-3.28	
17	83	ASA-GS	0.86	0.81	0.79	0.07	-3.21	0.12
		TOA	0.85	0.76	0.74	0.18	-3.33	
18	89	ASA-GS	0.86	0.84	0.75	-	-2.89	0.30
		TOA	0.87	0.77	0.76	0.06	-3.19	
19	93	ASA-GS	0.87	0.81	0.77	-	-2.90	0.31
		TOA	0.86	0.80	0.74	0.07	-3.21	
20	97	ASA-GS	0.85	0.81	0.80	0.05	-3.2	0.09
		TOA	0.85	0.78	0.75	0.14	-3.29	

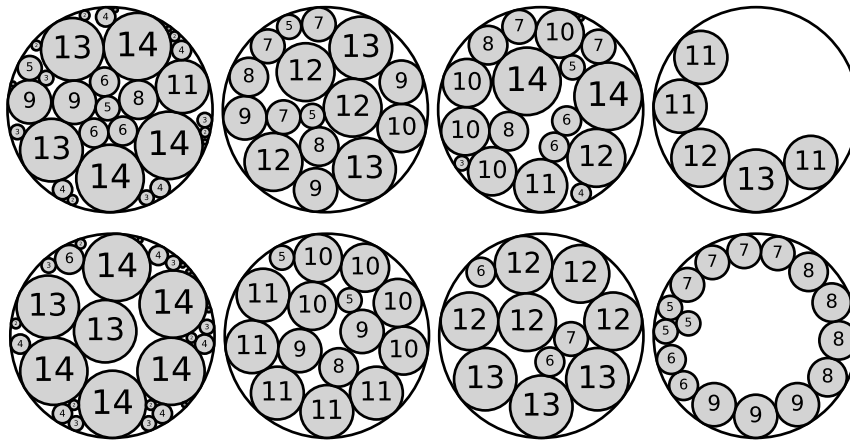

Fig. 6: Packing layouts generated by ASA-GS (top) and TOA (bottom) for the fixed benchmark $r_i = \sqrt{i}$ with 14×5 circles.

Table 6

T-test statistical analysis.

Group	Table	p-value
$r_i = i$	Table 2	0.0000662229
	Table 3	0.0000107770
$r_i = \sqrt{i}$	Table 4	0.0000000252
	Table 5	0.0000037048

and compared with the $\alpha = 0.05$ value as shown in Table 6. We reject the null hypothesis from the generated results and claim with a confidence interval (CI) of 95% that our proposed algorithms are statistically distinct.

To further demonstrate the typical performance pattern of the two algorithms, we illustrate the performance comparisons of the two algorithms.

In Fig. 8, the Y -axis represents the optimization function while the X -axis represents the number of circles (n). A red (blue) line presents ASA-GS (TOA). We can notice a distinct

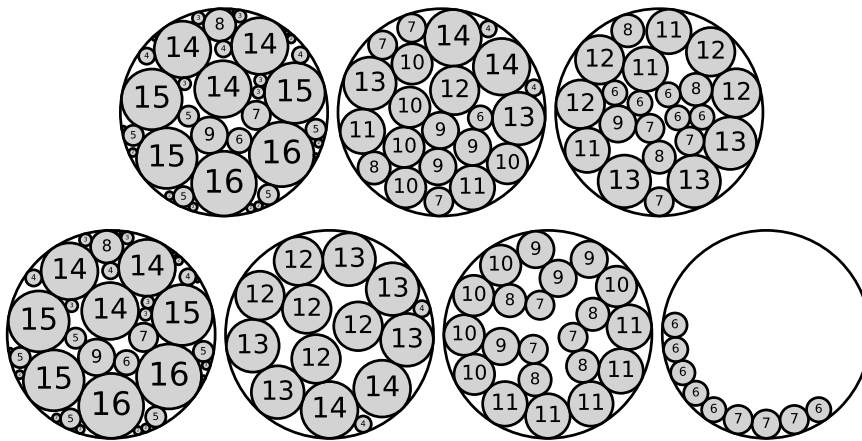


Fig. 7: Packing layouts generated by ASA-GS (top) and TOA (bottom) for the random benchmark $r_i = \sqrt{i}$ with 16 – 81 circles.

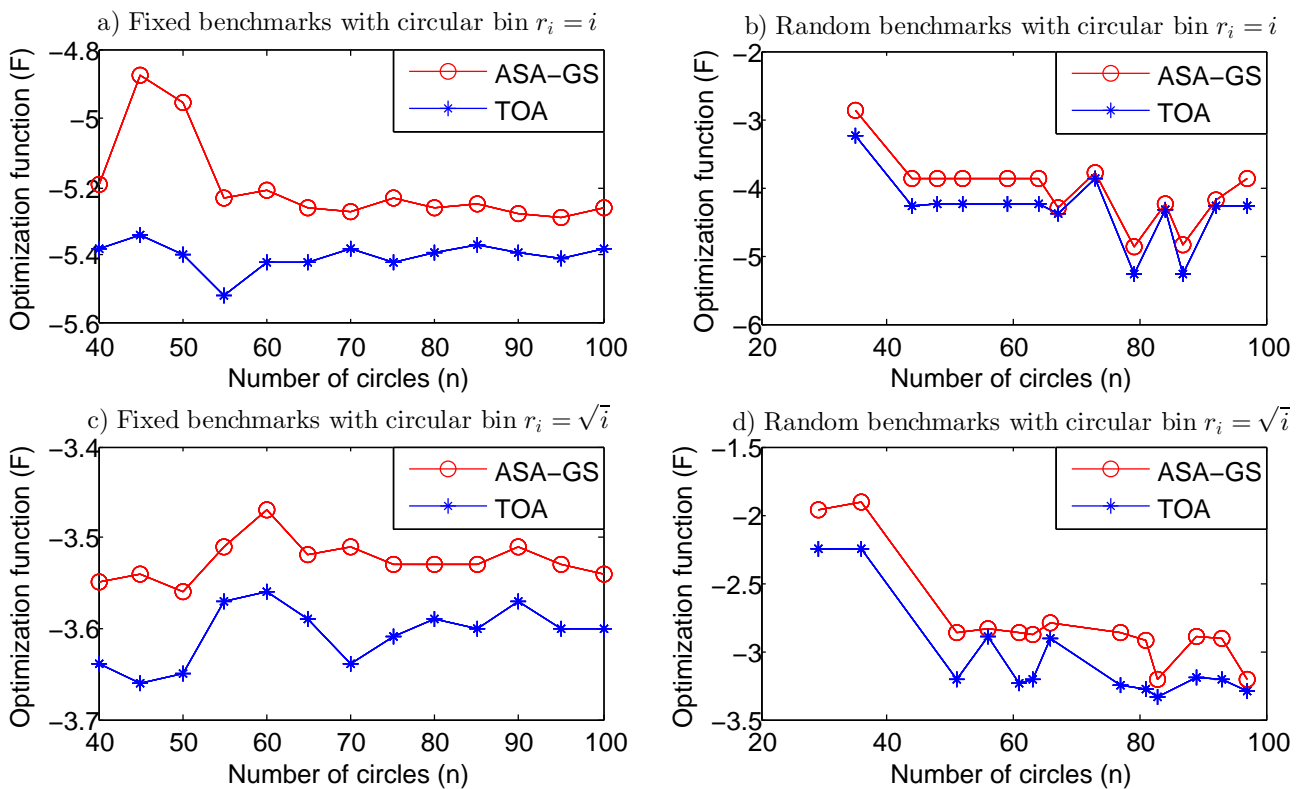


Fig. 8: ASA-GS versus TOA.

variation of ASA-GS and TOA lines that do not intersect, indicating that the ASA-GS completely outperforms the base TOA on all instances.

Lastly, we record the runtimes for $r = i$ and $r_i = \sqrt{i}$ benchmarks as shown in Table 7. The execution time of TOA is in micro-seconds while ASA-GS takes less than 200 seconds. In summary, the performance clearly shows ASA-GS efficiency outperforms TOA in a reasonable amount of time in all the instances. And in some instances, we can notice a reduction in the number of bins used. Moreover, Table 3 and Table 5 show that the density of $bin1$ and $bin2$ is

usually greater than that of Bin_0 . It indicates that the packing density of ASA-GS on the top few bins is much higher than the best packing results for SPP on the packomnia website, inferring the high quality of our solution for the CBPP-CI.

Table 7
Runtimes for ASA-GS execution on all benchmarks.

n_0	$r_i = i$				$r_i = \sqrt{i}$			
	fixed		random		fixed		random	
	n	t	n	t	n	t	n	t
8	40	12	35	34	40	24	29	53
9	45	23	44	32	45	30	36	73
10	50	19	48	36	50	39	51	65
11	55	22	52	41	55	47	56	92
12	60	26	59	59	60	51	61	90
13	65	30	64	68	65	63	63	95
14	70	35	67	45	70	68	66	122
15	75	40	73	98	75	75	77	152
16	80	45	79	60	80	89	81	157
17	85	52	84	70	85	98	83	95
18	90	56	87	86	90	110	89	196
19	95	62	92	82	95	128	93	198
20	100	68	97	144	100	138	97	146

6. Conclusion

In this paper, we introduce a new variant of bin packing problem termed the circle bin packing problem with circular items (CBPP-CI). For packing solutions, we define the tangent occupying action (TOA) to quickly pack the items into a bin as compactly as possible to minimize the number of bins used. Besides, we design a new form of optimization function embedding the number of bins used and the maximum density gap of the bins to evaluate the solution quality. We then propose the adaptive simulated annealing with greedy search (ASA-GS) algorithm to attain better solutions. The greedy search strategy can speed up the convergence rate. Based on the framework of simulated annealing algorithm, the parameters such as the times of greedy search, the acceptance probability are adaptive along with the number of items, which can help to sample the parameter space much more efficiently and attain a better solution for instances in a broad scale. To avoid getting trapped in local optimum, we propose two novel perturbation strategies, *sector perturbation* and *circle perturbation*. Experimental results show that ASA-GS exhibits good performance on the solution quality and computational time. Besides, the packing quality is better than that of the constructive algorithm TOA on all the CBPP-CI instances we generated. As this is a new problem, there is no baseline algorithm available. However, we see that the packing density of ASA-GS on the top few bins is much higher than the state-of-the-art results on the single circle packing problem, indicating the high quality of our solution.

Acknowledgement

This work is supported by National Natural Science Foundation (62076105) and Natural Science Foundation of Jiangsu Province (BK20181409).

References

- Akeb, H., Hifi, M., M'Hallah, R., 2009. A beam search algorithm for the circular packing problem. *Computers & Operations Research* 36, 1513–1528.
- Akeb, H., Hifi, M., Negre, S., 2011. An augmented beam search-based algorithm for the circular open dimension problem. *Computers & Industrial Engineering* 61, 373–381.
- Christensen, H.I., Khan, A., Pokutta, S., Tetali, P., 2017. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review* 24, 63–79.
- Chung, F.R., Garey, M.R., Johnson, D.S., 1982. On packing two-dimensional bins. *SIAM Journal on Algebraic Discrete Methods* 3, 66–76.
- Demaine, E.D., Fekete, S.P., Lang, R.J., 2010. Circle packing for origami design is hard. *arXiv preprint arXiv:1008.1224*.
- Eckardi, S., 2018. Packomania website 2018 www.packomania.com. Packomania.
- Faroe, O., Pisinger, D., Zachariasen, M., 2003. Guided local search for the three-dimensional bin-packing problem. *Informatics journal on computing* 15, 267–283.
- Fu, Z., Huang, W., Lü, Z., 2013. Iterated tabu search for the circular open dimension problem. *European Journal of Operational Research* 225, 236–243.
- Geng, X., Chen, Z., Yang, W., Shi, D., Zhao, K., 2011. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. *Applied Soft Computing* 11, 3680–3689.
- Grosso, A., Jamali, A., Locatelli, M., Schoen, F., 2010. Solving the problem of packing equal and unequal circles in a circular container. *Journal of Global Optimization* 47, 63–81.
- He, K., Dosh, M., 2017. A greedy heuristic based on corner occupying action for the 2d circular bin packing problem, in: *National Conference of Theoretical Computer Science*, Springer. pp. 75–85.
- He, K., Huang, M., Yang, C., 2015. An action-space-based global optimization algorithm for packing circles into a square container. *Computers & Operations Research* 58, 67–74.
- He, K., Huang, W., Jin, Y., 2012. An efficient deterministic heuristic for two-dimensional rectangular packing. *Computers & Operations Research* 39, 1355–1363.
- He, K., Tole, K., Ni, F., Yuan, Y., Liao, L., 2021. Adaptive large neighborhood search for solving the circle bin packing problem. *Computers and Operations Research* 127, 105140. doi:<https://doi.org/10.1016/j.cor.2020.105140>.

- Hifi, M., M'Hallah, R., 2002. A best-local position procedure-based heuristic for two-dimensional layout problems. *Stud. Inform. Univ.* 2, 33–56.
- Hifi, M., Paschos, V.T., Zissimopoulos, V., 2004. A simulated annealing approach for the circular cutting problem. *European Journal of Operational Research* 159, 430–448.
- Huang, W.Q., Li, Y., Li, C.M., Xu, R.C., 2006. New heuristics for packing unequal circles into a circular container. *Computers & Operations Research* 33, 2125–2142.
- Johnson, D.S., 1973. Near-optimal bin packing algorithms. Ph.D. thesis. Massachusetts Institute of Technology.
- Kang, J., Park, S., 2003. Algorithms for the variable sized bin packing problem. *European Journal of Operational Research* 147, 365–372. doi:[https://doi.org/10.1016/S0377-2217\(02\)00247-3](https://doi.org/10.1016/S0377-2217(02)00247-3).
- Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. *science* 220, 671–680.
- Lodi, A., Martello, S., Monaci, M., 2002. Two-dimensional packing problems: A survey. *European journal of operational research* 141, 241–252.
- Lodi, A., Martello, S., Vigo, D., 1999. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing* 11, 345–357.
- López, C.O., Beasley, J., 2016. A formulation space search heuristic for packing unequal circles in a fixed size circular container. *European Journal of Operational Research* 251, 64–73.
- Lü, Z., Huang, W., 2008. Perm for solving circle packing problem. *Computers & Operations Research* 35, 1742–1755.
- Lubachevsky, B.D., Graham, R.L., 1997. Curved hexagonal packings of equal disks in a circle. *Discrete & Computational Geometry* 18, 179–194.
- Mhand, H., Rym, M., 2004. Approximate algorithms for constrained circular cutting problems. *Computers and Operations Research* 31, 675–694.
- Monaci, M., Toth, P., 2006. A set-covering-based heuristic approach for bin-packing problems. *INFORMS Journal on Computing* 18, 71–85.
- Parreño, F., Alvarez-Valdés, R., Oliveira, J.F., Tamarit, J.M., 2010. A hybrid grasp/vnd algorithm for two-and three-dimensional bin packing. *Annals of Operations Research* 179, 203–220.
- Wang, H., Huang, W., Zhang, Q., Xu, D., 2002. An improved algorithm for the packing of unequal circles within a larger containing circle. *European Journal of Operational Research* 141, 440–453.
- Wei, L., Oon, W.C., Zhu, W., Lim, A., 2011. A skyline heuristic for the 2d rectangular packing and strip packing problems. *European Journal of Operational Research* 215, 337–346.
- Zhizhong, Z., Xinguo, Y., Kun, H., Zhanghua, F., 2018. Adaptive tabu search and variable neighborhood descent for packing unequal circles into a square. *Applied Soft Computing* 65, 196–213.