Computers and Operations Research 127 (2021) 105140

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/caor

Adaptive large neighborhood search for solving the circle bin packing problem

Kun He [a], Kevin Tole [a], Fei Ni [a,*], Yong Yuan [a,*], Linyun Liao [a]

[a] School of Computer Science, Huazhong University of Science and Technology, Wuhan 430074, China

ARTICLE INFO

Article history:
Received 15 January 2020
Revised 10 June 2020
Accepted 4 November 2020
Available online 12 November 2020

Keywords:
NP-hard
Circle bin packing problem
Adaptive large neighborhood search
Simulated annealing
Greedy heuristic

ABSTRACT

We address a new packing problem variant known as the 2D circle bin packing problem (2D-CBPP), which involves packing circles into multiple square bins as densely as possible so as to minimize the number of bins used. To this end, we propose an adaptive large neighborhood search (ALNS) algorithm, which uses our Greedy Algorithm with Corner Occupying Action (GACOA) to construct an initial layout. The greedy solution is usually in a local optimum trap, and ALNS enables multiple neighborhood searches that depend on a stochastic annealing schedule to avoid local minimum traps. Specifically, ALNS perturbs the current layout to escape local optima by iteratively reassigning some circles and accepting the new layout with some probability during the search. The acceptance probability is adjusted adaptively using simulated annealing, which fine-tunes the search direction in order to reach the global optimum. We benchmark computational results against GACOA in heterogeneous instances. In all cases, ALNS outperforms GACOA in improving the objective function, and in several cases, the number of bins used for packing is significantly reduced.

1. Introduction

Packing problems form an important class of combinatorial optimization problems that have been well studied under numerous variants (Specht, 2015; Mostafa et al., 2006; Szab et al., 2007; Stracquadanio et al., 2015). It is in general a classic NP-hard problem for which there is no deterministic algorithm to find exact solutions in polynomial time unless $P = NP$, although in some special cases optimal solutions have been reported, e.g., packing rectangular items when $N = 32$ (Korf, 2003). As an important constraint optimization problem, the packing problem has found numerous applications in both academia and industry, including applied mathematics (Garrido et al., 2020), shipping (Peeraya et al., 2012; Túlio et al., 2017), materials manufacturing (Gerhard et al., 2007; Maddaloni et al., 2016), advertisement placement (Freund and Naor, 2004; Dawande et al., 2005), loading problems (Birgin et al., 2005; Leonardo et al., 2012), and more exotic applications such as origami folding (Martin et al., 2010; Byoungkwon et al., 2018).

Packing problems have been well studied since 1832. Farkas (1904) and Szab et al. (2007) investigated the occupancy rate (density) of circles packed in a bounded equilateral triangle bin, and since then tremendous improvements have been made (Graham and Lubachevsky, 1995; Graham and Lubachevsky, 1995). In the past three decades, most research has focused on single container packing. The containers are typically squares, circles, rectangles, or polygons (López and Beasley, 2011), while the items are usually rectangles, circles, triangles, or polygons.

As one of the most classic packing problems, the circle packing problem (CPP) determines how to pack circular items into a container as densely as possible. Researchers have proposed various methods for finding feasible near-optimal packing solutions (Mhand and Rym, 2007; Kun and Wenqi, 2011; Kun and Mohammed, 2017; Zeng et al., 2016). These approaches generally fall into two types: constructive optimization approaches and global optimization approaches.

The construction approach places circular items one by one into a container based on a heuristic that defines the building rules for forming a feasible solution. Most studies that use this approach either (1) fix the container dimensions and pack the items sequentially to satisfy the constraints (Dickinson et al., 2011), or (2) adjust the size of the container using a constructive approach (Zeng et al., 2016). Representative approaches include maximum hole degree

Abbreviations: 2D-CBPP, two-dimensional circle bin packing problem; ALNS, adaptive large neighborhood search; COA, Corner Occupying Action; GACOA, Greedy Algorithm with Corner Occupying Action; SCPP, single circle packing problem; CI, confidence interval; COP, constraint optimization problem.

* Corresponding author.
E-mail addresses: nifei@hust.edu.cn (F. Ni), m201873064@hust.edu.cn (Y. Yuan).

https://doi.org/10.1016/j.cor.2020.105140
0305-0548/

(MHD)-based algorithms (Huang et al., 2003; Wenqi et al., 2006; Huang et al., 2005), among which are two greedy algorithms proposed by Huang et al. (2003): "B.10" places the circular items based on MHD, while "B.15" strengthens the solution via a self-look-ahead search strategy. Another approach, known as the pruned-enriched Rosenbluth method (PERM) (Zhipeng and Wenqi, 2008; Hsu et al., 2003; Huang et al., 2005), is a population control algorithm incorporating the MHD strategy. There are also other heuristics such as best local position (BLP)-based approaches (Mhand and Rym, 2004; Mhand and Rym, 2007; Mhand and Rym, 2008; Akeb and Hifi, 2010), which select the best feasible positions for items by focusing on minimizing the size of the container. However, in our opinion, designing an effective evaluation function with this approach is problematic because its solution is usually incomplete and it is difficult to judge whether the best solution has been found.

On the other hand, the global optimization approach (Castillo et al., 2008) aims to solve the packing problem by improving the solution iteratively based on an initial solution. Unlike other approaches, this approach always generates complete solutions and one can easily weigh the solution quality by devising an evaluation function. However, improving current solutions through this approach is often a difficult task. Based on the above analysis, global optimization techniques are subdivided into two categories. The first category is the quasi-physical quasi-human algorithm (Dingju, 2016; Jingfa et al., 2017; Kun et al., 2018), which aims to encapsulate various physical phenomenon and intuition observed in humans in order to enhance solutions based on problem-oriented heuristics (Kun et al., 2015; Kun et al., 2016). The second category, metaheuristic optimization, is mainly built by defining an evaluation function that employs a tradeoff between randomization and local search, with the goal of directing and remodeling basic heuristics to generate feasible solutions. The metaheuristic searches an estimation in the solution space close to the global optimum. Representative algorithms include a hybrid algorithm (Defu and Ansheng, 2005) that combines simulated annealing and tabu search (Glover, 1989; Glover, 1990). Recently, another hybrid algorithm was proposed by combining tabu search and variable neighborhood descent, and yielded state-of-the-art results (Zhizhong et al., 2018).

In this work, we address a new packing problem variant called the two-dimensional circle bin packing problem (2D-CBPP). Given a collection of circles defined by their radii, we are tasked with packing all items into a minimum number of identical square bins. A packing is called feasible if no circles overlap with each other and no circles cross the bin boundaries. We use the second approach (global optimization) to solve this problem in two alternating phases: the construction phase and the local search phase. The construction phase builds the initial solution while the local search phase stochastically optimizes the solution generated by the construction phase.

Numerous proficient local search algorithms have been proposed for BPP; examples include simulated annealing (Rao and Iyengar, 1994), genetic algorithms (Luo et al., 2017), and tabu search (Viegas et al., 2015). In our case, we use an adaptive local neighbor search metaheuristic that, to our best knowledge, is the first search-based metaheuristic ever adopted for solving CBPP. Our intuition is based on the well-studied two-dimensional (2D) bin packing problem (Andrea et al., 1999; Nikhil et al., 2005), which involves packing a set of rectangular items into a minimum number of identical rectangular bins. Similarly, Osogami and Okano (2003) designed various construction heuristics as their initial solution and developed a local search based on exchanges of items from two randomly selected bins.

This manuscript is an extended version of our previous conference publication (Kun and Mohammed, 2017), in which we first introduced this problem and proposed the Greedy Algorithm with Corner Occupying Action (GACOA) to construct a feasible dense layout (Kun and Mohammed, 2017). In this paper, we describe significant improvements to our algorithm designed to further strengthen the packing quality, and propose an adaptive large neighborhood search (ALNS) algorithm. ALNS first calls GACOA to construct an initial solution. This is followed by a local neighborhood search phase that iteratively perturbs the current solution by randomly selecting any two used bins; this process is slightly similar to that proposed by Osogami and Okano (2003), and follows the idea of He et al. (2018) (considering a case of the Single Circle Packing Problems (SCPPs)) in which each circular item is approximated as a square in order to find unoccupied candidate locations, with the goal of escaping local minima by swapping two randomly selected circles or two equally sized circles. Utilizing our intuition, instead of approximating each circular item as a square, we randomly select equally sized rectangular areas in each bin, and all circles that intersect the two rectangular areas are removed and added to the remaining two assigned sets. Here, we simply use the envelope rectangle of the circle to check its intersection with the area. We then use GACOA to pack the outside circles back into the bin in order to form a complete solution. The complete solution is accepted if the updated layout increases the objective function or if the decrease in the objective function is probabilistically allowed under the current annealing temperature (Kirkpatrick et al., 1983). Note that the objective function is not the number of bins used, but is defined to assist in measuring performance during the effort to reach the global optimum of the new candidate solution. Computational numerical results show that ALNS always outperforms GACOA in improving the objective function, and occasionally ALNS even outputs packing patterns with fewer bins. Additionally, based on our stochastic model solution approach to this problem, we conducted a two-tailed hypothesis test experiment to evaluate the significance of the two algorithms' performance in light of the presented results. From the comparison, we reject the null hypothesis with a 95% Confidence Interval (CI).

We make three main contributions in this work:

1) We design a new form of objective function, in which we specify the number of containers used and the maximum difference between the highest and lowest container densities. The new objective function could help identify the quality of the assignment, particularly in the general case with the same number of bins.
2) We propose a method for local searches on the complete assignment solution. We randomly select two bins and generate equal rectangular areas for each bin. All circles that intersect the rectangular areas are unassigned and the remaining circles form the new partial solution.
3) We modify the conditions for receiving the new partial solution. The previous local neighborhood search algorithm only accepts new partial solutions with larger objective functions. To some extent, this is not conducive to reaching the global optimum. We apply the idea of simulated annealing to this new algorithm such that partial solutions with lower objective function values can also be accepted with a variable possibility.

The remainder of this paper is organized as follows. Section 2 introduces the mathematical constraints for the problem, and Section 3 presents two frameworks used for the development of our algorithm. Section 4 further describes the objective function as well as the experimental setup. Algorithms are computationally evaluated and the results are presented in Section 5. Finally, Section 6 concludes with recommendations for future work.

2. Problem formulation

Given a set of n circles where circle C_i has radius r_i, and n identical square bins with side length L (w.l.g. for any circle $C_i, 2 \cdot r_i \leqslant L$), the two-dimensional circle bin packing problem (2D-CBPP) is to locate the center coordinates of each circle C_i such that all circles are completely contained in the bins and there is no overlap between any two circles. The goal is to minimize the number of bins used, denoted as K $(1 \leqslant K \leqslant n)$.

A feasible solution to CBPP is to partition the circles into sets $\mathscr{S} = \langle S_1, S_2, \ldots, S_K \rangle$ for the bins, with the packing constraints satisfied in each bin. An optimal solution is one in which K, the number of bins used, cannot be reduced any further. A summary of the necessary parameters is given in Table 1.

Assume that the bottom left corner of each bin B_k is placed at $(0, 0)$ in its own 2D geometric Cartesian coordinate system. We formulate the 2D-CBPP as a constraint optimization problem.

$$\sum_{k=1}^{n} I_{ik} = 1, \tag{1}$$

where

$$I_{ik} \in \{0, 1\}, \ i, k \in \{1, \ldots, n\}, \tag{2}$$

which implies that each circle is packed exactly once. Further, if a bin B_k is used, then

$$Y_k = \begin{cases} 1, \text{ if} \sum_{i=1}^{n} I_{ik} > 0, \ i, k \in \{1, \ldots, n\}, \\ 0, \text{ otherwise.} \end{cases} \tag{3}$$

And finally, for circles that are in the same bin, $I_{ik} = I_{jk} = 1$, and $i, j, k \in \{1, \ldots, n\}$, no overlap is allowed, implying that

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geqslant (r_i + r_j) I_{ik} I_{jk}. \tag{4}$$

Specifically, let the circles be ordered by their radii so that $r_1 \geqslant r_2, \ldots \geqslant r_n, r_i \in \mathscr{R}^+$. To ensure that no item crosses the boundary of the bin, we ask that

$$r_i \leqslant x_i \leqslant L - r_i, \ r_i \leqslant y_i \leqslant L - r_i \tag{5}$$

Conditions (1)-(4) along with (5) are the constraints for CBPP.

The overall goal of CBPP is to use as few bins as possible to pack the n circles, which is

$$\min K = \sum_{k=1}^{n} Y_k. \tag{6}$$

3. General search framework

In this section, we introduce two general search optimization frameworks for the constraint optimization problem that we will use to develop our CBPP algorithm. The two frameworks are large neighborhood search (LNS) and adaptive large neighborhood search (ALNS) (Gendreau and Potvin, 2010), the latter of which is a variation of the well-studied simulated annealing process (Kirkpatrick et al., 1983). A particular advantage of ALNS is its capacity to move the iterated solution out of the local optimum.

Table 1
Variable definition.

Variable	Description
n	number of circles
C_i	the i-th circle
r_i	radius of the i-th circle
(x_i, y_i)	center coordinates of C_i
B_k	the k-th bin
L	side length of square bin
I_{ik}	indicates the placement of C_i into the k-th bin
Y_k	indicates the use of the k-th bin
d_{ij}	distance between (x_i, y_i) and (x_j, y_j)

3.1. Large neighborhood search

Large neighborhood search (LNS) is a technique for iteratively solving constraint optimization problems (Gendreau and Potvin, 2010). At each iteration, the goal is to find a more promising candidate solution to the problem, and traverse a better search path through the solution space.

Definition 1. (Constraint optimization problem, COP). A constraint optimization problem $P = \langle n, D^n, \mathscr{C}, f \rangle$is defined by an array of n variables that can take values from a given domain D^n, subject to a set of constraints $\mathscr{C}$. f is the objective function for measuring the performance of an assignment. An assignment is an array of values $\mathbf{a}^n \in D^n$. A constraint $c \in \mathscr{C} : D^n \rightarrow \{0, 1\}$ is a predicate that determines whether an assignment $\mathbf{a}^n \in D^n$ is locally valid. A solution to P is an assignment $\mathbf{a}^n$ that is locally valid for all constraints in $\mathscr{C}$; i.e., $c(\mathbf{a}^n)$ is true for all $c \in \mathscr{C}$. The optimal solution of P is a solution that maximizes the objective function f.

In a nutshell, LNS starts from a non-optimal solution $\mathbf{a}^n$ and iteratively improves the solution until it reaches an optimal or a near-optimal solution. The main ingredient in LNS is an effective algorithm for completing a partial solution.

Definition 2. (Partial solution). A partial solution $\langle k, \mathbf{a}^k, I \rangle$is an assignment $\mathbf{a}^k$ to a subset of k variables (with their indexes I) of a constraint optimization problem P. Completing a partial solution means finding an assignment for the remaining variables $\{a_i | i \notin I\}$such that $\mathbf{a}^n$ is a solution to P.

At each iteration, LNS relaxes and repairs the solution by randomly generating and then completing a partial solution (Alg. 1). If the new assignment $\mathbf{b}^n$ produces a higher objective function value, the previous assignment $\mathbf{a}^n$ will be replaced.

Algorithm 1: Large neighborhood search

```
Input  : A COP P = ⟨n, D^n, C, f⟩,
         number of iteration steps N
Output: A near-optimal solution a^n
1 a^n ← compute_initial_solution(P)
2 for i ← 1 to N do
3 |   I ← generate_partial_solution(a^n);
4 |   b^n ← complete_partial_solution (a^n, I, f);
5 |   if f (b^n) > f (a^n) then
6 |   |   a^n ← b^n;
7 |   end
8 end
```

3.2. Adaptive large neighborhood search

The new solution $\mathbf{b}^n$ obtained using the complete_partial_solution of LNS should be better than the previous solution $\mathbf{a}^n$ in order to be accepted (lines 5–7 in Alg. 1). However, in a sense, this approach actually limits the search for finding a global optimal solution. Thus, we consider an adaptive version of LNS, called ALNS (Gendreau and Potvin, 2010), which is obtained by altering LNS to allow for acceptance of worse solutions. This depends on a prede-

fined stochastic annealing schedule (Gendreau and Potvin, 2010; Kirkpatrick et al., 1983), thus allowing the solution search space to escape local optima. We present a generic description of the ALNS algorithm in Alg. 2, and a more complete explanation of the framework will be presented in Section 4. The key difference is that LNS cannot accept worse solutions, while ALNS can accept worse solutions with probability θ, which facilitates escaping local optima and increases the scope of exploration for globally optimal solutions. For the packing problem, we first set the initial temperature Θ, which is the initial acceptance probability. In each iteration, a certain algorithm (the specific algorithm will be discussed in Section 4) executes to obtain a new perturbed partial solution from the complete solution I. We then execute a heuristic algorithm to complete the partial solution generated during the previous step. Finally, we compare the objective function values of the two solutions. If the new perturbed solution provides better performance, we accept the new solution, otherwise we will accept it with probability θ, which can reduce the possibility of falling into local optimum. After this iteration, we dynamically adjust the acceptance probability of the next iteration, which explains why this search framework is named "adaptive large neighborhood search.".

Algorithm 2: Adaptive large neighborhood search

Input : A COP $P = \langle n, D^n, C, f \rangle$,
number of iteration steps N
Output: A near-optimal solution $\mathbf{a}^n$

$\mathbf{a}^n \leftarrow$ compute_initial_solution(P);
$\Theta \leftarrow$ initial temperature;
$\theta \leftarrow \Theta$;
for $i \leftarrow 1$ **to** N **do**
  $I \leftarrow$ generate_partial_solution ($\mathbf{a}^n$);
  $\mathbf{b}^n \leftarrow$ complete_partial_solution($\mathbf{a}^n, I, f$);
  if *acceptMove*($\mathbf{b}^n, \mathbf{a}^n, \theta$) **then**
    $\mathbf{a}^n \leftarrow \mathbf{b}^n$;
  end
  $\theta = \theta - \Theta/N$;
end

4. ALNS algorithm for CBPP

In this section, we provide an in-depth explanation of the objective function implemented in this work, and then present a comprehensive description of the systematic flow setup of our algorithm. Lastly, we use a flowchart in Fig. 1 to summarize the entire process of our algorithm..

4.1. Domain and objective function

For CBPP as a constraint optimization problem (COP), we define its domain D^n as follows. The assignment variables for each circle C_i include $\langle x_i, y_i, I_{i1}, \ldots, I_{in} \rangle$. Because each circle C_i can be placed only in a single bin, as constrained by an indicator function, we simplify the notation to $a_i = \langle x_i, y_i, b_i \rangle$, where b_i represents the ID of the bin that C_i is assigned to; thus, $I_{ib_i} = 1$. The corresponding domain $D = \mathbb{R}^2 \times \{1, \ldots, n\}$ defines all possible assignments of a circle in the bin-coordinate space, and places an emphasis on the circles rather than the containers; each of the components where $i \in \{1, \ldots, n\}$ is a three-tuple denoted as $a_i = \langle x_i, y_i, b_i \rangle$. Thus, when referring to a solution to the optimization problem P, we write $\mathbf{a}^n = \{a_1, \ldots, a_n\}$, and $D^n = D \times D \times \ldots \times D$ corresponds to the domain for the n tuple variables. Therefore, $\mathbf{a}^n$ denotes a possible packing.

Let L^2 be the area of a bin, and let the density of a bin B_k of a packing be defined as

$$d_k(\mathbf{a}^n) = \frac{1}{L^2} \sum_{i \in \{j | C_j \in S_k\}} \pi r_i^2, \tag{7}$$

where S_k is the set of items assigned to the k-th bin B_k. Let K be the number of bins used for a solution $\mathbf{a}^n$, so

$$K = \sum_{k=1}^{n} Y_k. \tag{8}$$

$$\begin{aligned} \text{Let } d_{\min} &= \min\{d_k(\mathbf{a}^n) | 1 \leqslant k \leqslant K\}, \\ d_{\max} &= \max\{d_k(\mathbf{a}^n) | 1 \leqslant k \leqslant K\}. \end{aligned}$$

Here, d_{min} denotes the density of the sparsest bin and d_{max} the density of the densest bin. We define a useful objective function, which will form part of our algorithm as

$$f(\mathbf{a}^n) = -K + d_{max} - d_{min}. \tag{9}$$

The larger the value of $f(\cdot)$, the better the packing, because an increase in $f(\cdot)$ corresponds to a denser packing as circles move out of lower density bins. In order to clarify the process for taking a complete candidate solution $\mathbf{a}^n$ for P to a partial solution.

Note that $0 \leqslant d_{max} - d_{min} \leq 1$. This term, which is used for regularization, implies that using fewer bins is preferable, and that a difference in the number of bins is sufficient to compare two candidate solutions. When different solutions all use the same number of bins, we focus on the fullest bin and emptiest bin in each candidate solution. The denser the fullest bin is, the lower the amount of wasted space. The sparser the emptiest bin is, the more concentrated the remaining still-reserved space is, which means it will be easier to assign subsequent circles. Therefore, the difference in density between the fullest bin and the emptiest bin determines the quality of each candidate solution.

4.2. Construct initial solution

An initial solution can be rapidly constructed by our greedy algorithm GACOA (Alg. 3).

$$\text{compute\_initial\_solution}(P) = GACOA(L, \{C_i | 1 \leqslant i \leqslant n\}) \tag{10}$$

For each circle, GACOA computes a set of candidate positions by greedily moving on to the next bin if a circle cannot be packed in any of the previous bins. In particular, each circle is packed according to the following criteria.

Definition 3. (Candidate packing position). A candidate-packing position of a circle in a bin is any position that places the circle tangent to a) any two packed items, b) a packed item and the border of the bin, or c) two perpendicular sides of the bin (i.e., a corner).

Definition 4. (Feasible packing position). A packing position of a circle in a bin is feasible if it does not violate any constraints; i.e., circles must not overlap and must be fully contained in a bin. (See Eqs. (1)–(5) for detailed constraints).

Definition 5. (Quality of packing position). The distance between the feasible packing position and the border of the bin is given by

$$q(x, y) = \{\min(d_x, d_y), \max(d_x, d_y)\}. \tag{11}$$

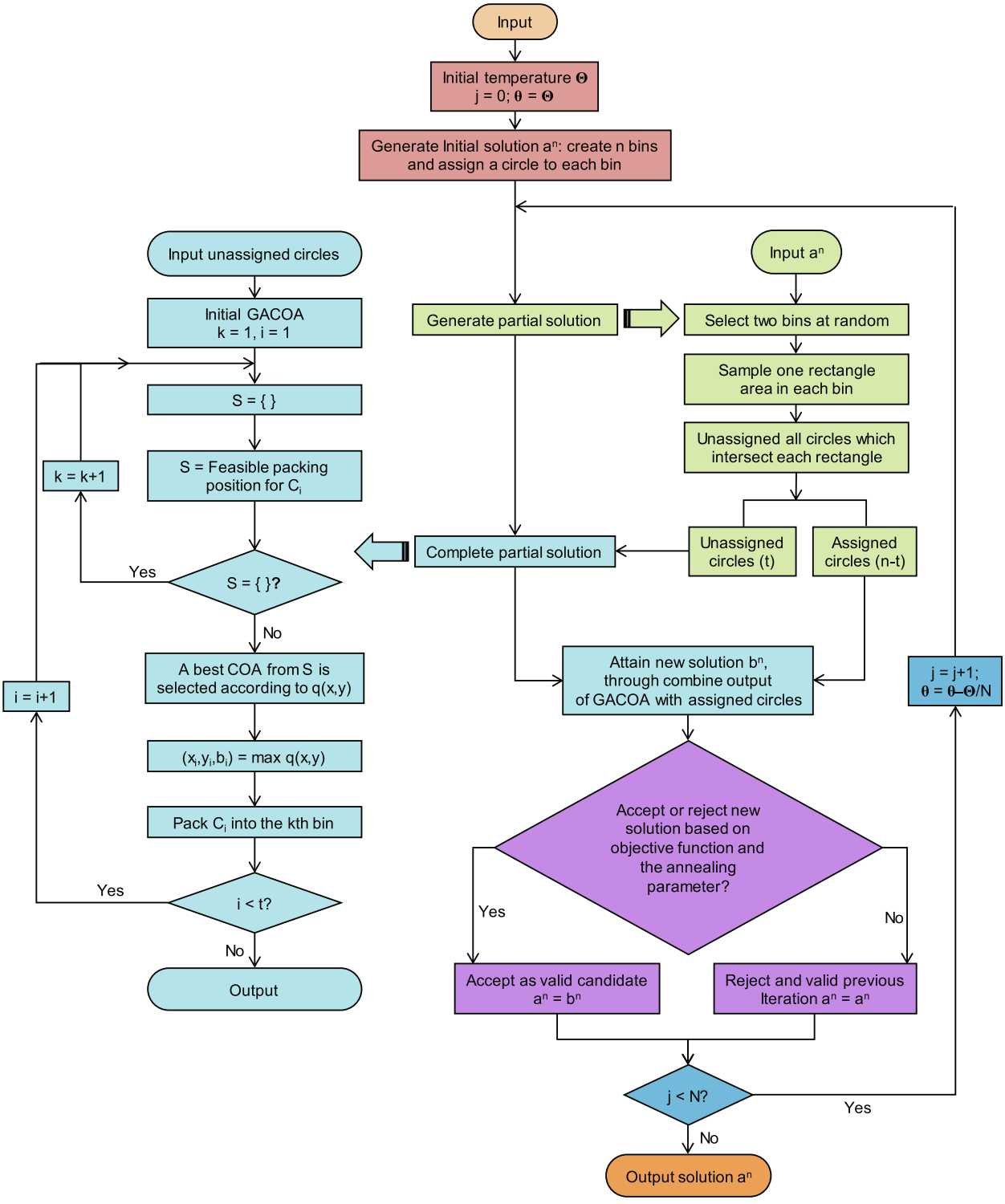

Fig. 1. Parameter setup flow of ALNS.

where d_x (resp. d_y) is the distance between the center of the circle and the closer side of the bin in the horizontal (resp. vertical) direction. For a circle in the current target bin, all feasible positions in the bin are sorted in the ascending order of $q(x, y)$. A smaller $q(x, y)$ means the packing position is closer to the edge of the bin. We recommend that the circle be placed closer to the edge of the bin, which will allow the free space to be more concentrated and facilitate the placement of subsequent circles.

Algorithm 3: GACOA-Algorithm

Input: Bin side length L, a set of n circles $\{C_i | 1 \le i \le n\}$ with radii $r_1, \ldots, r_n$ $(r_i \ge r_{i+1})$
Result: For each circle C_i, find a bin B_k, and place the circle center at (x_i, y_i)

```
1  K ← 0
2  for i ← 1 to n do
3      for k ← 1 to n do
4          S_k ← ∅;
5          while true do
6              if k > K then
7                  K = k
8              end
9              S_k ← Feasible packing positions for C_i;
10             if S_k ≠ ∅ then
11                 break
12             end
13             k ← k + 1
14         end
15         A best packing position from S_k is selected
             according q(x, y);
16         (x_i, y_i) ← arg max_{(x,y)∈S} q(x, y);
17         Execute this packing position to pack circle
             C_i into bin B_k;
18     end
19 end
```

An action that places a circle onto one of its candidate packing positions is known as a Corner Occupying Action (COA). GACOA works by packing circles one by one in descending order of their radii size. Target bins are considered for each circle in ascending order of the bin index k. For bin B_k, a feasible candidate packing position with the highest quality is selected; i.e., a feasible assignment that maximizes $q(x, y)$ will be executed. Thus, the best feasible COA that favors positions closer to the border of the bin is selected. If there is no feasible assignment in bin B_k, we will attempt to pack in the subsequent bin B_{k+1}. The pseudocode is provided in Alg. 3.

As Alg. 3 shows, partial solutions are generated from a complete solution by selecting two bins at random and performing perturbations. We randomly select a rectangular area of equal size in each bin; all circles that intersect the two rectangular areas will be taken out and added to the remain2assign set, and the complete solution then becomes a partial solution. The unassigned circles will be reassigned based on the partial solution. This is equivalent to perturbing a complete solution. Let function random_ints(m, M) return m distinct integers randomly selected from set M. Let function random_real (R) return a random real number $0 < r \leqslant R$.

Alg. 5 randomly selects a circle in a non-empty bin, and then randomly generates a rectangular area with the circle center as its center. This guarantees that at least one circle will intersect the generated rectangular area (Here, we simply use the envelope rectangle of the circle to check its intersection with the area). In most cases, more than one circle will intersect the rectangular area and will be unassigned at each iteration. As line 6 shows, we randomly select one circle from B_k by using the function random_ints (m, M) again. We select two bins and generate a rectangular area for each bin. Only one bin can be sampled at a time, but when the partial solution and unassigned circle set are placed in the future, in the worst case, it will be put back as it was to obtain the same complete solution as before. The worst instance is that the previous partial solution did not leave sufficient free space to allocate the unassigned circles except for the generated area. If there is only one bin and one rectangular area, the unassigned circles are very likely to be placed back into the previous generated rectangular area during the iteration, which means that this iteration process has no effect and does not facilitate escaping the local optimum. Thus, two bins are selected at each iteration such that the unassigned circles have more free space in which to be allocated. Even in the worst case, the algorithm will attempt to exchange the circles in the two rectangular areas, which ensures that there will be some disturbance per iteration. Of course, an alternative method is to sample only one bin and generate two rectangular areas for the bin, but two rectangular areas in different bins can increase the randomness.

Algorithm 4: Generate partial solution

Input: A (complete) solution $\mathbf{a}^n$
Output: A partial solution given by the indexes to keep I

```
//Select two bins randomly
1 (k_1, k_2) ← random_ints(2, {1, ..., K)})
//Select a rectangular area in the first bin
2 rect_1 ← sample_rects(B_{k_1})
// Select a rectangular area in the second bin
3 rect_2 ← sample_rects(B_{k_2})
4 remain2assign ← ⋃_{j∈{1,2}} {i | ⟨x_i, y_i, k_j⟩ ∈ a^n ⋀ I_{ik_j} = 1 ⋀ intersects(C_i, rect_j) == True}
5 I ← I/remain2assign
```

Algorithm 5: sample_rects

Input : bin B_k; side length of bin L
Output: rectangle area Rects
// Each rectangle is represented as a bottom-left point and a top-right point
1 $w \leftarrow$ random_real(L) // the width of rectangle area is w
2 $h \leftarrow$ random_real(L) // the height of rectangle area is h
3 let $l_x \leftarrow 0, l_y \leftarrow 0$ // where (l_x, l_y) is the coordinate of bottom-left point
4 let $b_j \leftarrow k$ // k is the id of bin B_k
5 **if** $(!B_k.empty())$ **then**
6 $\quad i \leftarrow$ random_ints$(1, \{ j \mid \langle x_j, y_j, b_j \rangle \in \mathbf{a}^n \})$
7 **end**
// select a circle randomly from B_k
8 $l_x \leftarrow C_i.x - 0.5w$
9 $l_y \leftarrow C_i.y - 0.5h$ $Rects =$ make_pair(point(l_x, l_y),point$(l_x + w, l_y + h)$)

Alg. 6 can determine whether a circle C_i intersects the selected rectangle area. If the circle C_i intersects the selected rectangle area, it will be added to the unassigned circle set in Alg. 5 line 4.

Algorithm 6: intersects

Input : circle C_i, $rect_k$ with coordinate tuple $<l_x, l_y, l_x + w, l_y + h>$
Output: true or false
// returns if C_i intersects the rectangle
1 **if** $C_i.x - C_i.r \geq l_x + w$ **then**
2 $\quad$ return false // non-intersect
3 **end**
4 **if** $C_i.y - C_i.r \geq l_y + h$ **then**
5 $\quad$ return false // non-intersect
6 **end**
7 **if** $C_i.x + C_i.r \leq l_x$ **then**
8 $\quad$ return false // non-intersect
9 **end**
10 **if** $C_i.y + C_i.r \leq l_y$ **then**
11 $\quad$ return false // non-intersect
12 **end**
13 return true // intersect

4.3. Complete a partial solution

Completing a partial solution is performed efficiently by the GACOA algorithm (Alg. 3), restricted to the bins B_{k1}, B_{k2} from which circles were unassigned in the previous step.

$$\text{complete\_partial\_solution}(\mathbf{a}^n, I) = \text{GACOA}(L, \{C_i | i \in remain2assign\}), \tag{12}$$

where I is the partial solution generated by generate_partial_solution(). GACOA is used to complete the partial solution.

4.4. Acceptance metric

A solution is accepted if it increases the value of the objective function or if the decrease in the objective function is probabilistically allowed given the current annealing temperature θ. This is the well-known simulated annealing move acceptance criteria (Kirkpatrick et al., 1983),

$$\text{acceptMove}(\mathbf{b}^n, \mathbf{a}^n, \theta) = f(\mathbf{b}^n) > f(\mathbf{a}^n) \vee random\_real(1) \leqslant e^{\frac{f(\mathbf{b}^n) - f(\mathbf{a}^n)}{\theta}} \tag{13}$$

4.5. ALNS for CBPP

The complete algorithm for solving CBPP requires various steps described above. The ALNS procedure is started using the initial solution along with temperature Θ and number of iterations N. The initial solution is then broken using Alg. 3 and re-completed using the new solution filled by GACOA. This procedure outputs a new candidate solution, which is then either accepted or rejected based on the acceptance metric with simulated annealing. At this point, if the iteration limit has not been reached, the process restarts using the new solution as an input.

An illustration of the parameter setup flow of the ALNS algorithm is shown in Fig. 1.

5. Computational experiments

To evaluate the competency of the proposed approach, we implemented the ALNS for CBPP using the Visual C++ programming language. All results were generated by setting $N = 2 \times 10^6$ (Alg. 2), and were obtained using a computer equipped with an Intel Core i7-8550U CPU @ 1.80 GHz.

We generated benchmarks based on two groups of instances downloaded fromwww.packomonia.com for single circle packing problems (SCPPs): $r\_i = i$ for wide variation instances (i.e., the circle sizes vary widely), and $r\_i = \sqrt{i}$ for smaller variation instances. On the Packomania website, n_0 (the number of different circle sizes for SCPP) ranged from 8 through 20, and sets of benchmarks were generated as follows: For each square bin, the best solution found in Eckardi (2018) for the SCPP was used to fix the bin size L as the best known record *L_bestknownrecord*, and we generated two sets of benchmarks called "fixed" and "random." Let $S = \{C\_i | 1 \leqslant i \leqslant n\}$ be the set of circles packed in the current best solution for SCPP. The fixed set of CBPP benchmarks contains exactly 5 copies of each size of circle packed for SCPP. The random set of CBPP benchmarks contains a random number $(2 \leqslant r \leq 5)$ of copies of each circle (i.e., the number of copies of each size of circle varies across the same benchmark) packed for SCPP.

In the following tables, we list 52×2 generated instances from the two groups of instances. We compared the number of bins and the objective value in the best solution obtained by our search algorithm ALNS against the solution obtained from our constructive algorithm GACOA. For each original number of circles (n_0) and actual number of replicated circles in the CBPP benchmark (n), there are two rows showing the bin densities generated by the ALNS (A) and GACOA (G) algorithms. The last two columns contain the final value of the objective function obtained with each algorithm and the relative improvement of ALNS over GACOA. Additionally, for the fixed benchmarks (Table 2 and Table 4), we insert bin_0 in column 4, which denotes the best results indexed from state of the art inwww.packomonia.com for reference. Fig. 2 details the typical behavior of ALNS and GACOA during the comparison conducted with the two benchmark instances. The objective function (f) is represented by the y-axis and the number of

Table 2
Experimental results on the fixed benchmarks with square bins when $r_i = i$. The average improvement is 0.24.

n_0	n	alg.	Bin_o	bin 1	bin 2	bin 3	bin 4	bin 5	bin 6	f	$f_A - f_G$
8	40	A	0.76	0.83	0.80	0.76	0.74	0.65	–	−4.82	0.06
		G		0.83	0.74	0.71	0.76	0.73	–	−4.88	
9	45	A	0.79	0.81	0.81	0.80	0.76	0.75	–	−4.94	0.51
		G		0.76	0.72	0.75	0.75	0.72	0.21	−5.45	
10	50	A	0.81	0.83	0.82	0.79	0.78	0.76	0.08	−5.25	0.09
		G		0.84	0.78	0.80	0.75	0.72	0.18	−5.34	
11	55	A	0.80	0.84	0.82	0.80	0.78	0.77	–	−4.93	0.43
		G		0.83	0.78	0.77	0.75	0.69	0.19	−5.36	
12	60	A	0.81	0.84	0.81	0.80	0.80	0.80	–	−4.96	0.43
		G		0.84	0.79	0.78	0.72	0.69	0.23	−5.39	
13	65	A	0.82	0.83	0.81	0.81	0.81	0.80	0.05	−5.22	0.26
		G		0.83	0.79	0.76	0.72	0.69	0.31	−5.48	
14	70	A	0.83	0.83	0.82	0.82	0.81	0.79	0.09	−5.26	0.22
		G		0.84	0.82	0.77	0.71	0.71	0.32	−5.48	
15	75	A	0.83	0.83	0.83	0.82	0.81	0.81	0.05	−5.22	0.25
		G		0.85	0.81	0.76	0.70	0.70	0.32	−5.47	
16	80	A	0.84	0.84	0.84	0.82	0.81	0.79	0.08	−5.24	0.17
		G		0.85	0.82	0.81	0.73	0.71	0.26	−5.41	
17	85	A	0.85	0.85	0.83	0.82	0.81	0.80	0.11	−5.26	0.15
		G		0.85	0.82	0.79	0.75	0.75	0.26	−5.41	
18	90	A	0.85	0.85	0.83	0.82	0.81	0.81	0.11	−5.26	0.18
		G		0.85	0.80	0.80	0.79	0.71	0.29	−5.44	
19	95	A	0.85	0.85	0.83	0.82	0.81	0.81	0.12	−5.27	0.21
		G		0.84	0.81	0.80	0.77	0.69	0.32	−5.48	
20	100	A	0.85	0.84	0.83	0.82	0.81	0.80	0.12	−5.28	0.17
		G		0.86	0.81	0.78	0.77	0.71	0.31	−5.45	

Table 3
Experimental results on the random benchmarks for $r_i = i$. The average improvement is 0.18.

n_0	n	alg.	bin 1	bin 2	bin 3	bin 4	bin 5	f	$f_A - f_G$
8	34	A	0.83	0.80	0.74	0.73	0.24	−4.41	0.09
		G	0.83	0.74	0.74	0.70	0.33	−4.50	
9	30	A	0.81	0.80	0.77	0.44	–	−3.63	0.18
		G	0.78	0.75	0.71	0.59	–	−3.81	
10	37	A	0.81	0.80	0.79	0.76	–	−3.95	0.40
		G	0.83	0.78	0.71	0.67	0.18	−4.35	
11	38	A	0.83	0.81	0.79	–	–	−2.96	0.25
		G	0.82	0.81	0.78	0.03	–	−3.21	
12	43	A	0.83	0.81	0.80	0.29	–	−3.46	0.12
		G	0.82	0.79	0.72	0.40	–	−3.58	
13	44	A	0.83	0.81	0.78	0.43	–	−3.60	0.17
		G	0.82	0.75	0.69	0.59	–	−3.77	
14	48	A	0.84	0.82	0.80	0.40	–	−3.56	0.15
		G	0.82	0.77	0.73	0.53	–	−3.71	
15	53	A	0.85	0.83	0.81	0.32	–	−3.47	0.11
		G	0.85	0.81	0.72	0.43	–	−3.58	
16	55	A	0.85	0.82	0.81	0.46	–	−3.61	0.14
		G	0.84	0.78	0.73	0.59	–	−3.75	
17	53	A	0.84	0.82	0.80	0.32	–	−3.48	0.15
		G	0.82	0.79	0.73	0.45	–	−3.63	
18	72	A	0.84	0.83	0.83	0.72	–	−3.88	0.36
		G	0.84	0.81	0.78	0.70	0.08	−4.24	
19	58	A	0.83	0.83	0.81	0.07	–	−3.24	0.12
		G	0.83	0.76	0.76	0.19	–	−3.36	
20	78	A	0.84	0.84	0.81	0.59	–	−3.75	0.08
		G	0.86	0.82	0.80	0.69	–	−3.83	

circles is represented by the x-axis. We can observe that the packing occupancy rate of ALNS is higher than that of GACOA. An in-depth analysis is provided in Section 5.1 and Section 5.2.

5.1. Comparison on $r = i$

We executed ALNS and GACOA on the benchmark instances of $r = i$. The number of different circle sizes n_0 ranged from 8 to 20 for both fixed and random square settings. Table 2 shows the computational results achieved with a fixed number of circle copies (5) for each n_0. The second column in Table 3 displays the random number of circle copies, ranging from $2 \leqslant r \leq 5$ of the corresponding n_0. From Table 2 we can observe that when $n = 45, 55$, or 60 on the fixed benchmark, ALNS utilizes five bins to pack the circles, while GACOA uses six bins. Fig. 3 illustrates the packing layout for $n = 45$. For the random instances in Table 3 when $n = 37$ or 72, ALNS also minimizes the number of bins by packing circles in four bins while GACOA packs in five bins. Fig. 4 illustrates the packing layout for $n = 72$. In summary, the results of all the instances show that ALNS returns feasible results, and provides an overall

Table 4
Experimental results on the fixed benchmarks when $r_i = \sqrt{i}$. The average improvement is 0.10.

n_0	n	alg.	bin_0	bin 1	bin 2	bin 3	bin 4	f	$f_A - f_G$
8	40	A	0.77	0.84	0.81	0.78	0.30	−3.46	0.22
		G		0.80	0.76	0.70	0.47	−3.68	
9	45	A	0.78	0.82	0.82	0.79	0.33	−3.5	0.19
		G		0.82	0.73	0.70	0.51	−3.69	
10	50	A	0.81	0.83	0.82	0.79	0.38	−3.55	0.15
		G		0.84	0.76	0.69	0.54	−3.70	
11	55	A	0.82	0.83	0.82	0.81	0.39	−3.57	0.08
		G		0.83	0.78	0.77	0.48	−3.65	
12	60	A	0.82	0.83	0.83	0.81	0.39	−3.56	0.07
		G		0.83	0.80	0.77	0.46	−3.63	
13	65	A	0.83	0.84	0.84	0.79	0.39	−3.55	0.10
		G		0.83	0.82	0.72	0.49	−3.65	
14	70	A	0.82	0.85	0.82	0.80	0.36	−3.52	0.07
		G		0.84	0.79	0.78	0.43	−3.59	
15	75	A	0.83	0.84	0.83	0.82	0.37	−3.53	0.08
		G		0.85	0.80	0.75	0.46	−3.61	
16	80	A	0.84	0.84	0.83	0.82	0.39	−3.56	0.09
		G		0.85	0.81	0.73	0.49	−3.64	
17	85	A	0.84	0.84	0.83	0.82	0.39	−3.55	0.11
		G		0.83	0.79	0.77	0.50	−3.67	
18	90	A	0.84	0.85	0.83	0.81	0.40	−3.56	0.07
		G		0.85	0.80	0.76	0.48	−3.63	
19	95	A	0.85	0.85	0.83	0.81	0.40	−3.55	0.04
		G		0.85	0.82	0.77	0.45	−3.59	
20	100	A	0.85	0.86	0.83	0.81	0.40	−3.55	0.06
		G		0.87	0.80	0.77	0.47	−3.60	

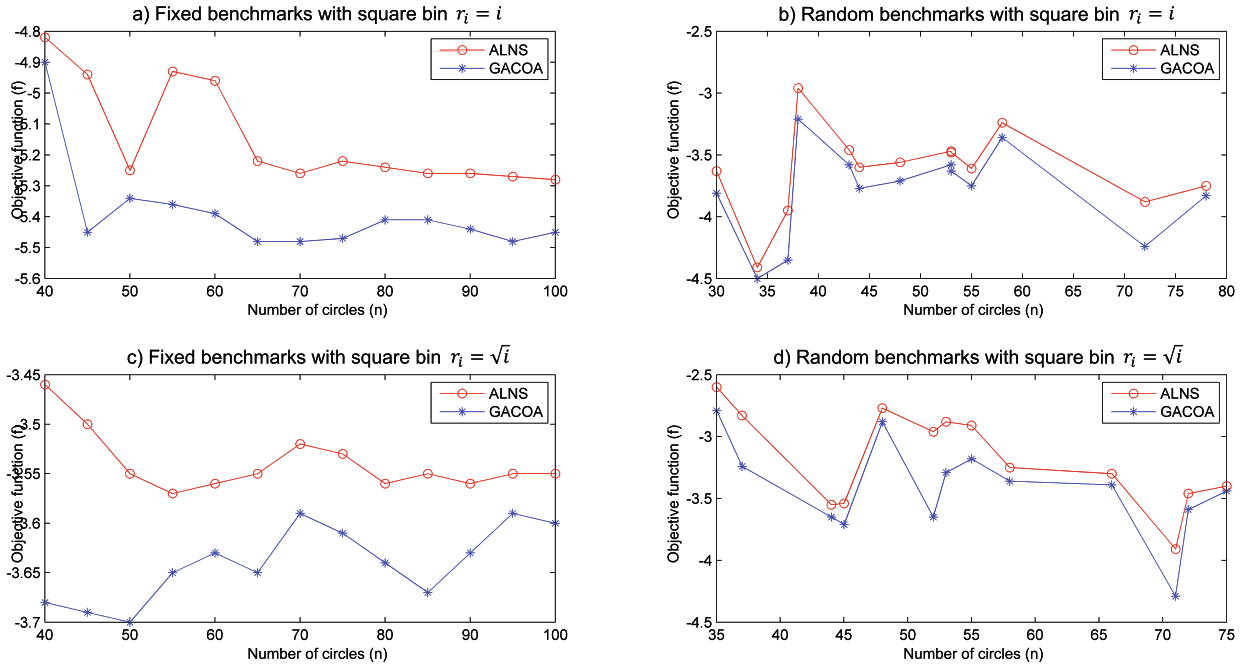

Fig. 2. ALNS vs. GACOA.

average improvement of 19% for the fixed benchmarks and 12% for the random benchmarks when compared to GACOA.

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

Similarly, we executed ALNS and GACOA on the benchmark instances of $r_i = \sqrt{i}$, which generates smaller variations on the radii. Table 4 contains fixed instances while Table 5 contains random instances. The instances also range n_0 from 8 to 20. As an illustration, we select $n_0 = 8$ and $n = 40$ from Table 4 and show the layout in Fig. 5-6. The occupying rate for the first three bins is higher for ALNS than for GACOA, showing that most circle items have maximally occupied each bin area. The density of the packing in the fourth (last) bin is lower for ALNS than for GACOA, indicating that ALNS packed fewer circles in the last bin. On the random benchmarks in Table 5 for n_0 values of 8, 12, 13, 14, and 15, we also observe that ALNS uses one less bin when compared to GACOA. The average improvement provided by ALNS over GACOA is 22% for the fixed instances and 23% for the random instances.

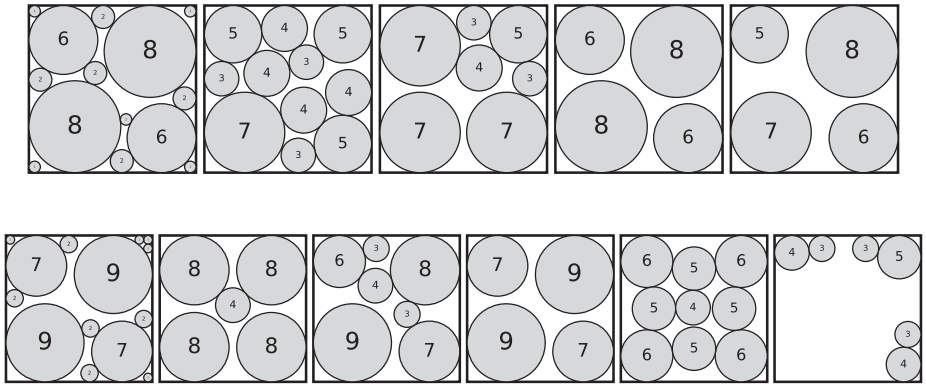

Fig. 3. Packing layouts generated by ALNS (top) and GACOA (bottom) when $N_0 = 9$ and n = 45 (bin densities shown in Table 2).

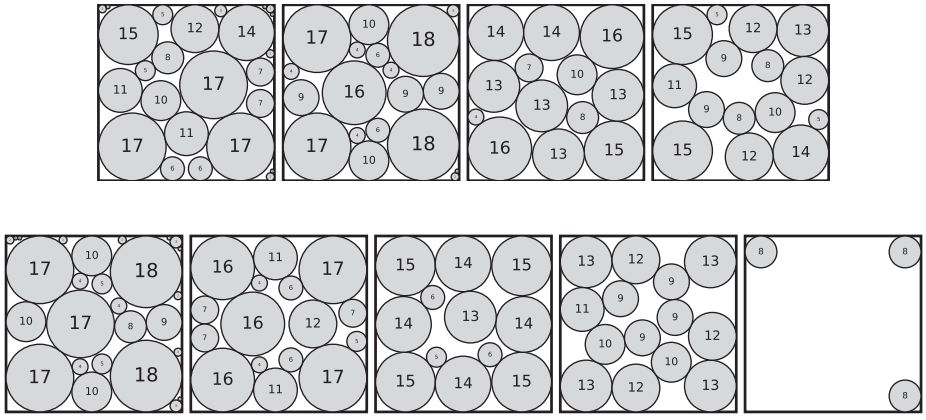

Fig. 4. Packing layouts generated by ALNS (top) and GACOA (bottom) when n_0 = 18 and n = 72 (bin densities shown in Table 3).

Table 5
Experimental results on the random benchmarks for $r_i = \sqrt{i}$. The average improvement is 0.24.

n_0	n	alg.	bin 1	bin 2	bin 3	bin 4	bin 5	f	$f_A - f_G$
8	37	A	0.84	0.83	0.67	–	–	−2.83	0.41
		G	0.80	0.75	0.74	0.04	–	−3.24	
9	35	A	0.82	0.81	0.42	–	–	−2.6	0.19
		G	0.77	0.72	0.56	–	–	−2.79	
10	45	A	0.83	0.82	0.81	0.37	–	−3.54	0.17
		G	0.83	0.76	0.69	0.54	–	−3.71	
11	44	A	0.84	0.81	0.81	0.39	–	−3.55	0.10
		G	0.83	0.78	0.77	0.48	–	−3.65	
12	52	A	0.84	0.82	0.80	–	–	−2.96	0.69
		G	0.83	0.78	0.77	0.48	–	−3.65	
13	71	A	0.85	0.83	0.81	0.76	–	−3.91	0.38
		G	0.83	0.83	0.75	0.70	0.12	−4.29	
14	55	A	0.84	0.83	0.75	–	–	−2.91	0.27
		G	0.85	0.80	0.76	0.03	–	−3.18	
15	53	A	0.83	0.83	0.71	–	–	−2.88	0.41
		G	0.84	0.74	0.65	0.13	–	−3.29	
16	48	A	0.83	0.82	0.60	–	–	−2.77	0.11
		G	0.81	0.75	0.69	–	–	−2.88	
17	72	A	0.84	0.82	0.81	0.30	–	−3.46	0.13
		G	0.82	0.77	0.77	0.41	–	−3.59	
18	66	A	0.82	0.81	0.80	0.12	–	−3.30	0.09
		G	0.82	0.78	0.75	0.21	–	−3.39	
19	75	A	0.84	0.82	0.81	0.24	–	−3.40	0.04
		G	0.84	0.81	0.78	0.28	–	−3.44	
20	58	A	0.81	0.81	0.79	0.06	–	−3.25	0.10
		G	0.78	0.78	0.76	0.14	–	−3.36	

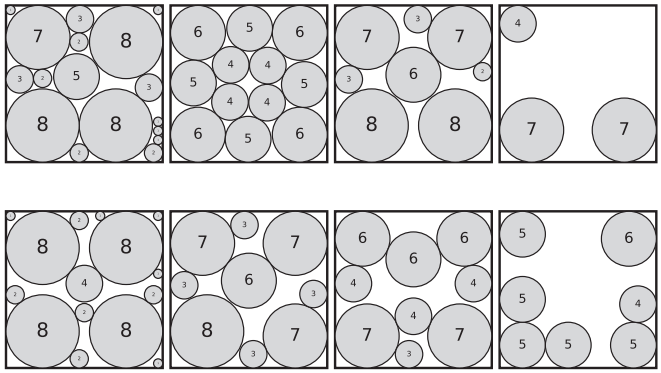

Fig. 5. Packing layouts generated by ALNS (top) and GACOA (bottom) when n_0 = 8 and n = 40 (bin densities shown in Table 4).

5.3. Further discussion

From the above computational results, we clearly observe that our ALNS algorithm outperforms the GACOA algorithm in almost all instances. To assess the significance difference between the two implemented algorithms, we used SPSS statistics software to conduct a two-tailed hypothesis statistical test on $H_O : \mu_A = \mu_G$ verses $H_1 : \mu_A \neq \mu_G$. We used the commonly used level of significance $\alpha = 0.05$, as our threshold value and compared it against the p-values obtained for each table, as shown in Table 7. We can notice that, for all the tables, our p-values $< \alpha$ (0.05); therefore, we can deduce that $H_O : \mu_A$ - $\mu_G \neq 0$, meaning that the differences between the solutions generated by the algorithms statistically significant, attaining a 95% CI. We reject the null hypothesis H_O and claim that there is a significant difference between ALNS metaheuristic and GACOA algorithm.

Table 7
Statistical Test.

Group	Table	p Value
$r_i = i$	Table 2	0.000015
	Table 3	0.000031
$r_i = \sqrt{i}$	Table 4	0.012382
	Table 5	0.000593

The runtimes for ALNS at each benchmark are shown in Table 6 (column t in seconds). We see that ALNS computes the instances efficiently in less than 300 s for 100 circles. By comparison, as a greedy algorithm, GACOA completes the calculation in microseconds on any of these benchmarks. Such results indicate the high efficiency of the ALNS algorithm.

In summary, for each generated instance, we compared the number of bins and objective value in the best solutions obtained by ALNS and GACOA. The results clearly show that, compared to GACOA, ALNS consistently improves the objective function value over all sets of benchmarks; moreover, ALNS was even able to

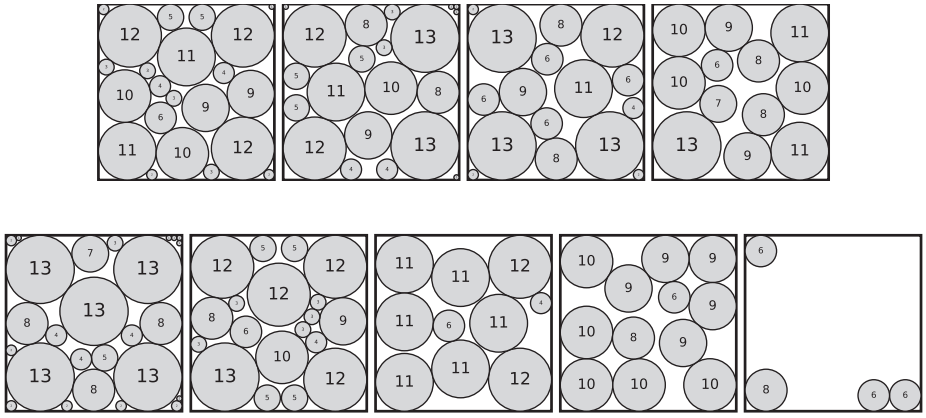

Fig. 6. Packing layouts generated by ALNS (top) and GACOA (bottom) when n_0 = 13 and n = 71 (bin densities shown in Table 5).

Table 6
Runtimes for ALNS execution in all benchmarks.

	$r_i = i$				$r_i = \sqrt{i}$			
	fixed		random		fixed		random	
n_0	n	t	n	t	n	t	n	t
8	40	82	34	59	40	58	37	42
9	45	72	30	63	45	91	35	69
10	50	74	37	69	50	80	45	78
11	55	91	38	160	55	96	44	82
12	60	103	43	120	60	110	52	81
13	65	116	44	127	65	120	71	155
14	70	135	48	148	70	143	55	132
15	75	154	53	177	75	171	53	201
16	80	179	55	189	80	186	48	161
17	85	204	53	179	85	212	72	177
18	90	222	72	339	90	233	66	198
19	95	247	58	209	95	256	75	264
20	100	279	78	366	100	282	58	242

reduce the number of bins used in some benchmarks. The results show that our objective function effectively guides the ALNS algorithm to search for dense packing and reduces the number of bins used.

6. Conclusions

We address the two-dimensional circle bin packing problem (2D-CBPP), a new type of packing problem, and propose an adaptive local search algorithm for solving this NP-hard problem. The algorithm adopts a simulated annealing search onto our greedy constructive algorithm. The initial solution is built by the greedy algorithm. Subsequently, during the search, we generate a partial solution by randomly selecting rectangular areas in two bins and removing the circles that intersect the areas. We implement our greedy algorithm for completing partial solutions during the search. To facilitate the search, we design a new form of objective function, and specify the number of containers to be used and the maximum difference between the highest and lowest container densities. A new solution is conditionally accepted by simulated annealing, completing one iteration of the search.

Despite the improvements in this work, it is highly noted that the proposed problem is indeed challenging for combinatorial optimization heuristics, and future research is needed to ensure that better solutions and high quality benchmarks can be obtained. Implementing an adaptive local neighborhood search appears to be an attractive metaheuristic to adopt. We plan to apply the idea to other circle bin packing problems, and extend our approach to address the three-dimensional circle bin packing problem (3D-CBPP), which is more challenging and has many applications that deserve proper attention.

CRediT authorship contribution statement

Kun He: Conceptualization, Funding acquisition, Methodology, Project administration, Resources, Software, Supervision, Validation, Writing - review & editing. **Kevin Tole:** Conceptualization, Data curation, Formal analysis, Methodology, Investigation, Project administration, Resources, Software, Validation, Visualization, Writing - original draft, Writing - review & editing. **Fei Ni:** Visualization, Project administration, Writing - original draft, Writing - review & editing. **Yong Yuan:** Validation, Visualization, Project administration, Formal analysis, Writing - review & editing. **Linyun Liao:** Validation, Visualization, Project administration, Formal analysis, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. U1836204, U1936108) and the Fundamental Research Funds for the Central Universities (2019kfyXKJC021).

References

Akeb, H., Hifi, M., 2010. A hybrid beam search looking-ahead algorithm for the circular packing problem. J. Combinatorial Optim. 20, 101–130.
Andrea, L., Silvano, M., Daniele, V., 1999. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. J. Comput. 11, 345–357.
Birgin, E.G., Martine, J.M.z., Ronco, D.P., 2005. Optimizing the packing of cylinders into a rectangular container: a nonlinear approach. Eur. J. Oper. Res. 160, 19–33.
Byoungkwon, A., Shuhei, M., Aaron, O., Michael, T.T., Martin, L.D., Erik, D.D., Robert, J.W., Daniela, R., 2018. An end-to-end approach to self-folding origami structures. IEEE Trans. Rob. 34, 1409–1424.
Castillo, I., Kampas, F.J., Pinter, J.D., 2008. Solving circle packing problems by global optimization: numerical results and industrial applications. Eur. J. Oper. Res. 191, 786–802.
Dawande, M., Kumar, S., Sriskandarajah, C., 2005. A note on the minspace problem. J. Sched. 8, 97–106.
Defu, Z., Ansheng, D., 2005. An effective hybrid algorithm for the problem of packing circles into a larger containing circle. Comput. Oper. Res. 32, 1941–1951.
Dickinson, W., Guillot, D., Keaton, A., Xhumari, S., 2011. Optimal packings of up to five equal circles on a square flat torus. Contrib. Algebra Geometry 52, 315–333.
Dingju, Z., 2016. Quasi-human seniority-order algorithm for unequal circles packing. Chaos Solitons Fractals 89, 506–517.
Eckardi, S., 2018. Packomania website 2018.www.packomania.com..
Farkas, B., 1904. Wolfgangi bolyai de bolya: Tentamen iuventutem studiosam in elementa math-eseos purae elementaris ac sublimioris methodo intuitiva evidentiaque huic propria introducendi,cum appendice triplici. In latin. marosvasarhelyini, second ed. 2 (1832-33), 119–122..
Freund, A., Naor, S., Joseph, 2004. Approximating the advertisement placement problem. J. Sched. 7, 365–374.
2020. Using lel and scenarios to derive mathematical programming models. application in a fresh tomato packing problem. Comput. Electron. Agric. 170, 105242..
Gendreau, M., Potvin, J., 2010. Handbook of metaheuristics, vol. 272. Springer International Publishing..
Gerhard, W., Heike, H., Holger, S., 2007. An improved typology of cutting and packing problems. Eur. J. Oper. Res. 183, 1109–1130.
Glover, F., 1989. Tabu search–part 1. ORSA J. Comput. 1, 190–206.
Glover, F., 1990. Tabu search–part ii. ORSA J. Comput. 2, 4–32.
Graham, R.L., Lubachevsky, B.D., 1995. Dense packings of equal disks in an equilateral triangle: From 22 to 34 and beyond. J. Combinatorics, 2.
Graham, R., Lubachevsky, B., 1995. Dense packings of equal disks in an equilateral triangle: From 22 to 34 and beyond. Electron. J. Combinatorics 2, A1 (2004), 39..
He, K., Dosh, M., Jin, Y., Zou, S., 2018. Packing unequal circles into a square container based on the narrow action spaces. Sci. China Inf. Sci. 61, 048104.
Hsu, H.P., Mehra, V., Nadler, W., Peter, G., 2003. Growth based optimization algorithm for lattice heteropolymers. Phys. Rev. E 68, 021113.
Huang, W., Li, Y., Jurkowiak, B., Li, C., Xu, R., 2003. A two-level search strategy for packing unequal circles into a circle container. In: Principles and Practice of Constraint Programming. Springer, Berlin Heidelberg, pp. 868–872.
Huang, W., Li, Y., Akeb, H., Li, C., 2005. Greedy algorithms for packing unequal circles into a rectangular container. J. Oper. Res. Soc. 56, 539–548.
Huang, W., Lu, Z.P., Shi, H., 2005. Growth algorithm for finding low energy configurations of simple lattice proteins. Phys. Rev. E 72, 016704.
Jingfa, L., Jian, L., Zhipeng, L., Yu, X., 2017. A quasi-human strategy-based improved basin filling algorithm for the orthogonal rectangular packing problem with mass balance constraint. Comput. Ind. Eng. 107, 196–210.
Kirkpatrick, S., Gelatt, C., Vecchi, M., 1983. Optimization by simulated annealing. Science 220, 671–680.
Korf, R.E., 2003. Optimal rectangle packing: initial results. In: ICAPS.
Kun, H., Mohammed, D., 2017. A greedy heuristic based on corner occupying action for the 2d circular bin packing problem. Springer Singapore, 75–85.
Kun, H., Wenqi, H., 2011. An efficient placement heuristic for three-dimensional rectangular packing. Comput. Oper. Res. 38, 227–233.
Kun, H., Menglong, H., Chenkai, Y., 2015. An action-space-based global optimization algorithm for packing circles into a square container. Comput. Oper. Res. 58, 67–74.
Kun, H., Mohammed, D., Shenghao, Z. Packing unequal circles into a square container by partitioning narrow action, spaces and circle items. CoRR abs/1701.00541..
Kun, H., Hui, Y., Zhengli, W., Jingfa, L., 2018. An efficient quasi-physical quasi-human algorithm for packing equal circles in a circular container. Comput. Oper. Res. 92, 26–36.
Leonardo, J., Reinaldo, M., Denise, S.Y., 2012. Three-dimensional container loading models with cargo stability and load bearing constraints. Comput. Oper. Res. 39, 74–85.
López, C.O., Beasley, J., 2011. A heuristic for the circle packing problem with a variety of containers. Eur. J. Oper. Res. 214, 512–525.
Luo, F., Scherson, I.D., Fuentes, J., 2017. A novel genetic algorithm for bin packing problem in jmetal. In: 2017 IEEE International Conference on Cognitive Computing (ICCC), pp. 17–23.
Maddaloni, A., Colla, V., Nastasi, G., Seppia, M., Iannino, V., 2016. A bin packing algorithm for steel production, in. In: European Modelling Symposium, pp. 19–24.
Martin, L.D., Sandor, P.F., Robert, J.L., 2010. Circle packing for origami design is hard, ArXiv abs 1008.1224..
Mhand, H., Rym, M., 2004. Approximate algorithms for constrained circular cutting problems. Comput. Oper. Res. 31, 675–694.
Mhand, H., Rym, M., 2007. A dynamic adaptive local search algorithm for the circular packing problem. Eur. J. Oper. Res. 183, 1280–1294.
Mhand, H., Rym, M., 2008. Adaptive and restarting techniques based algorithms for circular packing problems. Comput. Optim. Appl. 39, 17–35.
Mostafa, A.M., Sohel, R., Kaykobad, M., Manning, E.G., Shoja, G.C., 2006. Solving the multidimensional multiple-choice knapsack problem by constructing convex hulls. Comput. Oper. Res. 33, 1259–1273.

Nikhil, B., Andrea, L., Maxim, S., 2005. A tale of two dimensional bin packing. In: 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS, pp. 657–666.
Osogami, T., Okano, H., 2003. Local search algorithms for the bin packing problem and their relationships to various construction heuristics. J. Heuristics 9, 29–49.
Peeraya, T., Pupong, P., Chris, H., Warattapop, C., 2012. Development of a stochastic optimisation tool for solving the multiple container packing problems. Int. J. Prod. Econ. 140, 737–748.
Rao, R., Iyengar, S., 1994. Bin-packing by simulated annealing. Comput. Math. Appl. 27 (5), 71–82.
Specht, E., 2015. A precise algorithm to detect voids in polydisperse circle packings. Proc. R. Soc. Lond. Ser. A 471, 19.
Stracquadanio, G., Greco, O., Conca, P., Cutello, V., Pavone, M., Nicosia, G., 2015. Packing equal disks in a unit square: an immunological optimization approach. In: International Workshop on Artificial Immune Systems (AIS), pp. 1–5.
Szab, P.G., Markot, M.C., Csendes, T., Specht, E., Casado, L.G., Garcia, A.I., 2007. New Approaches to Circle Packing in a Square: With Program Codes (Springer Optimization and Its Applications). Springer-Verlag.
Túlio, A.M.T., Eline, E., Tony, W., Greet, V.B., 2017. A two-dimensional heuristic decomposition approach to a three-dimensional multiple container loading problem. Eur. J. Oper. Res. 257, 526–538.
Viegas, J.L., Vieira, S.M., Henriques, E.M.P., Sousa, J.M.C., 2015. A tabu search algorithm for the 3d bin packing problem in the steel industry. In: CONTROLO'2014 Proceedings of the 11th Portuguese Conference on Automatic Control. Springer International Publishing, pp. 355–364.
Wenqi, H., Yu, L., Chumin, L., Ruchu, X., 2006. New heuristics for packing unequal circles into a circular container. Comput. Oper. Res. 33, 2125–2142.
Zeng, Z., Yu, X., He, K., Huang, W., Fu, Z., 2016. Iterated tabu search and variable neighborhood descent for packing unequal circles into a circular container. Eur. J. Oper. Res. 250, 615–627.
Zhipeng, L., Wenqi, H., 2008. Perm for solving circle packing problem. Comput. Oper. Res. 35, 1742–1755.
Zhizhong, Z., Xinguo, Y., Kun, H., Zhanghua, F., 2018. Adaptive tabu search and variable neighborhood descent for packing unequal circles into a square. Appl. Soft Comput. 65, 196–213.