A heuristic algorithm with multi-scale perturbations for point arrangement and equal circle packing in a convex container

Xiangjing Lai[1], Jin-Kao Hao*[2], Dong Yue[3], and Yangming Zhou[4,5]

[1]School of Business, Nanjing University of Information Science and Technology, Nanjing 210044, China.
laixiangjing@gmail.com (Xiangjing Lai)
[2]LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France.
jin-kao.hao@univ-angers.fr
*Corresponding author
[3]Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China.
medongy@vip.163.com (Dong Yue)
[4]Data-Driven Management Decision Making Lab, Shanghai Jiao Tong University, Shanghai 200030, China.
yangming.zhou@sjtu.edu.cn (Yangming Zhou)
[5]Sino-US Global Logistics Institute, Antai College of Economics and Management, Shanghai Jiao Tong University, Shanghai 200030, China.

April 26, 2025

Computers and Operations Research, https://doi.org/10.1016/j.cor.2025.107099

Abstract

The point arrangement and equal circle packing problems are a category of classic max-min constrained optimization problems with many important applications. Being computationally very challenging to solve, they have been widely studied in operations research and mathematics. We propose a heuristic algorithm for the point arrangement and equal circle packing problems in various convex containers. The algorithm relies on several complementary search components, including an unconstrained optimization procedure that ensures diversified and intensified searches, an optima exploitation based adjustment method for the radius of circles, and a monotonic basin-hopping method with multi-scale perturbations. Computational results on numerous benchmark instances show that the proposed algorithm significantly outperforms the existing state-of-the-art algorithms, especially for hard instances or large-scale instances. For the well-known equal circle packing problem in a circular container, it improves the best-known result for 69 out of the 96 hardest instances widely used in the literature. For the majority of the remaining instances tested, the algorithm improves or matches the best-known results with a high success rate, despite of the fact that these instances have been tested by many existing algorithms. Experimental analysis shows that the optima exploitation based adjustment method for the radius of circles plays a crucial role for the high performance of the algorithm and that the multi-scale perturbations are able to significantly enhance

the search ability and robustness of the algorithm. Given the general feature of the proposed framework, it can be applied to other related max-min constrained optimization problems.

Keywords: Packing; circle packing; global optimization; constrained optimization; heuristics.

1 Introduction

The equal circle packing problem and the point arrangement problem are two very classic global optimization problems in the fields of mathematic and operations search, where the equal circle packing problem aims to pack N non-overlapping circles in a given container C such that the common radius of N circles is maximized (Goldberg, 1970), while the point arrangement problem aims to place N points in C such that the minimum distance between points is maximized (Akiyama et al., 2002; Melissen, 1993; Maranas et al., 1995). These two problems are closely related, and both are referred to in the literature as the continuous p-dispersion problem (Drezner and Erkut, 1995; Baur and Fekete, 2001; Dimnaku et al., 2005). Moreover, they can be mutually converted into each other for some regular containers such as a circle or a convex polygon.

The equal circle packing problem and the point arrangement problem have a number of relevant applications, such as the max-min distance design (Van Dam et al., 2007), circular cutting, container loading, cylinder packing, facility dispersion, and communication networks (Castillo et al., 2008). On the other hand, they are computationally very challenging because the number of local optima on the potential energy surface (PES) of the objective function increases exponentially with the problem size, and the global optimum solution may locate at a very narrow and deep funnel on the PES of the objective function. Furthermore, they are shown to be NP-hard (Demaine et al., 2010).

Due to their practical importance and theoretical significance, the equal circle packing problem and the point arrangement problem have been widely studied by researchers from various fields, and a large number of algorithms have been proposed in the literature, including the analytical approaches, exact algorithms, and heuristic algorithms. Due to the continuity of the solution space and the large number of local optima, the capabilities of analytical approaches and exact algorithms are severely limited, leading to a small number of related approaches to solve some small instances exactly. For example, using an analytical approach, Melissen (1994) and Fodor (1999) respectively proved the optimality of solutions for the problems of packing $N = 11$ and 19 equal circles in a circular container. Based on different pruning strategies, several branch-and-bound algorithms have been developed for the equal circle packing problem in a square container (Locatelli and Raber, 2002; Markót and Csendes, 2005; Markót, 2021; Costa et al., 2013). However, it should be noted that according to the reported results, the search capabilities of these approaches are very limited and that the best algorithm among them is only able to solve the small instances with $N < 40$ in a reasonable computational time.

To handle large instances, many heuristic algorithms have been proposed in the literature, mainly including the billiard simulation methods, the discrete optimization methods, and the continuous nonlinear constrained optimization methods.

The billiard simulation approach (Lubachevsky, 1991), which simulates the physical collisions of billiards in an N-component continuous-time dynamic system, is one of the most

popular heuristic algorithms in the early research on circle packing problems. Using a billiard simulation method, Graham and Lubachevsky (1996) and Graham et al. (1998) respectively reported many high-quality solutions for the equal circle packing problems in a square and circular container. Boll et al. (2000) proposed a two-phase billiard simulation algorithm for packing equal circles in a square container and improved the best-known solution for several small instances by the proposed algorithm. Szabó and Specht (2007) systematically predicted the putatively optimal solutions for the equal circle packing problem in a square with up to $N = 200$ circles using a modified billiard simulation algorithm.

The discrete optimization methods, which treat the continuous optimization problems using the discrete strategies, are another category of popular heuristic algorithms for the circle packing problems. For example, Casado et al. (1998) devised a threshold acceptance algorithm called TAMSASS for the packing equal circles in a square and evaluated the algorithm on the instances with $N \leq 100$. By constructing a grid in a given bounded domain, Galiev and Lisafina (2013) proposed several integer linear programming models for the problem of packing equal circles into a given domain. Based on several inequalities and different grids, Litvinchev and Ozuna (2014) and Litvinchev et al. (2016) devised several integer linear programming-based approaches. In addition, based on the Monte Carlo simulation, Liu et al. (2009) proposed an energy landscape paving algorithm for the problem of packing circles into a circular container. Hifi and M'Hallah (2007) presented a dynamic adaptive local search algorithm for the circular packing problem. Based on the maximum hole degree, Akeb et al. (2009) proposed an a beam search algorithm for the circular packing problem. Recently, based on an idea of filtration, Chen et al. (2024) proposed a beam search-based constructive heuristic algorithm for packing unit circles into a circular container.

Most of the existing heuristic algorithms belong to the continuous nonlinear constrained optimization approaches, which aim to find a high-quality solution of a constrained nonlinear continuous function. These approaches can be further divided into three categories according to the local optimization methods used, since the local optimization method plays a crucial role for the performance of nonlinear optimization algorithms.

The first category of continuous nonlinear constrained optimization approaches straightforwardly employ constrained local solvers as their local search procedure. For example, Maranas et al. (1995) proposed a simple heuristic approach that combines the multi-start strategy and a nonlinear programming (NLP) solver MINOS 5.3 for packing equal circles in a square. Based on an NLP solver and two different formulations of the equal circle packing problem, Mladenović et al. (2005) devised a reformulation descent algorithm in which two formulations of the problem are used alternately to escape local optimum traps. Birgin et al. studied the equal circle packing problem by using a popular NLP solver called Algencan, which is an augmented Lagrangian method for smooth general constrained minimization (Birgin and Gentil, 2010; Birgin and Sobral, 2008). Addis et al. (2008) and Grosso et al. (2010) respectively developed the monotonic basin hopping (MBH) method and the population basin hopping (PBH) method for the equal circle packing problem, where the NLP solver SNOPT are used for local optimization. Then, based on SNOPT, López and Beasley (2011) proposed a formulation space search approach that uses different formulations of the equal circle packing problem to escape local traps. Based on the interior-point solver IPOPT and the decomposition method, Stoyan et al. (2020) proposed a unified heuristic approach for solving several circle and sphere packing problems and improved the best-known results for several large-scale instances of the equal circle packing problem in a circular container.

The second category of continuous nonlinear constrained optimization approaches is based

on the idea of first transforming the original constrained optimization problem into an unconstrained optimization problem, and then using an unconstrained local solver, such as the steepest descent method, as the local search procedure. For example, to pack equal circles in a square, Nurmela and Östergård (1997) used a multi-start method combined with the steepest descent method to minimize the unconstrained energy functions $E_m(X) = \sum_{i<j}(\frac{\lambda}{d_{ij}^2})^m$, where d_{ij} is the distance between the centers of the circles c_i and c_j and is represented by trigonometric functions to remove the constraints, λ is a scaling factor, and m is an increasing positive integer, each value of which defines an unconstrained function. Similarly, by using this energy function as well as its variants, Amore and Morales (2023) and Amore (2023) recently investigated the equal circle packing problems in a square container or several regular polygons and produced the encouraging results.

The third category of continuous nonlinear constrained optimization approaches share a common mechanism that dynamically converts the original constrained problem into a series of unconstrained subproblems with a fixed radius of circles, and then handle the corresponding subproblems by an unconstrained optimization method that uses an unconstrained local solver, such as the conjugate gradient method, as its local search method. Moreover, an adjustment method for the radius of circles (or the minimum distance between points) is used to slightly adjust the coordinates of circles so that the radius of circles is maximized while maintaining the feasibility of the resulting solution, once a feasible solution is found by the unconstrained optimization method. In other words, this category of approaches are mainly composed of an adjustment method for the radius of circles and an unconstrained optimization method for solving the subproblems. For example, Huang and Ye (2011) proposed a quasi-physical global optimization algorithm, where a global optimization method based on the quasi-physical local search is used to solve the subproblems, and a binary search method is used to determine the radius of circles (or container). By employing the limited memory BFGS (L-BFGS) method (Liu and Nocedal, 1989) to perform the local optimization and the binary search method to adjust radius of circles (or the circular container), He et al. (2018) proposed a quasi-physical quasi-human algorithm. Recently, Lai et al. (2022, 2023) presented two highly efficient heuristic algorithms for packing equal circles into a convex container by integrating a dynamic thresholding search method to solve the subproblems and the sequential unconstrained minimization technique (SUMT) (Fiacco and McCormick, 1964) to adjust the radius of circles. Their computational results showed that the performance of the algorithm depends not only on the method used to solve the subproblems, but also largely depends on the method used to adjust the radius of the circles. The computational results also showed that the SUMT method is the most efficient method to adjust the radius of the circles for most instances. Subsequently, based on such a radius adjustment method, Zhou et al. (2024) recently proposed a decomposition-based global optimization algorithm for the equal circle packing problem in a circular container on large scale. Very recently, Basurto et al. (2024) proposed a replica exchange/event-chain Monte Carlo method for packing equal circles into a minimum circular container.

In addition to the above approaches, there are a number of other circle packing methods according to the update history of the best-known results on the popular Packomania website (Specht, 2023), such as the Packntile program from the Pack'n'tile contest (http://www.algit.eu/htmlji/Packntile/Packing_Contest_01052010.html), Cantrell's algorithms, and Specht's algorithms (see the update history of the Packomania website (Specht, 2023) for details). These methods produced or improved on the best-known solutions for a number of benchmark

instances when they were proposed, although many of them were not published.

The recent studies in (Lai et al., 2023; Zhou et al., 2024) revealed that the third category of continuous nonlinear constrained optimization approaches introduced above are the state-of-the-art algorithms for the equal circle packing problems and that the SUMT method is the best performing method for adjusting the radius of circles or the minimum distance between points in most cases and significantly outperforms the popular binary search method (He et al., 2018; Huang and Ye, 2011). On the other hand, recent studies (Zhou et al., 2024) also showed that the SUMT method, which is the dominant method used for radius adjustment, can perform poorly for some hard instances in terms of the success rate of the algorithm. For example, even the iterated dynamic thresholding search (IDTS) algorithm (Lai et al., 2022), which is one of the best performing algorithms for the equal circle packing problem and relies on the SUMT method, has a very low success rate ($= 1/20$) for many hard instances. Therefore, there is still room to improve the effectiveness of the adjustment method for the radius of circles or the minimum distance between points.

Motivated by this observation, this work first aims to explore the adjustment methods for the radius of circles or the minimum distance between points to further improve the algorithm performance for the two packing problems, especially to better solve the hardest instances. This work is also motivated by another observation. In fact, compared to the non-convex or multi-connected containers (Dai et al., 2021), which require dedicated models and approaches, the convex containers are much easier to handle by means of a unified optimization approach. However, most of the existing studies focus only on some particular regular containers, such as a circular container and a square container, while ignoring other forms of containers. Therefore, the second objective of this work is to fill the gap by devising a general-purpose, high-performance algorithm that can be applied to a variety of convex containers for both the point-arrangement and equal-circle packing problems.

The main contributions of this work can be summarized as follows. First, we propose a novel adjustment method for the minimum distance between points for the point arrangement and equal circle packing problems. This adjustment method has a certain global search ability since it is able to visit a number of local optimal solutions by using many small perturbations. Because of its generality, this method can benefit solution approachs for other max-min (or min-max) constrained optimization problems, such as the equal sphere packing problem. Second, based on this distance adjustment method and a new unconstrained optimization method for handling the unconstrained subproblems derived from the initial constrained problems, we propose a new heuristic algorithm for both the point arrangement and equal circle packing problems in a variety of convex containers. The framework of the proposed algorithm is general enough to be applied to related constrained optimization problems, such as covering a region by equal circles or spheres (Birgin et al., 2024).

The rest of paper is organized as follows. Section 2 gives a mathematical formulation of the point arrangement problem in a convex container and a transformation method from the equal circle packing problem to the point arrangement method. In section 3, the proposed global optimization algorithm is described in detail. In Section 4, the performance of the proposed algorithm is evaluated based on a number of benchmark instances respectively for the point arrangement problem and the equal circle packing problem in a variety of convex containers. In Section 5, the key algorithmic components are analyzed to check their impacts on the performance of the algorithm. In the last section, we summarize the present work and provide research perspectives.

2 Formulations of the point arrangement and equal circle packing problems

In this section, we formulate the point arrangement problem and its subproblems, and then introduce a conversion method from the equal circle packing problem to the point arrangement problem.

2.1 The point arrangement problem

Given a positive integer N and a convex container, the point arrangement problem aims to place N points in the container, such that the minimum distance d between the points is maximized. Formally, given a convex container C with a piecewise smooth boundary which is composed of m two-variable smooth functions $g_k(x, y)$ $(1 \leq k \leq m)$, and a candidate solution $X = (x_1, y_1, x_2, y_2, \ldots, x_N, y_N)$ (i.e., a packing configuration of N points), the corresponding point arrangement problem can be written as a constrained nonlinear optimization problem as follows:

$$\text{Maximize} \quad d \tag{1}$$

$$\text{Subject to} \quad \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq d, 1 \leq i \neq j \leq N \tag{2}$$

$$g_k(x_i, y_i) \leq 0, \quad 1 \leq i \leq N, 1 \leq k \leq m \tag{3}$$

where d is the objective to be optimized and represents the minimum distances between N points in the candidate X, the set of separation constraints (2) indicates that the distance between any two points must be larger or equal to d, and the set of containment constraints (3) indicates that all N points must be contained in the container.

Such a constrained optimization problem can be converted into a series of unconstrained optimization subproblems with a fixed d value by converting all the constraints to the objective by means of a penalty method. Specifically, suppose that the value of d is fixed to a constant, the corresponding subproblem can be described as follows:

$$E_d(X) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} O_{ij}^2 + \sum_{i=1}^{N} \sum_{k=1}^{m} O_{ik}^2 \tag{4}$$

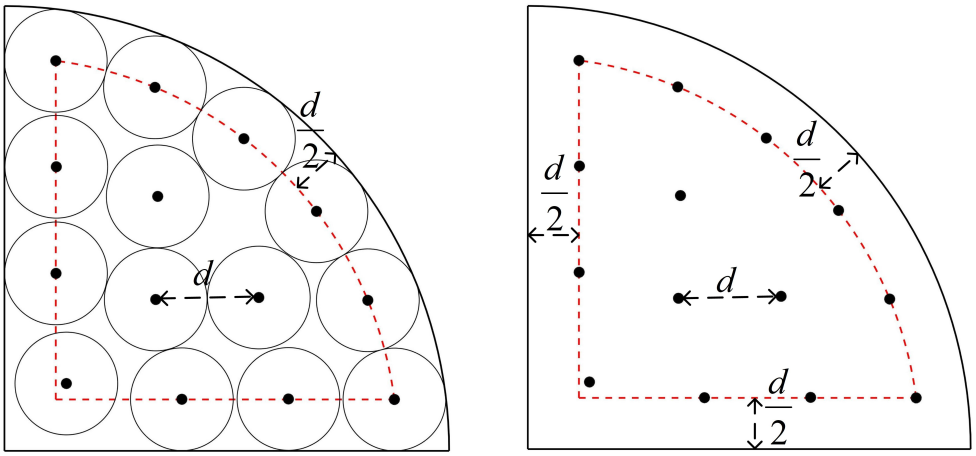
$$O_{ij} = max\{0, d - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\} \tag{5}$$

$$O_{ik} = max\{0, g_k(x_i, y_i)\} \quad 1 \leq i \leq N, 1 \leq k \leq m \tag{6}$$

where O_{ij} denotes the degree of violation on the separation constraint between points i and j, and O_{ik} indicates the degree of violation on the containment constraint between point i and the boundary $g_k(x, y)$ $(1 \leq k \leq m)$. As such, $E_d(X)$ measures the total degree of constraint violation of candidate solution X, and $E_d(X) = 0$ means that X is a feasible solution, and infeasible otherwise.

As a result, the original point arrangement problem can be handled by optimizing in order the above unconstrained subproblems by an unconstrained optimization method and a minimum distance adjustment method.

2.2 Conversion of the circle packing problem into the point arrangement problem



(a) Circle packing in a container ($r = \frac{d}{2}$)

(b) Point arrangement in a subregion

Figure 1: An illustrative example of the conversion of the equal circle packing problem into the point arrangement problem, where the former is converted into the latter by restricting all points (i.e., the centers of circles) to a subregion indicated by the dotted lines.

Similar to the point arrangement problem, the equal circle packing problem can also be converted into a series of optimization subproblems with a fixed radius r of circles. Then, the original circle packing problem can be solved via solving in order the subproblems with an increasing radius of circles.

Due to the close relationship between the point arrangement problem and the equal circle packing problem, the subproblems of the equal circle packing problem with a fixed radius r of circles can be easily converted into the corresponding subproblems of the point arrangement problem defined in Section 2.1, especially for convex containers whose boundary is composed of m ($m > 0$) line segments or circular arcs.

Specifically, given the radius r of N circles and a regular convex container C, the problem of packing N circles with a radius r into the container C can be converted into a point arrangement problem that aims to place the N points (i.e., the centers of the circles) into a subregion of the container such that the minimum distance between N points is greater than or equal to $2r$. Figure 1 provides an illustrative example of this conversion, where the subregion indicated by the dotted lines in the subfigure (b) represents the container of the corresponding point arrangement problem and its boundaries can be written as $g_k^d(x, y)$ ($1 \leq k \leq m$), depending on the value of d and the original boundary function $g_k(x, y)$ of container.

3 Optimization Method

In this study, we propose a new heuristic algorithm for the point arrangement problem and the equal circle packing problem in a convex container. The central ideas of the algorithm can be summarized as follows. First, these two constrained optimization problems are dynamically

converted into a series of unconstrained subproblems, and these subproblems are handled in order by a new unconstrained optimization algorithm. Second, the proposed algorithm employs in its components the multi-scale perturbation strategy to enhance the search ability and the robustness of the algorithm. The preliminary experiments reveal that the strength of perturbations plays a key role for the search ability especially for hard instances. Therefore, the proposed algorithm is named as the heuristic algorithm with multi-scale perturbations (abbreviated as HAMSP). Third, the algorithm uses a new and highly efficient adjustment method for the radius of the circles (or the minimum distance between the points). In the following subsections, the general procedure and ingredients of the proposed algorithm are described.

3.1 Main framework of the proposed HAMSP algorithm

The proposed HAMSP algorithm is performed in three phases and contains five main components: the generation of initial solutions, the monotonic basin-hopping (MBH) method, the diversification and intensification based heuristic (DIH) method, and two adjustment methods for the minimum distance d between points. The pseudo-code of the algorithm is given in Algorithm 1, where X and d respectively denote the current solution and the corresponding minimum distance between the points (or the centers of the circles), and X^* and d^* denote the best solution found so far and the corresponding distance between the points. First of all, the algorithm estimates an initial value for the minimum distance d between N points by supposing that a proper equal circle packing configuration has an estimated packing density p (line 1).

During the first phase, the algorithm tries to find a high-quality initial value for the minimum distance d between the points by a multi-start strategy combined with the MBH method (lines 3–11). At each time, an initial solution is generated by randomly distributing the N points (or the circle centers) in the container. Subsequently, the MBH method is used to minimize the potential function $E_d(X)$ starting from the initial solution and then a distance adjustment method (i.e., $Adjust\_Minimum\_Dist$) is used to adjust the minimum distance d between the points of the solution returned by the MBH method. After that, the best d value found so far is saved as d^*.

The second phase of the algorithm is the main search phase of this optimization approach, where the solution initialization, the optimization of the unconstrained subproblem associated with the current d, and an optima exploitation based (OEB) adjustment procedure are performed iteratively until the time limit t_{max} is reached (lines 13-28). At each iteration, based on the current d value determined as $d = d^* + \Delta_d$ (line 14), the algorithm first generates randomly an initial solution and then improves its quality using the diversification and intensification based heuristic (DIH) method that aims to minimize the unconstrained function $E_d(X)$ (lines 15-16). Then, the MBH procedure with very small perturbations (denoted by $Weak\_MBH$) is used to further minimize the function $E_d(X)$ (line 17). Subsequently, the OEB adjustment method (i.e., $OEB\_Adjustment()$) is used to adjust the minimum distance d between the points if a feasible solution X with $E_d(X) < 10^{-30}$ is obtained by the $Weak\_MBH$ procedure. X^*, d^* and Δ_d are updated accordingly if an improving solution is obtained by the distance adjustment method, and Δ_d is reset to 0 otherwise (lines 20–26). It should be noted that the use of Δ_d in the setting of d aims to produce a squeezing effect for the packing configuration.

At the final phase of the algorithm (line 29), the best solution found (X^*) is further

Algorithm 1: Main framework of the proposed HAMSP algorithm

Input: Number of points (or circles) to be packed (N), maximum time limit (t_{max}), estimated packing density of initial configuration used (p), the number of initial solution tested (T), strengthes of perturbation of MBH (Δ_1 and Δ_2)

Output: The best solution found (X^*, d^*)

$d \leftarrow \sqrt{\frac{4\times Area\times p}{N\times\pi}}$ /* Estimate the minimum distance d among N points, and $Area$ denotes the area of container */

$d^* \leftarrow 0$

/* Find a high-quality d^* by checking T initial solutions randomly generated. */

for $i \leftarrow 1$ to T do

  $X \leftarrow RandomSolution()$ /* Generate randomly a configuration */

  $MBH(X, E_d(\cdot), d, \Delta_1)$ /* Optimize $E_d(X)$ by Algorithm 2 */

  $(X, d) \leftarrow Adjust\_Minimum\_Dist(X, d)$ /* Adjust the minimum distance between points by the SUMT method, Algorithm 5, */

  if $d > d^*$ then

    $d^* \leftarrow d$

    $X^* \leftarrow X$

  end

end

/* The main search engine of optimization algorithm */

$\Delta_d \leftarrow 0$

while time() $\leq t_{max}$ do

  $d \leftarrow d^* + \Delta_d$ /* Aims to produce a squeezing effect on configuration by increasing the value of d or the radius of circles */

  $X \leftarrow RandomSolution()$ /* Generate randomly a configuration */

  $X \leftarrow Diversification\_Intensification(X, E_d(\cdot), d)$ /* Based on the current d, handle the subproblem by optimizing $E_d(X)$, Algorithm 3 */

  $Weak\_MBH(X, E_d(\cdot), d^*, \Delta_2)$ /* Further optimize $E_d(X)$ with $d = d^*$ using MBH with very small perturbations, Algorithm 2 */

  /* Adjust minimum distance d once a feasible solution X (i.e., $E_{d^*}(X) < 10^{-30}$) is met */

  if $E_{d^*}(X) < 10^{-30}$ then

    $(X, d) \leftarrow OEB\_Adjustment(X, d^*)$ /* Adjust minimum distance d by Algorithm 4 */

    if $d > d^*$ then

      $d^* \leftarrow d$

      $X^* \leftarrow X$

      $\Delta_d \leftarrow d - d^*$ /* Update adaptively Δ_d */

    else

      $\Delta_d \leftarrow 0$

    end

  end

end

/* The postprocessing of the best solution found */

$(X^*, d^*) \leftarrow OEB\_Adjustment(X^*, d^*)$ /* Algorithm 4 */

optimized by a post-processing procedure, where the minimum distance d between the points is maximized by a reinforcement adjustment procedure, i.e., the OEB adjustment procedure with a stronger parameter setting of $\mu_0 = 10^4$. Finally, the best solution found X^* and the corresponding distance d^* are returned as the output of algorithm.

3.2 Monotonic basin-hopping method

Algorithm 2: Monotonic basin hopping (MBH) method for optimizing the function $E_d(X)$

Input: Input solution X_0, the objective function $E_d(X)$ as well as the corresponding d, search depth ($MaxNoImprove$), strength of perturbation (Δ)
Output: The best solution found X

```
Function MBH() or Weak_MBH()
X^b ← X_0
NoImprove ← 0
while (NoImprove ≤ MaxNoImprove) ∧ (E_d(X^b) > 0) do
    X ← Shake(X^b, Δ)    /* Shift randomly and uniformly each coordinate of X^b in [−Δ, Δ] */
    X ← LocalSearch(X, E_d(·))    /* Minimize function E_d(X) by L-BFGS */
    if E_d(X) < E_d(X^b) then
        X^b ← X
        NoImprove ← 0
    else
        NoImprove ← NoImprove + 1
    end
end
```

The proposed algorithm employs the popular monotonic basin-hopping method (Leary, 2000) to perform an intensified search (Algorithm 2). Given an input solution $X = (x_1, y_1, x_2, y_2, \ldots, x_N, y_N)$, the monotonic basin-hopping method performs a number of iterations to improve the input solution until the current solution can not be improved during $MaxNoImprove$ consecutive iterations or the optimal solution has been found, where $MaxNoImprove$ is a parameter called the search depth. At each iteration, the current solution is first perturbed by a shaking operator and then improved by a local optimization method, and the resulting solution is used to replace the current solution if and only if the resulting solution has a better objective value. Specifically, the shaking operator shifts simultaneously and randomly each coordinate x_i (or y_i) of the solution X in an interval $[-\Delta, \Delta]$ in which Δ is a parameter called the strength of perturbation. As for the local optimization method, many popular mathematical solvers can be used, such as the quasi-Newton methods and the conjugate gradient methods. In this study, the limited-memory quasi-Newton method (i.e., L-BFGS) (Liu and Nocedal, 1989) is used as the local optimization method. It is worth noting that due to its simplicity and effectiveness the monotonic basin-hopping method has been applied to a number of global optimization problems, such as the circle packing problem and the structural optimization of atomic clusters (Addis et al., 2008; Doye et al., 2004).

As a fundamental component of the proposed algorithm, the monotonic basin-hopping method is used in the different situations according to the current purposes. When the strong perturbations such as $\Delta = 0.4 \times d$ are adopted, the monotonic basin-hopping method has a

strong ability to escape the local optimal traps and the corresponding procedure is denoted as $MBH()$ in the proposed algorithm. On the contrary, when the weak perturbations such as $\Delta = 0.05 \times d$ are adopted, the purpose is to intensify the search and the corresponding procedure is denoted as $Weak\_MBH()$.

3.3 Diversification and intensification based heuristic method

Algorithm 3: Diversification and intensification-based heuristic (DIH) method for optimizing the function $E_d(X)$

Input: Input solution X_0, the objective function $E_d(X)$ as well as the corresponding d, search depth ($MaxIter$), degree of diversification (β), strength of perturbation of MBH (Δ_1)
Output: The best solution found $X^b_{exploit}$

```
Function Diversification_Intensification()
Iter ← 0
X^b_exploit ← X_0
while (Iter ≤ MaxIter) ∧ (E_d(X^b_exploit) > 0) do
    /* Perform diversified search starting from X^b_exploit */
    X ← X^b_exploit
    for i ← 1 to β do
        X ← Shake(X, Δ_1)
        X ← LocalSearch(X, E_d(·))        /* Minimize E_d(X) by L-BFGS */
    end
    /* Perform intensified search using the MBH method */
    X ← MBH(X, E_d(·), d, Δ_1)            /* Algorithm 2 */
    /* Update the best solution found X^b_exploit */
    if E_d(X) < E_d(X^b_exploit) then
        X^b_exploit ← X
    end
    Iter ← Iter + 1
end
```

The diversification and intensification based heuristic (DIH) method can be regarded as an unconstrained optimization approach, which aims to find a high-quality solution of a high-dimensional nonlinear differentiable function. The basic idea of the DIH method is to reach a desirable tradeoff between intensification and diversification of the search by performing alternately an intensified search procedure and a diversified search procedure. However, unlike the existing diversification mechanisms in the literature, to escape from a very deep local trap, our diversification phase of the DIH method performs a fixed number (β) of perturbations followed by a short local optimization, instead of a violent perturbation that may destroy dramatically the solution. That is, each perturbation of the diversification phase slightly destroys the current solution and the total diversification degree depends on the accumulated change of many small perturbations over the current solution.

The pseudo-code of the DIH method is given in Algorithm 3, where X denotes the current solution, $E_d(X)$ is the objective function to be minimized, and $X^b_{exploit}$ records the best solution found during the search process. Starting from an input solution X_0, the DIH

method performs a number of iterations to improve the solution until the maximum number $MaxIter$ of iterations is reached or the global optimal solution X with $E_d(X) = 0$ is found (lines 4–15). At each iteration, three main steps are performed in order. At first, starting from the best solution found $X^b_{exploit}$, the diversified search is conducted by performing β consecutive random perturbations and subsequent local optimizations (lines 5–9), where β is a parameter that measures the degree of diversification and a larger β value means a stronger diversification degree. Then, the intensified search is conducted by running the MBH procedure with the moderate-strength perturbations (i.e., Algorithm 2) from the solution returned by the diversified search (line 10). Finally, $X^b_{exploit}$ is updated once an improving solution is found by the MBH procedure (lines 11-13). As one of main components of the proposed HAMSP algorithm, the DIH method is used to solve the subproblems of the point arrangement problem and the equal circle packing problem, i.e., minimizing the potential function $E_d(X)$ defined by Eq. (4).

3.4 Adjustment method for the minimum distance between the points

Given a packing configuration (X_0, d_0), how to maximize the minimum distance d between points (or the centers of circles) while maintaining the feasibility of the resulting solution is a very crucial decision for the proposed algorithm. In fact, this is equivalent to solving a max-min constrained optimization problem, which is difficult to handle by the popular local optimization methods, such as the quasi-Newton methods and the conjugate gradient methods. In the previous studies, several distance adjustment methods have been proposed to maximize the radius of circles, such as the binary search method (He et al., 2018; Huang and Ye, 2011) and the sequential unconstrained minimization technique (SUMT) (Fiacco and McCormick (1964); Huang and Ye (2010); Lai et al. (2022, 2023)).

However, according to our experiments (see Section 5.1), the methods like SUMT perform poorly on some hard instances. The fundamental reason why these methods failed to optimize these hard instances is that there may exist a large number of local optimal solutions or saddle points in the vicinity of the input solution for the constrained optimization problem, but these methods return only one of them and ignore the rest, which greatly affects the search ability of the algorithm.

To better handle this problem, we propose in this work an optima exploitation based (OEB) adjustment method whose fundamental idea to exploit as many local optima as possible in the vicinity of the input solution for the constrained optimization problem and then returns the best local optimal solution found. Our OEB adjustment method is mainly composed of two algorithmic components: the standard SUMT method and the MBH method using small perturbations (denoted by $Weak\_MBH$), where the standard SUMT method aims to find a close local optima from the input solution and the MBH algorithm aims to find a feasible solution by exploring a number of additional local optima near the input solution. Hence, unlike the previous adjustment methods, the OEB algorithm can be regarded as a global search method.

The pseudo-code of our OEB adjustment method is given in Algorithm 4. Starting from a feasible input solution, the OEB adjustment method first performs the SUMT procedure to preliminarily adjust the packing configuration (line 2), and then enters a 'while' loop to further seek for possible improvement. At each iteration of the loop, the value of d is first increased by a small number δ that is a predetermined parameter, and then the packing configuration is improved by minimizing the function $E_d(X)$ using the $Weak\_MBH$ procedure (lines 6-7).

Algorithm 4: Optima exploitation based adjustment method for the minimum distance between points

Input: Input solution X_0, minimum distance between points (d_0), expansion factor δ, strength of perturbation of MBH (Δ_2)

Output: The best solution found (X_{best}) and the minimum distance between points (d_{best})

Function $OEB\_Adjustment()$
$(X_{best}, d_{best}) \leftarrow Adjust\_Minimum\_Dist(X_0, d_0)$ /* Algorithm 5, adjust the minimum distance between points */
$Improve \leftarrow$ True
while $Improve$ do
    $d \leftarrow d_{best} \times (1 + \delta)$ /* δ is an expansion factor of d */
    /* Minimizing $E_d(S)$ by MBH with tiny perturbations */
    $X \leftarrow Weak\_MBH(X, E_d(\cdot), d, \Delta_2)$ /* Minimize $E_d(X)$ by Algorithm 2 */
    $X \leftarrow Weak\_MBH(X, E_{d_{best}}(\cdot), d_{best}, \Delta_2)$ /* Minimize $E_{d_{best}}(X)$ by Algorithm 2 */
    /* $E_{d_{best}}(X) < 10^{-30}$ means X is feasible, where 10^{-30} denotes the precision */
    if $E_{d_{best}}(X) < 10^{-30}$ then
        $(X, d) \leftarrow Adjust\_Minimum\_Dist(X, d_{best})$ /* Algorithm 5 */
        if $d > d_{best}$ then
            $X_{best} \leftarrow X$
            $d_{best} \leftarrow d$
            $Improve \leftarrow$ True
        else
            $Improve \leftarrow$ False
        end
    else
        $Improve \leftarrow$ False
    end
end

Algorithm 5: Adjusting the minimum distance between points by the standard SUMT method

Input: Input solution $S_0 = (X_0, d_0)$, parameter μ_0

Output: The feasible local minimum solution $S = (X, d)$

Function Adjust_Minimum_Dist()
$X \leftarrow X_0$, $d \leftarrow d_0$
/* $\mu_0 = 10^2$, $K = 30$ in this study */
$\mu \leftarrow \mu_0$
for $k \leftarrow 1$ to K do
    $(X, d) \leftarrow LocalsSearch(X, d)$ /* Minimize $U_\mu(X, d)$ using L-BFGS */
    $\mu \leftarrow 2 \times \mu$
end
return (X, d)

Then, the SUMT method is used to adjust the configuration again once a feasible solution X with $E_d(X) = 0$ is found by the $Weak\_MBH$ method and the best solution is updated if an improved solution is found (lines 9-12). The OEB adjustment procedure stops once the $Weak\_MBH$ procedure does not find a feasible solution for the current d value or the SUMT procedure does not find an improved solution.

The basic idea of the underlying SUMT method is to convert a constrained optimization problem into a series of unconstrained optimization subproblems, and then solve in order the subproblems using an unconstrained optimization method. Finally, the SUMT method returns a feasible solution in which the objective function is locally optimized.

Formally, for the point arrangement problem and the equal circle packing problem, the corresponding unconstrained subproblems can be written as follows:

$$Minimize \quad U_\mu(X, d) = -\frac{d^2}{\mu} + E(X, d) \tag{7}$$

where μ is a penalty factor and thus $U_\mu(X, d)$ represents an unconstrained optimization function when the value of μ is fixed to a constant, $X = (x_1, y_1, x_2, y_2, \ldots, x_N, y_N)$ are decision variables representing the dispersion points (or the centers of circles) and d is a decision variable representing the allowed minimum distance among the points or the centers of the circles. The term $E(X, d)$ containing $2N + 1$ variables measures the degree of constraint violations and can be written as follows:

$$E(X, d) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} O_{ij}^2 + \sum_{i=1}^{N} \sum_{k=1}^{m} (O_{ik}^{'})^2 \tag{8}$$

where O_{ij} is defined in the previous equation (5). As for the term $O_{ik}^{'}$, which measures the degree of violation on the containment constraint between the point i and the k-th boundary of the container, two different formulations are used for the point arrangement problem and the equal circle packing problem, respectively. First, for the point arrangement problem $O_{ik}^{'}$ is the same as O_{ik} in Eq. (6). However, for the equal circle packing problem, $O_{ik}^{'}$ contains an additional variable d compared to O_{ik}, due to the fact that the circle packing problem is first transformed into a point arrangement problem whose boundary functions $g_k^d(x, y)$ $(1 \leq k \leq m)$ of the container vary dynamically, depending on the variable d (see Figure 1 for an illustrative example of the dependence of boundary of region on the variable d).

The pseudo-code of the SUMT method is given in Algorithm 5. Starting from an input solution (X_0, d_0) and an initial μ, the SUMT method performs K (=30) iterations. At each iteration, the function $U_\mu(X, d)$ is optimized by the L-BFGS method, and then resulting solution is used as the input solution of next iteration, and the value of μ is increased by a fact of 2. Finally, a feasible solution is obtained when μ reaches a very large number, where the value of d is locally maximized.

Two aspects of the OEB adjustment method should be noted because they can greatly affect the effectiveness of the OEB adjustment method. First, the method uses two $Weak\_MBH()$ procedures to search for a feasible solution (lines 5–7), where the first $Weak\_MBH()$ procedure uses a larger d value ($d = (1 + \delta) \times d_{best}$) than d_{best} to squeeze the geometrical configuration of the solution, and the second $Weak\_MBH()$ procedure aims to find a feasible solution for the current d_{best}. Such a strategy is similar to the potential energy transformation method (Doye et al., 2004), and its purpose is not only to speed up the search process, but

also to guide the search direction. Second, the $Weak\_MBH$ procedures perform a series of very small random perturbations to explore the solution space around the current solution, where two small perturbation scales $\Delta_2 = 0.05d$ and $\Delta_2 = 0.15d$ are alternately applied to improve the search robustness. The principle behind the $Weak\_MBH$ method is that in many cases the current solution X is very close to the feasible region of the problem, and large perturbations followed by a local optimization will change the solution sharply and miss feasible solutions in the region. Thus, in these cases the small perturbations followed by a local optimization will be much more efficient to approach the feasible region. Moreover, the preliminary experiments show that the $Weak\_MBH()$ procedure with small random perturbations plays a very important role in enhancing the search capability of the algorithm, especially for some hard instances.

3.5 Discussions on the proposed algorithm

The proposed algorithm has the following remarkable features. First, the proposed algorithm is able to deal with both the point arrangement problem with a piecewise smooth boundary of the container and the equal circle packing problem in which the boundary of the container is composed of line segments or circular arcs. Second, for the proposed algorithm, the circle packing problem is handled from the point of view of the point arrangement problem. That is, the circle packing problem is first dynamically converted to the corresponding point arrangement problem, then is handled using the point arrangement algorithm. Third, the present algorithm is designed only for solving the problems with a convex container, differing from some existing studies such as (Lai et al., 2024), which concentrate on non-convex and multi-connected containers. Fourth, the underlying optima exploitation based (OEB) adjustment method employs the potential energy transformation strategy and the MBH method with multi-scale small perturbations to enhance its robustness, differing the existing adjustment methods in the literature. Fifth, the proposed algorithm uses a new heuristic method called the DIH method, which integrates explicitly the diversified search and the intensified search to optimize the subproblems (i.e., to minimize the objective function $E_d(X)$). As we show in the next section, the proposed algorithm with these new algorithmic components performs well, especially on the most challenging problem instances. Finally, the OEB adjustment method and the DIH method are of general nature. Consequently, they can be applied to other related max-min constrained optimization problems with a proper modification on the optimization model.

4 Experimental Evaluation and Computational Results

In this section, we evaluate the proposed HAMSP algorithm by presenting computational results and making a comparison with the best-known results in the literature. We carry out extensive computational experiments based on a variety of benchmark instances widely used in the previous studies for the point arrangement problem and the equal circle packing problem in a convex container.

4.1 Parameter settings and experimental protocol

The HAMSP algorithm uses several parameters whose default settings given in Table 1 have been empirically determined by preliminary experiments. In Table 1, the parameter p is the

Table 1: Settings of parameters

Parameters	Section	Description	Values
p	3.1	estimated density of initial solutions	$\{0.85, 1.0\}$
T	3.1	number of initial solutions	10
Δ_1	3.3	strength of large perturbation of MBH	$0.4d$
Δ_2	3.2	strength of small perturbation of MBH	$\{0.05d, 0.15d\}$
$MaxNoImprove$	3.2	search depth of MBH	$\{20, 30\}$
$MaxIter$	3.3	degree of intensification	10
β	3.3	degree of diversification	10
δ	3.4	expansion factor of d	10^{-10}
μ_0	3.4	initial value of μ in SUMT	$\{10^2, 10^4\}$

estimated packing density of initial solutions and is set 0.85 and 1.0 respectively for the equal circle packing problem and the point arrangement problem. The parameter T is the number of initial solutions examined in the first phase of the algorithm and its value is set to 10 by default. The parameters Δ_1 and Δ_2 represent the strengthes of perturbations of the $MBH()$ and $Weak\_MBH()$ procedures, and the parameter $MaxNoImprove$ represent the search depth of the MBH method, and their values are set according to the situations that the MBH method is used. In the $Weak\_MBH$ procedure, the value of Δ_2 is alternately used between $0.05d$ and $0.15d$ to enhance the robustness of the algorithm, and the value of $MaxNoImprove$ is set to 30. In the other MBH procedures, the values of Δ_1 and $MaxNoImprove$ are respectively set to $0.4d$ and 20. As for the parameters $MaxIter$ and β that respectively represent the degree of the intensified search and the degree of the diversified search in the DIH method, their values are both set to 10. The parameter δ is an expansion factor of d, and is used in the optima exploitation based adjustment method. As for the parameter μ_0 that is the initial penalty coefficient of the SUMT method, two values 10^2 and 10^4 are respectively used in the second and third phases of the algorithm. In this work, all the computational experiments of the proposed algorithm were executed with the above default parameter settings, unless otherwise stated.

The HAMSP algorithm was implemented in the C++ language and all computational experiments were executed on a computer with an Intel(R) Xeon (R) Platinum 9242 CPU (2.3 GHz), running a Linux operating system. The experiments were conducted on three sets of benchmark instances for the point arrangement problem and four sets of benchmark instances for the equal circle packing problem. Due to the randomness of the proposed algorithm, the algorithm was run 20 times for each tested instance with different random seeds, and the time limit t_{max} of each run was set according to the size of instances. For the small instances with $N \leq 100$ and the medium-sized instances with $101 \leq N \leq 200$, t_{max} is set to one and two hours, respectively. For the larger instances with $N > 200$, t_{max} is set to 6 hours.

4.2 Computational results and comparisons on the point arrangement problem

This section aims to evaluate the performance of the proposed algorithm on the point arrangement problem on three sets of benchmark instances. The first set consists of 50 instances with $151 \leq N \leq 200$ and the container is an isosceles right triangle. These instances are taken from the well-known Packomania website (Specht, 2023). The second set consists of 25 instances in the range of $75 \leq N \leq 200$ and the container is defined by two smooth nonlinear

Table 2: Computational results and comparison on 50 representative instances with $151 \leq N \leq 200$ for the point arrangement problem in an isosceles right triangle. In terms of d_{best}, d_{ave} and d_{worst}, the improved results are indicated in bold compared to the best-known results (BKR) in the literature (Specht, 2023).

N	Best-Known (d^*)	d_{best}	d_{ave}	d_{worst}	$d_{best} - d^*$	SR	time(s)
151	0.06539389	0.06541182	0.06541109	0.06540952	1.79E-05	13/20	3676
152	0.06520869	0.06524338	0.06524338	0.06524338	3.47E-05	20/20	535
153	0.06498504	0.06498782	0.06498782	0.06498782	2.78E-06	20/20	518
154	0.06476024	0.06476033	0.06476033	0.06476033	8.74E-08	20/20	1913
155	0.06453303	0.06455659	0.06455533	0.06454403	2.36E-05	17/20	3344
156	0.06429295	0.06432531	0.06431426	0.06430455	3.24E-05	8/20	2897
157	0.06410261	0.06410720	0.06410720	0.06410720	4.59E-06	20/20	666
158	0.06386645	0.06387430	0.06387430	0.06387430	7.86E-06	20/20	568
159	0.06366169	0.06367830	0.06367830	0.06367830	1.66E-05	20/20	1318
160	0.06343523	0.06347833	0.06347636	0.06343898	4.31E-05	19/20	2386
161	0.06323523	0.06325269	0.06325182	0.06323523	1.75E-05	19/20	3742
162	0.06300838	0.06301209	0.06301209	0.06301209	3.70E-06	20/20	707
163	0.06281728	0.06284458	0.06284458	0.06284458	2.73E-05	20/20	3133
164	0.06257985	0.06263200	0.06262837	0.06262502	5.21E-05	3/20	2922
165	0.06241150	0.06244573	0.06243604	0.06242125	3.42E-05	8/20	4161
166	0.06225576	0.06225576	0.06225576	0.06225576	0.0	20/20	927
167	0.06200927	0.06206068	0.06206068	0.06206068	5.14E-05	20/20	2555
168	0.06179469	0.06184040	0.06184012	0.06183645	4.57E-05	17/20	4385
169	0.06160887	0.06162159	0.06162159	0.06162159	1.27E-05	20/20	792
170	0.06145443	0.06145443	0.06145443	0.06145443	0.0	20/20	806
171	0.06128908	0.06129168	0.06129168	0.06129168	2.60E-06	20/20	820
172	0.06106357	0.06108015	0.06108015	0.06108015	1.66E-05	20/20	1136
173	0.06089417	0.06089672	0.06089672	0.06089672	2.54E-06	20/20	1696
174	0.06065999	0.06071446	0.06070851	0.06068820	5.45E-05	14/20	3518
175	0.06050951	0.06050951	0.06050951	0.06050951	0.0	20/20	661
176	0.06034694	0.06034711	0.06034711	0.06034711	1.72E-07	20/20	974
177	0.06015600	0.06017464	0.06017126	0.06016491	1.86E-05	11/20	4480
178	0.05993953	0.05995694	0.05995370	0.05995242	1.74E-05	5/20	3911
179	0.05977985	0.05980101	0.05979777	0.05979350	2.12E-05	11/20	4174
180	0.05960124	0.05960667	0.05960667	0.05960667	5.42E-06	20/20	2608
181	0.05944638	0.05946775	0.05946775	0.05946775	2.14E-05	20/20	894
182	0.05922093	0.05925016	0.05925016	0.05925016	2.92E-05	20/20	1090
183	0.05907345	0.05911292	0.05910926	0.05909532	3.95E-05	9/20	4211
184	0.05890057	0.05893863	0.05893358	0.05893095	3.81E-05	4/20	2242
185	0.05874263	0.05877579	0.05876359	0.05874973	3.32E-05	4/20	3161
186	0.05858326	0.05859843	0.05859790	0.05858794	1.52E-05	18/20	2815
187	0.05841723	0.05844370	0.05844370	0.05844370	2.65E-05	20/20	3040
188	0.05822837	0.05824421	0.05824245	0.05823527	1.58E-05	10/20	4123
189	0.05807468	0.05808931	0.05808125	0.05807587	1.46E-05	5/20	2056
190	0.05792661	0.05793564	0.05793564	0.05793564	9.03E-06	20/20	1393
191	0.05779322	0.05779422	0.05779422	0.05779422	9.94E-07	20/20	1403
192	0.05758947	0.05761475	0.05761475	0.05761475	2.53E-05	20/20	2405
193	0.05741069	0.05747589	0.05747265	0.05746864	6.52E-05	11/20	4011
194	0.05728257	0.05734092	0.05734092	0.05734092	5.83E-05	18/20	2729
195	0.05714176	0.05716857	0.05715184	0.05714484	2.68E-05	5/20	2826
196	0.05701607	0.05701942	0.05701942	0.05701942	3.35E-06	20/20	1023
197	0.05681954	0.05683607	0.05683607	0.05683607	1.65E-05	20/20	2198
198	0.05667839	0.05670143	0.05669789	0.05669043	2.30E-05	6/20	4129
199	0.05653446	0.05654672	0.05654672	0.05654670	1.23E-05	17/20	4174
200	0.05639931	0.05641389	0.05640459	0.05639950	1.46E-05	1/20	1700
#Improve		47	47	46			
#Equal		3	3	4			
#Worse		0	0	0			

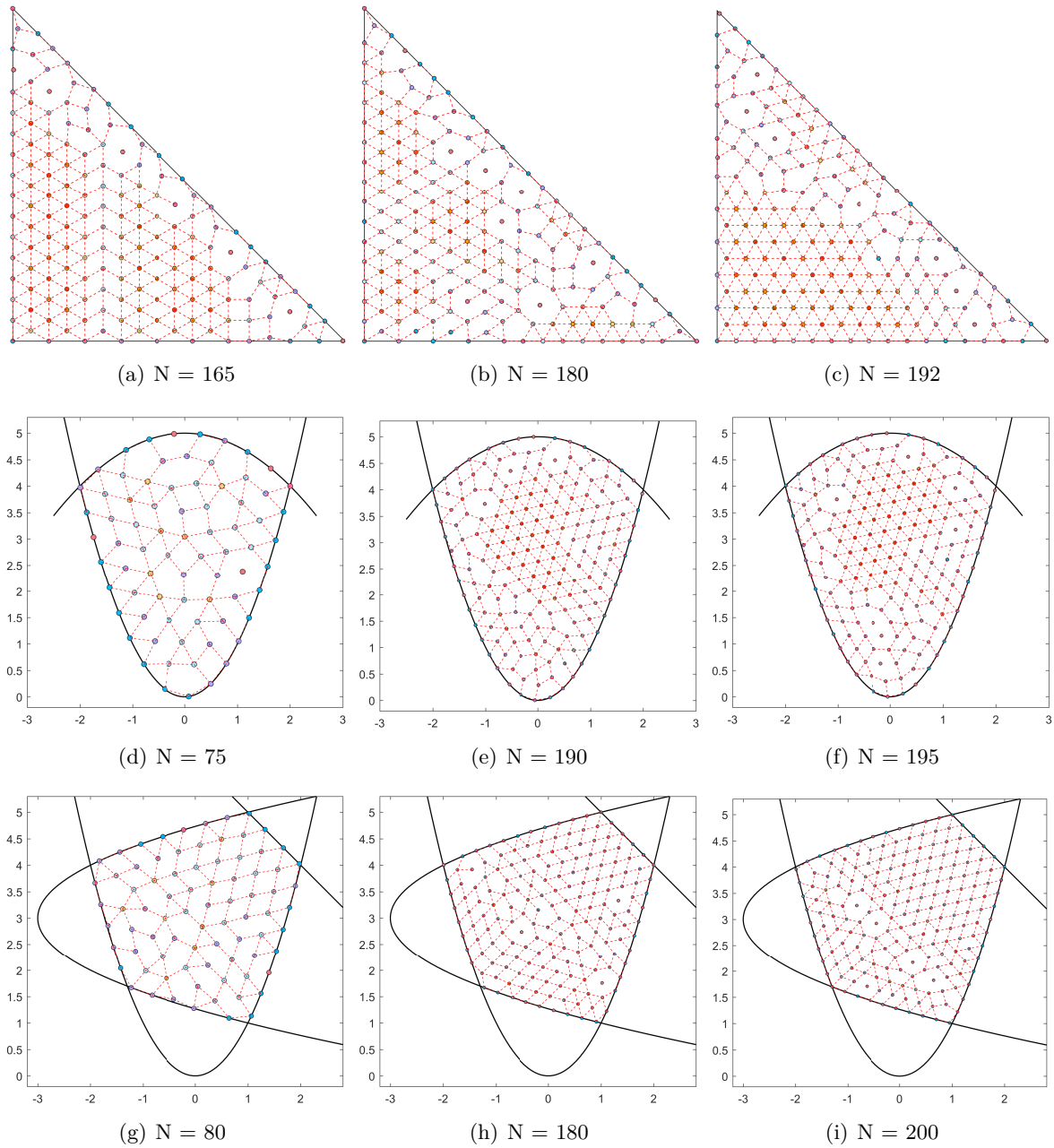
functions: (1) $g_1(x, y) = x^2 - y$, (2) $g_2(x, y) = x^2/4 + y - 5$. Similar to the second set, the third set consists of 25 instances in the range of $75 \leq N \leq 200$ and the container is defined by three smooth nonlinear functions: (1) $g_1(x,y) = x^2 - y$, (2) $g_2(x, y) = -x + y^2 - 6y + 6$, (3) $g_3(x, y) = x + y - 6$. The containers of the second and third sets are taken from (Birgin et al., 2006) for the rectangle packing problem.

The computational results are summarized in Tables 2–4 respectively for these three sets of benchmark instances. In Table 2, the first two columns give the instance size (N) and the best-known result (d^*) reported on the Packomania website (Specht, 2023), columns 3–5 give the best objective value d_{best} (i.e., the minimum distance between points) over 20 independent runs, the average objective value d_{ave}, and the worst objective value d_{worst}. The last three columns respectively give the difference between d_{best} and d^*, the success rate to hit the best result (SR), and the average computational time in seconds to reach the final result ($time(s)$), where a positive value of $d_{best} - d^*$ means that an improved solution is found by the proposed algorithm. In addition, the last three rows "#Improve", "#Equal" and "#Worse" respectively indicate the numbers of instances for which the proposed algorithm obtains an improved, equal and worse result compared to the best-known result in the literature in terms of d_{best}, d_{ave} and d_{worst}. We compare the results of the proposed algorithm only with the best-known results due to the lack of source codes of previous algorithms and the fact that these instances have been widely tested by a number of algorithms (Specht, 2023). In Tables 3 and 4, we give only the detailed computational results of the proposed algorithm due to the fact that these instances are used for the first time and no available results exist in the literature.

Table 3: Computational results on 25 representative instances with $75 \leq N \leq 200$ for the point arrangement problem in a convex region whose boundary is composed of two functions.

N	d_{best}	d_{ave}	d_{worst}	SR	time(s)
75	0.48293031	0.48293031	0.48293031	20/20	48
80	0.46629954	0.46629954	0.46629954	20/20	44
85	0.45058151	0.45058151	0.45058151	20/20	45
90	0.43796527	0.43796527	0.43796527	20/20	76
95	0.42534047	0.42534047	0.42534047	20/20	79
100	0.41286482	0.41286482	0.41286482	20/20	100
105	0.40338203	0.40338203	0.40338203	20/20	97
110	0.39271321	0.39271321	0.39271321	20/20	141
115	0.38314630	0.38314630	0.38314630	20/20	584
120	0.37500452	0.37500452	0.37500452	20/20	203
125	0.36753081	0.36753081	0.36753081	20/20	173
130	0.35946377	0.35946377	0.35946377	20/20	70
135	0.35206437	0.35206437	0.35206437	20/20	3223
140	0.34571829	0.34571829	0.34571829	20/20	245
145	0.33934576	0.33934576	0.33934576	20/20	313
150	0.33311847	0.33311847	0.33311847	20/20	864
155	0.32749615	0.32749615	0.32749615	20/20	936
160	0.32224426	0.32224426	0.32224426	20/20	369
165	0.31680994	0.31680994	0.31680994	20/20	540
170	0.31178504	0.31178504	0.31178504	20/20	1372
175	0.30720611	0.30720611	0.30720611	20/20	2047
180	0.30279972	0.30275695	0.30265606	12/20	3965
185	0.29856631	0.29851061	0.29846755	6/20	3228
190	0.29441376	0.29441376	0.29441376	20/20	2820
200	0.28682205	0.28681802	0.28678181	5/20	3998

Table 2 shows that the computational results of the proposed HAMSP algorithm dominate the best-known results for the tested instances in terms of d_{best}, d_{ave} and d_{worst}. In particular, HAMSP improves the best-known result for 47 out of 50 instances, while matching the best-known result for the remaining three instances. In terms of d_{ave} and d_{worst}, our re-



(a) N = 165 (b) N = 180 (c) N = 192

(d) N = 75 (e) N = 190 (f) N = 195

(g) N = 80 (h) N = 180 (i) N = 200

Figure 2: The best solutions found for nine representative instances of the point arrangement problem in three different containers, where two points are connected by a dotted line if the distance between them is the minimum distance among all the points.

Table 4: Computational results on 25 representative instances with $75 \leq N \leq 200$ for the point arrangement problem in a convex region whose boundary is composed of three functions.

N	d_{best}	d_{ave}	d_{worst}	SR	time(s)
75	0.44065436	0.44065436	0.44065436	20/20	17
80	0.42380589	0.42380589	0.42380589	20/20	28
85	0.40871077	0.40871077	0.40871077	20/20	75
90	0.39668273	0.39668273	0.39668273	20/20	56
95	0.38471740	0.38471740	0.38471740	20/20	274
100	0.37485528	0.37485528	0.37485528	20/20	61
105	0.36559582	0.36559582	0.36559582	20/20	67
110	0.35728310	0.35728310	0.35728310	20/20	115
115	0.34936407	0.34936407	0.34936406	7/20	768
120	0.34154019	0.34154019	0.34154019	20/20	4152
125	0.33421670	0.33421670	0.33421670	20/20	88
130	0.32671203	0.32671203	0.32671203	20/20	206
135	0.31979897	0.31979897	0.31979897	20/20	106
140	0.31365217	0.31365217	0.31365217	20/20	183
145	0.30769644	0.30769644	0.30769644	20/20	146
150	0.30197125	0.30197125	0.30197125	20/20	704
155	0.29697189	0.29697189	0.29697189	20/20	388
160	0.29236891	0.29236891	0.29236891	20/20	206
165	0.28796118	0.28796118	0.28796118	20/20	1170
170	0.28357494	0.28357494	0.28357494	20/20	576
175	0.27986725	0.27986725	0.27986725	20/20	787
180	0.27505978	0.27505978	0.27505978	20/20	516
185	0.27117086	0.27117086	0.27117084	19/20	2061
190	0.26753922	0.26753922	0.26753922	20/20	575
200	0.26014522	0.26014522	0.26014522	20/20	1405

sult outperforms the best-known result respectively for 47 and 46 instances, and matches the best-known result for the remaining instances. Moreover, the success rate of the algorithm is higher than 10/20 for 39 out of the 50 instances, which means a high robustness of the algorithm. These results indicate that the proposed algorithm is very competitive compared to the previous algorithms on the point arrangement problem.

Tables 3 and 4 show that for the instances where the boundary of the container consists of several piecewise-smooth nonlinear or linear functions, the proposed algorithm also performs very well. One observes from the tables that for most instances tested, the success rate of our algorithm is 20/20 and the average computational time of the algorithm is short compared to the existing packing algorithms in the literature.

To have an intuitive impression of the best solutions found, Figure 2 shows the graphical representation of the best solutions of 9 representative instances, where for each container the best solutions of three instances are given. We observe that the distance between two neighboring points matches the minimum distance d^* between points for most points.

In summary, the experimental results of this section clearly show that the HAMSP algorithm is very efficient for solving the point arrangement problem with a convex container whose boundary is piecewise-smooth and that the performance of the algorithm is very competitive compared to the existing algorithms.

4.3 Computational results and comparison on the circle packing problem

This section aims to assess the performance of the proposed HAMSP algorithm on four sets of benchmark instances for the equal circle packing problem in a convex container, where each set corresponds to a different container. The first set consists of two subsets and contains 96 instances with a circular container, where the first subset contains 55 instances with $N \leq 300$ and the second subset contains 41 large-scale instances in the range of $301 \leq N \leq 700$.

The reason of selecting these instances as the test bed of our algorithm is that they are the hardest instances for the state-of-the-art algorithms in the literature (Lai et al., 2022; Zhou et al., 2024). For example, for the iterated dynamic thresholding search (IDTS) algorithm, which is one of the state-of-the-art circle packing algorithms (Lai et al., 2022), the success rate of hitting the best-known solution is very low (1/20) for most of these instances. The second set consists of 50 instances in the range of $151 \leq N \leq 200$, and the container is a circular quadrant. The third and fourth sets both consist of 50 instances in the range of $151 \leq N \leq 200$, and the corresponding containers are respectively a semicircle and a regular hexadecagon. For these instances, the best-known results collected from different studies are online available on the Packomania website (Specht, 2023).

The computational results of our HAMSP algorithm are summarized in Tables 5 – 9 for the four sets of instances with the same statistical information as in Table 2. It should be noted that the objective values (i.e., the minimum distance d among the points) are converted as R using a rule of $R = \frac{2}{d}$ to make a direct comparison with the best-known results in the literature (Specht, 2023). As such, the value of R corresponds to $\frac{1}{r}$ in which r is the common radius of circles, and thus a smaller R value means a better result in terms of the objective value. For most instances, the results of the proposed algorithm are compared with the best-known results in the literature due to the lack of source codes of previous algorithms and the fact that these instances have been widely tested by a number of algorithms (see (Specht, 2023) for the updating history of the best-known results).

The computational results of HAMSP on the first part of instances with a circular container are summarized in Table 5, together with the best-known results (Specht, 2023). Table 5 shows that the HAMSP algorithm performs very well and has a high performance for each considered indicator. In terms of R_{best}, HAMSP improves the best-known results for 41 out of the 55 instances and matches the best-known results for the remaining instances, which means that our algorithm has a stronger search ability compared to all the previous algorithms on these instances. Moreover, the average result of proposed HAMSP algorithm over 20 runs is superior to the best-known result for 33 out of the 55 instances, while matching the best-known results for 14 instances. In terms of R_{worst}, the HAMSP algorithm obtains a better, equal and worse result compared to the best-known result respectively for 20, 17 and 18 instances. Moreover, the success rate of the proposed algorithm is very high for most instances, which means a good robustness of the algorithm. This implies that the proposed algorithm significantly outperforms the state-of-the-art algorithms in the literature. In addition, it is worth noting that the proposed HAMSP algorithm improves on the best-known solutions for several small instances such as $N = 156$ and 197 with a perfect success rate 20/20 and very short computational time compared to that of the state-of-the-art algorithms in the literature (Lai et al., 2022; Zhou et al., 2024). This performance is impressive due to the fact that these small instances have been widely studied in the literature for a very long time and a large number of algorithms have been used to search for their optimal solutions, according to the update history of these best-known results on the Packomania website (Specht, 2023).

To make a further comparison between the HAMSP algorithm and the state-of-the-art circle packing algorithm (i.e., the geometric batch optimization (GBO) algorithm that also is a randomized heuristic algorithm (Zhou et al., 2024)) on the large-scale instances, the computational results of the HAMSP and GBO algorithms on the second part of instances with a circular container are summarized in Table 6, where the GBO algorithm ran on an Intel Xeon E5-2650 processor and its time limits were set to 12 and 24 hours respectively for the instances with $301 \leq N \leq 320$ and the instances with $N \geq 500$, which are much longer than

that of our HAMSP algorithm. The second column gives the best-known results (R^*) in the literature. The best results (R_{best}), the average results (R_{ave}) and the average computational times are given in columns 3–8 for the two algorithms. The differences between the best results of HAMSP and the best-known results ($R_{best} - R^*$) are given in the last column, where a negative value means that an improved solution is found by HAMSP. The last three rows '#Better', '#Equal', and '#Worse' indicate the numbers of instances for which the corresponding algorithm obtains a better, equal or worse result than its reference algorithm in terms of R_{best}, R_{ave} and the computational time, respectively.

Table 6 shows that the HAMSP algorithm significantly outperforms the GBO algorithm in all the considered performance indicators. In terms of R_{best}, the HAMSP algorithm obtains a better result than the GBO algorithm for 32 out of 41 instances, while matching the results of the GBO algorithm for 8 instances. In terms of R_{ave}, the HAMSP algorithm obtains a better and worse result for 39 and 2 out of 41 instances, respectively. Moreover, the computational times of the HAMSP algorithm are much shorter than those of the GBO algorithm for all instances. In addition, the last column of table shows that the HAMSP algorithm improves or matches the best-known result except for two instances.

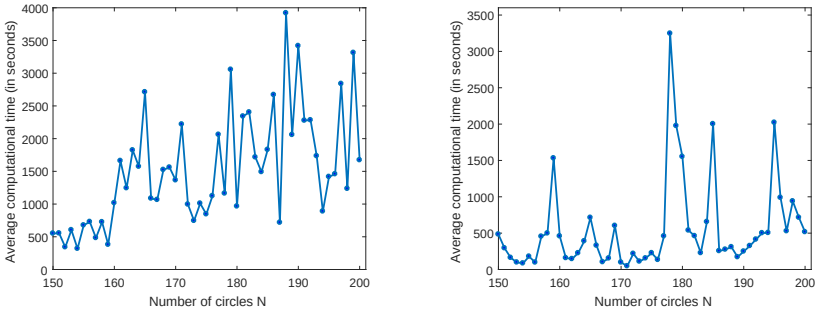
Tables 7 – 8 give the computational results of the algorithm on the second and third sets of instances. These instances are widely studied by different researchers according to the update history of the best-known solutions on the Packomania website (Specht, 2023). One can observe from the tables that the results of the proposed algorithm outperform the best-known results for almost all instances. Specifically, Table 7 shows that for the quadrant container the HAMSP algorithm improves the best-known results for 49 out of the 50 instances and matches the best-known result for the remaining instance. Furthermore, the worst result of the proposed algorithm is superior to the best-known results for 48 instances, and matches the best-known result for one instance. Moreover, the success rate of the algorithm is very high for most instances and reaches 20/20 for 41 instances. Table 8 shows that for the semicircular container the proposed algorithm performs very well too. It improves and matches the best-known results respectively for 47 and 3 out of the 50 instances. The success rate of the algorithm is 20/20 for 47 instances and is at least 12/20 for the remaining instances. The average computational times of the algorithm, which can be used to estimate the hardness of the instance, are plotted in Figure 3 respectively for the quadrant and semicircular containers. The results in Figure 3 show that the computational time of the algorithm displays some large fluctuations as the size of instance increases, which means that for the proposed algorithm the hardness of the instance depends mainly on its structural features, rather than its size.

Tables 9 shows that the proposed algorithm also performs very well for the instances with a regular hexadecagon container. These instances are intensively investigated in a recent work (Amore, 2023), and E. Specht who is the maintainer of the Packomania website (Specht, 2023) subsequently improved the best-known results for most instances. Table 9 shows that our algorithm performs very well. In particular, it improves and matches the best-known results respectively for 46 and 4 out of the 50 instances. Nevertheless, compared to the results of Table 8, the success rate of the algorithm decreases for several instances, and the computational time significantly increases for most instances, which implies that these instances are much more difficult to solve than those in Tables 7 and 8.

Figure 4 shows the graphical representation of the improved solutions of 12 representative instances with different containers.

In summary, the results of this section show that the proposed HAMSP algorithm is highly efficient for the equal circle packing problems in a variety of regular convex containers and

significantly outperforms the existing algorithms in terms of search capacity.



(a) Quadrant container (b) Semicircular container

Figure 3: Variation of the average computation time of the proposed algorithm as a function of the instance size.

Table 5: Computational results and comparison on the selected hard instances in the range of $N < 300$ for the well-known equal circle packing problem in a circular container. In terms of R_{best}, R_{ave} and R_{worst}, the improved results are indicated in bold compared to the best-known results (BKR) in the literature (Specht, 2023).

N	Best-Known (R^*)	R_{best}	R_{ave}	R_{worst}	$R_{best} - R^*$	SR	time(s)
156	13.71636305	13.71636279	13.71636279	13.71636279	-2.59E-07	20/20	1570
197	15.36749738	15.36737767	15.36737767	15.36737767	-1.20E-04	20/20	1306
206	15.73455751	15.73455751	15.73455751	15.73455751	0.0	20/20	8952
209	15.83987136	15.83987136	15.83987136	15.83987136	0.0	20/20	2778
215	16.04930621	16.04811230	16.04811230	16.04811230	-1.19E-03	20/20	5602
219	16.16915510	16.16915508	16.16915509	16.16915512	-1.76E-08	13/20	4191
221	16.25883435	16.25883435	16.25883435	16.25883435	0.0	20/20	3838
222	16.29878935	16.29868984	16.29868984	16.29868984	-9.95E-05	20/20	3967
223	16.33716660	16.33716660	16.33716660	16.33716660	0.0	20/20	887
224	16.36883169	16.36873161	16.36873161	16.36873161	-1.00E-04	20/20	863
230	16.59256499	16.59131228	16.59131228	16.59131228	-1.25E-03	20/20	4232
231	16.62906093	16.62893606	16.62893606	16.62893606	-1.25E-04	20/20	2174
233	16.69482994	16.69236002	16.69287019	16.69470787	-2.47E-03	2/20	12799
236	16.77439249	16.77439249	16.77439249	16.77439249	0.0	20/20	4481
237	16.80158286	16.80158033	16.80177814	16.80214550	-2.53E-06	13/20	8880
244	17.03480638	17.03480638	17.03480638	17.03480638	0.0	20/20	2064
250	17.26195491	17.26195483	17.26206684	17.26227118	-8.02E-08	9/20	8497
251	17.29538990	17.29538990	17.29538990	17.29538990	0.0	20/20	3431
252	17.32519590	17.32519590	17.32519590	17.32519590	0.0	20/20	2632
253	17.34595632	17.34594127	17.34612343	17.34620060	-1.51E-05	1/20	6677
254	17.39376319	17.39373178	17.39373717	17.39383945	-3.14E-05	19/20	5706
257	17.51796492	17.51796321	17.51836149	17.51938226	-1.71E-06	1/20	7926
258	17.53823578	17.53823578	17.53823578	17.53823578	0.0	20/20	7381
259	17.57395465	17.57336270	17.57377987	17.57425884	-5.92E-04	3/20	11016
261	17.62724067	17.62724067	17.62724067	17.62724067	0.0	20/20	2166
262	17.65790679	17.65746970	17.65760132	17.66010235	-4.37E-04	19/20	8309
263	17.68861384	17.68849774	17.68849774	17.68849774	-1.16E-04	20/20	6505
266	17.77608677	17.77608677	17.77608677	17.77608677	0.0	20/20	2708
267	17.79736130	17.79735106	17.79735157	17.79736130	-1.02E-05	19/20	6357
268	17.83206851	17.83187501	17.83190837	17.83208671	-1.93E-04	14/20	11358
270	17.88726567	17.88726567	17.88726567	17.88726567	0.0	20/20	6193
271	17.92969388	17.92967411	17.92967411	17.92967411	-1.98E-05	20/20	6038
274	18.03369836	18.03358504	18.03358504	18.03358504	-1.13E-04	20/20	5202
276	18.10341647	18.10341647	18.10341647	18.10341647	0.0	20/20	4714
277	18.13607946	18.13607836	18.13608058	18.13608325	-1.10E-06	10/20	8925
278	18.18055598	18.18002601	18.18012574	18.18024588	-5.30E-04	2/20	10675
279	18.21756845	18.21736923	18.21756168	18.21773521	-1.99E-04	1/20	8879
280	18.24592742	18.24592308	18.24592308	18.24592308	-4.34E-06	20/20	4057
281	18.28037613	18.28020456	18.28028991	18.28039300	-1.72E-04	3/20	10778
282	18.30878994	18.30727366	18.30809173	18.30929487	-1.52E-03	8/20	12341
283	18.34023658	18.34008331	18.34027483	18.34076016	-1.53E-04	11/20	11067
284	18.35945664	18.35942819	18.35953414	18.36117326	-2.85E-05	18/20	10407
285	18.40267860	18.40071061	18.40122240	18.40260402	-1.97E-03	8/20	11641
286	18.42924176	18.42924174	18.42924174	18.42924174	-1.96E-08	20/20	3644
287	18.46780004	18.46743979	18.46748244	18.46829272	-3.60E-04	19/20	8048
288	18.49434502	18.49428619	18.49428619	18.49428619	-5.88E-05	20/20	6817
289	18.51153892	18.51148362	18.51148362	18.51148362	-5.53E-05	20/20	9068
290	18.54870759	18.54862498	18.54862498	18.54862498	-8.26E-05	20/20	9384
291	18.56655121	18.56604139	18.56604139	18.56604139	-5.10E-04	20/20	6564
292	18.59450109	18.59433631	18.59440060	18.59454741	-1.65E-04	10/20	8005
293	18.62336670	18.62278847	18.62338391	18.62430519	-5.78E-04	4/20	11150
294	18.64484970	18.64427425	18.64427827	18.64429811	-5.75E-04	15/20	9956
295	18.65521752	18.65521752	18.65521752	18.65521752	0.0	20/20	6667
296	18.70275001	18.70257531	18.70257531	18.70257531	-1.75E-04	20/20	3912
297	18.72977481	18.72977465	18.72977475	18.72977481	-1.53E-07	7/20	5500
299	18.78542781	18.78525576	18.78525576	18.78525576	-1.72E-04	20/20	3045
#Improve		41	33	20			
#Equal		14	14	17			
#Worse		0	8	18			

Table 6: Comparison between the HAMSP method and the state-of-the-art algorithm (i.e., the GBO algorithm (Zhou et al., 2024)) on 41 selected large-scale instances, where the better results between the two compared algorithms are indicated for each performance indicator. The best-known results (R^*) are taken from (Specht, 2023) and (Zhou et al., 2024)

N	Best-Known (R^*)	R_{best}		R_{ave}		$time(s)$		HAMSP
		GBO	HAMSP	GBO	HAMSP	GBO	HAMSP	$R_{best} - R^*$
301	18.84346351	18.84346351	18.84346351	18.84355108	18.84346351	38618	3563	0.0
302	18.89178160	18.89178226	18.89178160	18.89203331	18.89178160	23987	7170	0.0
303	18.92974915	18.92975062	18.92970239	18.93032872	18.92970239	23652	5917	-4.68E-05
304	18.96362032	18.96362032	18.96273958	18.96466401	18.96307327	18852	10726	-8.81E-04
305	19.00169315	19.00172681	19.00123339	19.00273469	19.00135969	21922	12136	-4.60E-04
306	19.03038941	19.03065124	19.03038941	19.03139172	19.03038941	25226	5134	0.0
307	19.06016092	19.06084192	19.05981236	19.06195543	19.05981236	26981	3259	-3.49E-04
308	19.10499144	19.10723995	19.10468186	19.10982061	19.10484180	23291	8151	-3.10E-04
309	19.14133583	19.14262504	19.14132915	19.14362043	19.14132915	24278	5045	-6.68E-06
310	19.17841932	19.17841932	19.17791232	19.17944284	19.17791453	24718	8767	-5.07E-04
311	19.21056407	19.21056407	19.21052802	19.21256178	19.21104031	21153	12815	-3.61E-05
312	19.23358565	19.23358565	19.23358565	19.23452950	19.23361006	30386	10734	0.0
313	19.25699466	19.25710301	19.25699466	19.25828441	19.25705900	25457	10768	0.0
314	19.28619024	19.28619024	19.28619024	19.28647695	19.28624867	26386	10275	0.0
315	19.30227399	19.30227399	19.30227399	19.30228618	19.30227399	24488	4149	0.0
316	19.33404175	19.33404175	19.33404175	19.33458428	19.33405120	22840	9388	0.0
317	19.36759567	19.36759567	19.36759567	19.36787112	19.36759567	23129	6674	0.0
318	19.39156609	19.39156609	19.39156609	19.39163157	19.39156609	27084	3693	0.0
319	19.42427783	19.42427783	19.42427783	19.42531087	19.42427783	24782	5343	0.0
320	19.45158374	19.45173418	19.45120573	19.45445553	19.45120573	30039	11018	-3.78E-04
500	24.13125294	24.13125294	24.13096092	24.13137882	24.13123427	39561	12725	-2.92E-04
510	24.42105376	24.42105376	24.41401851	24.42491359	24.41690342	60259	13460	-7.04E-03
520	24.62580737	24.62580737	24.62378188	24.63141241	24.62496543	69426	15284	-2.03E-03
530	24.84552802	24.84552802	24.84401862	24.84726030	24.84567511	44648	13646	-1.51E-03
540	25.08559112	25.08559112	25.08223707	25.08771273	25.08505206	45255	15157	-3.35E-03
550	25.33425357	25.33425357	25.33122417	25.33576287	25.33231313	43783	13416	-3.03E-03
560	25.51224249	25.51224249	25.51215422	25.51467968	25.51445402	59827	17570	-8.83E-05
570	25.71348473	25.71348473	25.71341665	25.71385424	25.71390697	63852	15087	-6.81E-05
580	25.95110749	25.95110749	25.94749220	25.95273796	25.94860967	69140	16749	-3.62E-03
590	26.20250416	26.20250416	26.20034819	26.20691566	26.20496190	62246	15667	-2.16E-03
600	26.41697463	26.41768801	26.41565986	26.42168302	26.41693481	58273	15676	-1.31E-03
610	26.62277366	26.62277366	26.61661750	26.62561510	26.61977506	55976	18057	-6.16E-03
620	26.84158994	26.84312357	26.83650030	26.84576525	26.83915474	65063	17639	-5.09E-03
630	27.03400365	27.05257789	27.04637089	27.06185882	27.05799715	44900	19136	1.24E-02
640	27.23872827	27.23872827	27.23672227	27.24218949	27.23974319	59358	16687	-2.01E-03
650	27.43822860	27.43822860	27.43780972	27.44220589	27.43930454	71570	17066	-4.19E-04
660	27.65890939	27.65890939	27.65624913	27.66222253	27.65903910	69611	16314	-2.66E-03
670	27.90082150	27.90082150	27.89106316	27.90309048	27.89881537	65422	16112	-9.76E-03
680	28.08775360	28.08775360	28.08231895	28.09039650	28.08596757	63392	16352	-5.43E-03
690	28.24471863	28.24471863	28.25947460	28.26387873	28.27520876	55853	19037	1.48E-02
700	28.48398886	28.48398886	28.47465411	28.48776809	28.48224102	59161	18351	-9.33E-03
#Better		1	32	2	39	0	41	
#Equal		8	8	0	0	0	0	
#Worse		32	1	39	2	41	0	

Table 7: Computational results and comparison on 50 representative instances with $151 \leq N \leq 200$ for the circle packing problem in a circular quadrant. In terms of R_{best}, R_{ave} and R_{worst}, the improved results are indicated in bold compared to the best-known results (BKR) in the literature (Specht, 2023).

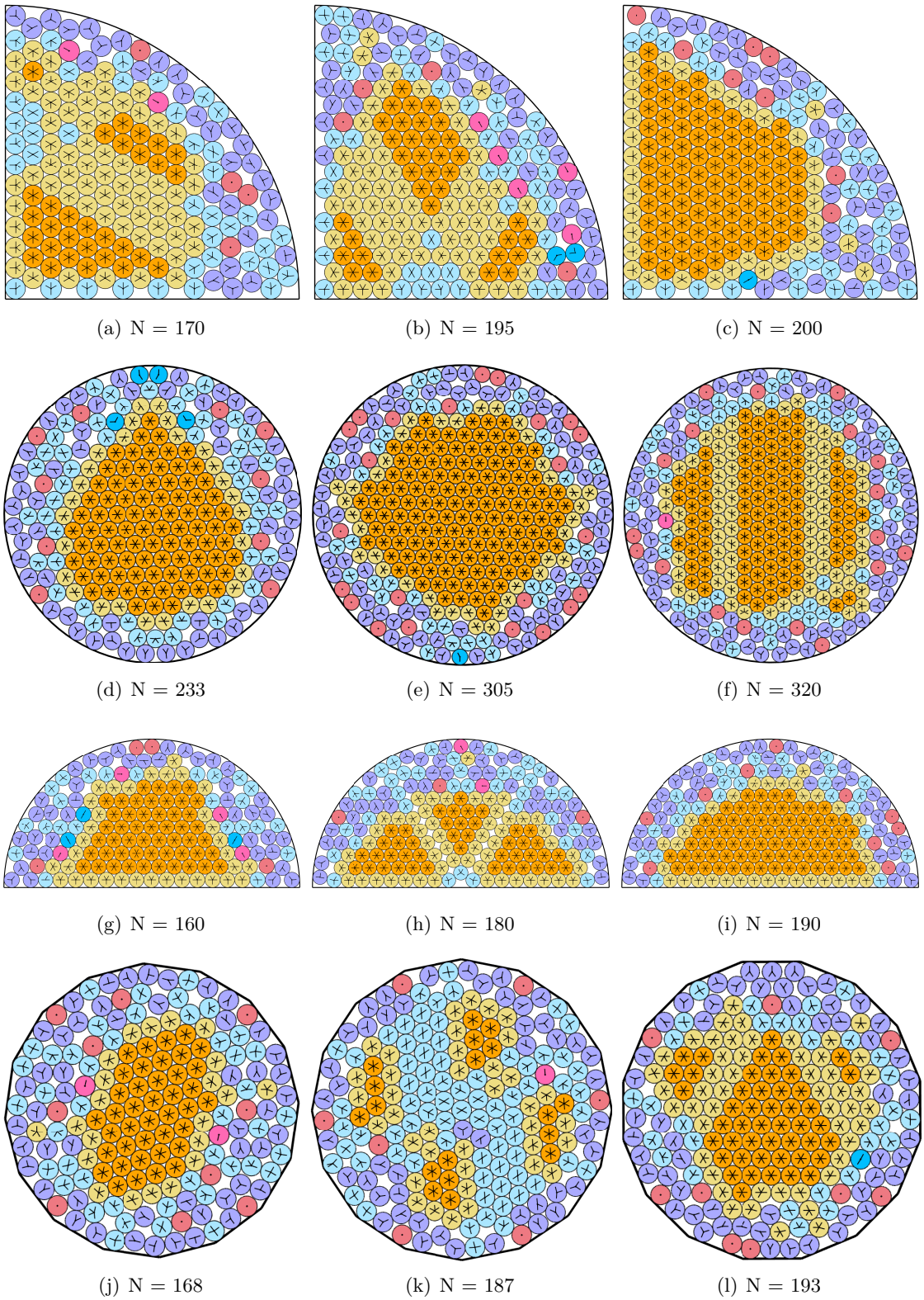
N	Best-Known (R^*)	R_{best}	R_{ave}	R_{worst}	$R_{best} - R^*$	SR	time(s)
151	26.88099748	26.88099650	26.88099650	26.88099650	-9.76E-07	20/20	554
152	26.98525902	26.97209464	26.97209464	26.97209464	-1.32E-02	20/20	341
153	27.05193768	27.03877679	27.03877679	27.03877679	-1.32E-02	20/20	603
154	27.12615743	27.12605925	27.12605925	27.12605925	-9.82E-05	20/20	318
155	27.20731019	27.19736572	27.19736572	27.19736572	-9.94E-03	20/20	676
156	27.33076708	27.32756819	27.32756819	27.32756819	-3.20E-03	20/20	728
157	27.44115885	27.42921417	27.42921417	27.42921417	-1.19E-02	20/20	482
158	27.52805142	27.51518692	27.51518692	27.51518692	-1.29E-02	20/20	725
159	27.61317610	27.60997397	27.60997397	27.60997397	-3.20E-03	20/20	380
160	27.68020720	27.67681210	27.67681210	27.67681210	-3.40E-03	20/20	1016
161	27.76710059	27.75873859	27.75873863	27.75873867	-8.36E-03	14/20	1660
162	27.85308885	27.85152800	27.85152800	27.85152800	-1.56E-03	20/20	1244
163	27.93802281	27.93472827	27.93472827	27.93472827	-3.29E-03	20/20	1823
164	28.01892441	28.00821229	28.00821229	28.00821229	-1.07E-02	20/20	1573
165	28.07820011	28.07689445	28.07689445	28.07689445	-1.31E-03	20/20	2711
166	28.13692699	28.13412529	28.13412529	28.13412529	-2.80E-03	20/20	1086
167	28.25175477	28.25170534	28.25170534	28.25170534	-4.94E-05	20/20	1065
168	28.33962115	28.33165430	28.33165430	28.33165430	-7.97E-03	20/20	1523
169	28.39994391	28.38450707	28.38450707	28.38450711	-1.54E-02	20/20	1560
170	28.46447993	28.46103505	28.46103505	28.46103505	-3.44E-03	20/20	1365
171	28.55117566	28.54989072	28.54989072	28.54989072	-1.28E-03	20/20	2219
172	28.62189446	28.62189446	28.62189446	28.62189446	0.0	20/20	995
173	28.70363885	28.70040562	28.70040562	28.70040562	-3.23E-03	20/20	745
174	28.79791377	28.79128943	28.79128943	28.79128943	-6.62E-03	20/20	1009
175	28.89832171	28.88487441	28.88487465	28.88487473	-1.34E-02	1/20	845
176	28.94798884	28.94032832	28.94032832	28.94032832	-7.66E-03	20/20	1123
177	29.04805340	29.04274053	29.04274053	29.04274053	-5.31E-03	20/20	2061
178	29.12143989	29.11064185	29.11064185	29.11064185	-1.08E-02	20/20	1162
179	29.22334274	29.19603528	29.19604503	29.19608405	-2.73E-02	16/20	3055
180	29.29833508	29.29626399	29.29626399	29.29626399	-2.07E-03	20/20	965
181	29.36945074	29.35840508	29.35840508	29.35840508	-1.10E-02	20/20	2340
182	29.46466038	29.45094711	29.45094711	29.45094711	-1.37E-02	20/20	2403
183	29.53271084	29.53271043	29.53271043	29.53271044	-4.05E-07	20/20	1715
184	29.59954138	29.59420367	29.59420367	29.59420367	-5.34E-03	20/20	1491
185	29.68283028	29.67814732	29.67814732	29.67814732	-4.68E-03	20/20	1830
186	29.77478130	29.76016486	29.76152952	29.77694134	-1.46E-02	17/20	2669
187	29.86148198	29.84738261	29.84738261	29.84738261	-1.41E-02	20/20	717
188	29.94923359	29.92043832	29.92277541	29.93202812	-2.88E-02	9/20	3919
189	30.01368591	30.00905944	30.00945286	30.01070396	-4.63E-03	15/20	2058
190	30.06775827	30.06335485	30.06403512	30.06764616	-4.40E-03	4/20	3417
191	30.09376814	30.09274593	30.09274593	30.09274593	-1.02E-03	20/20	2276
192	30.17107846	30.16981048	30.16981048	30.16981048	-1.27E-03	20/20	2283
193	30.26593359	30.26325469	30.26325469	30.26325469	-2.68E-03	20/20	1734
194	30.29458525	30.28990478	30.28990478	30.28990478	-4.68E-03	20/20	889
195	30.37795325	30.37603818	30.37603818	30.37603818	-1.92E-03	20/20	1416
196	30.47206849	30.47200511	30.47200511	30.47200511	-6.34E-05	20/20	1458
197	30.53231551	30.53100816	30.53101519	30.53102824	-1.31E-03	13/20	2838
198	30.65507058	30.64184993	30.64184993	30.64184993	-1.32E-02	20/20	1235
199	30.76162418	30.74562161	30.74568260	30.74623156	-1.60E-02	18/20	3312
200	30.84866371	30.83073970	30.83073970	30.83073970	-1.79E-02	20/20	1672
#Improve		49	49	48			
#Equal		1	1	1			
#Worse		0	0	1			

Table 8: Computational results and comparison on 50 representative instances with $151 \leq N \leq 200$ for the circle packing problem in a semicircular container. In terms of R_{best}, R_{ave} and R_{worst}, the improved results are indicated in bold compared to the best-known results (BKR) in the literature (Specht, 2023).

N	Best-Known (R^*)	R_{best}	R_{ave}	R_{worst}	$R_{best} - R^*$	SR	time(s)
151	19.02323592	19.02002407	19.02002407	19.02002407	-3.21E-03	20/20	294
152	19.06265540	19.06221437	19.06221437	19.06221437	-4.41E-04	20/20	162
153	19.12897454	19.12897454	19.12897454	19.12897454	0.0	20/20	98
154	19.19521364	19.19521364	19.19521364	19.19521364	0.0	20/20	85
155	19.25078841	19.24697435	19.24697435	19.24697435	-3.81E-03	20/20	179
156	19.30345692	19.29849436	19.29849436	19.29849436	-4.96E-03	20/20	97
157	19.35632456	19.34791907	19.34791907	19.34791907	-8.41E-03	20/20	454
158	19.41878265	19.41452997	19.41452997	19.41452997	-4.25E-03	20/20	498
159	19.47975867	19.47281127	19.47281125	19.47281128	-6.95E-03	20/20	1531
160	19.52284608	19.51547755	19.51547755	19.51547755	-7.37E-03	20/20	458
161	19.59446429	19.58799292	19.58799292	19.58799292	-6.47E-03	20/20	158
162	19.65555020	19.65399771	19.65399771	19.65399771	-1.55E-03	20/20	145
163	19.72570612	19.72416358	19.72416358	19.72416358	-1.54E-03	20/20	226
164	19.80858461	19.79489897	19.79489897	19.79489897	-1.37E-02	20/20	390
165	19.85154071	19.83674815	19.83674815	19.83674815	-1.48E-02	20/20	713
166	19.90811731	19.89775064	19.89775064	19.89775064	-1.04E-02	20/20	330
167	19.93789457	19.92693560	19.92693560	19.92693560	-1.10E-02	20/20	103
168	19.98453918	19.98439486	19.98439486	19.98439486	-1.44E-04	20/20	154
169	20.02459782	20.01939596	20.01939596	20.01939596	-5.20E-03	20/20	602
170	20.07169903	20.06926724	20.06926724	20.06926724	-2.43E-03	20/20	98
171	20.14172559	20.14172559	20.14172559	20.14172559	0.0	20/20	47
172	20.21795883	20.21213938	20.21213938	20.21213938	-5.82E-03	20/20	217
173	20.27787466	20.27665458	20.27665458	20.27665458	-1.22E-03	20/20	110
174	20.36530505	20.35767251	20.35767251	20.35767251	-7.63E-03	20/20	155
175	20.43874997	20.43105887	20.43105887	20.43105887	-7.69E-03	20/20	226
176	20.51716646	20.49078395	20.49078395	20.49078395	-2.64E-02	20/20	133
177	20.58943015	20.56707995	20.56707995	20.56707995	-2.24E-02	20/20	458
178	20.63989907	20.61560032	20.61562639	20.61612189	-2.43E-02	19/20	3246
179	20.67518227	20.67170894	20.67170894	20.67170894	-3.47E-03	20/20	1976
180	20.73853303	20.71613996	20.71711807	20.72592521	-2.24E-02	18/20	1551
181	20.77161098	20.76582985	20.76582985	20.76582985	-5.78E-03	20/20	538
182	20.81550922	20.80883762	20.80883762	20.80883762	-6.67E-03	20/20	461
183	20.87866550	20.87210569	20.87210569	20.87210569	-6.56E-03	20/20	227
184	20.91621379	20.90735959	20.90735959	20.90735959	-8.85E-03	20/20	655
185	20.95908902	20.95902727	20.95902727	20.95902727	-6.17E-05	20/20	2001
186	20.99553967	20.99487256	20.99487256	20.99487256	-6.67E-04	20/20	255
187	21.07051412	21.06701043	21.06701043	21.06701043	-3.50E-03	20/20	275
188	21.10907698	21.10720379	21.10720379	21.10720379	-1.87E-03	20/20	309
189	21.16865010	21.16333789	21.16333789	21.16333789	-5.31E-03	20/20	171
190	21.21639955	21.20971376	21.20971376	21.20971376	-6.69E-03	20/20	249
191	21.28735771	21.27027494	21.27027494	21.27027494	-1.71E-02	20/20	324
192	21.35886586	21.34165308	21.34165308	21.34165308	-1.72E-02	20/20	413
193	21.42376435	21.39944050	21.39944050	21.39944050	-2.43E-02	20/20	501
194	21.47454536	21.45598547	21.45598547	21.45598547	-1.86E-02	20/20	504
195	21.52404840	21.51619827	21.51619831	21.51619836	-7.85E-03	12/20	2021
196	21.56967305	21.56910957	21.56910957	21.56910957	-5.63E-04	20/20	988
197	21.62871647	21.62489082	21.62489082	21.62489082	-3.83E-03	20/20	528
198	21.68800177	21.67889729	21.67889729	21.67889729	-9.10E-03	20/20	939
199	21.73383636	21.72626039	21.72626039	21.72626039	-7.58E-03	20/20	715
200	21.77435759	21.77008392	21.77008392	21.77008392	-4.27E-03	20/20	516
#Improve		47	47	47			
#Equal		3	3	3			
#Worse		0	0	0			

Table 9: Computational results and comparison on 50 representative instances with $151 \leq N \leq 200$ for the circle packing problem in a regular hexadecagon. In terms of R_{best}, R_{ave} and R_{worst}, the improved results are indicated in bold compared to the best-known results (BKR) in the literature (Specht, 2023).

N	Best-Known (R^*)	R_{best}	R_{ave}	R_{worst}	$R_{best} - R^*$	SR	time(s)
151	13.67618656	13.67618656	13.67618656	13.67618656	0.0	20/20	1894
152	13.73253530	13.72590253	13.72590253	13.72590253	-6.63E-03	20/20	1854
153	13.77397972	13.76798264	13.76798264	13.76798264	-6.00E-03	20/20	1329
154	13.81635827	13.81494554	13.81494554	13.81494554	-1.41E-03	20/20	1790
155	13.85630968	13.85197084	13.85197084	13.85197084	-4.34E-03	20/20	2672
156	13.89607948	13.89201271	13.89201271	13.89201271	-4.07E-03	20/20	1616
157	13.94391511	13.93859447	13.93859596	13.93862419	-5.32E-03	19/20	2260
158	13.98397872	13.97810571	13.97810571	13.97810571	-5.87E-03	20/20	757
159	14.04592114	14.04370266	14.04375079	14.04382297	-2.22E-03	12/20	2693
160	14.09100872	14.08885285	14.08885285	14.08885285	-2.16E-03	20/20	544
161	14.14138067	14.13438254	14.13438254	14.13438254	-7.00E-03	20/20	2464
162	14.19981231	14.18862625	14.18912581	14.19429236	-1.12E-02	18/20	4104
163	14.24084739	14.23552782	14.23555921	14.23584162	-5.32E-03	16/20	4020
164	14.26205512	14.26195123	14.26195123	14.26195123	-1.04E-04	20/20	1857
165	14.31510039	14.30981306	14.30981306	14.30981306	-5.29E-03	20/20	3133
166	14.35034787	14.34398777	14.34529385	14.35100780	-6.36E-03	11/20	4313
167	14.38748643	14.38629573	14.38634388	14.38725873	-1.19E-03	19/20	4111
168	14.42660836	14.42660836	14.42688729	14.43218908	0.0	19/20	2725
169	14.46993030	14.46522926	14.46522926	14.46522926	-4.70E-03	20/20	1213
170	14.51016657	14.50713079	14.50713079	14.50713079	-3.04E-03	20/20	2061
171	14.54120514	14.53854478	14.53854478	14.53854478	-2.66E-03	20/20	1758
172	14.58646525	14.58482768	14.58482768	14.58482768	-1.64E-03	20/20	2051
173	14.63403212	14.62806564	14.62806564	14.62806564	-5.97E-03	20/20	2072
174	14.67419569	14.67333554	14.67333554	14.67333554	-8.60E-04	20/20	2479
175	14.71532571	14.71532571	14.71532571	14.71532571	0.0	20/20	1725
176	14.74655587	14.74421399	14.74421399	14.74421399	-2.34E-03	20/20	1777
177	14.78655948	14.78593783	14.78593783	14.78593783	-6.22E-04	20/20	1887
178	14.81828379	14.81574646	14.81574646	14.81574646	-2.54E-03	20/20	1252
179	14.86357121	14.86235043	14.86235043	14.86235043	-1.22E-03	20/20	1147
180	14.90383558	14.90318314	14.90318314	14.90318314	-6.52E-04	20/20	2182
181	14.93492487	14.93464881	14.93609123	14.94102909	-2.76E-04	6/20	4770
182	14.96788416	14.96741213	14.96741213	14.96741213	-4.72E-04	20/20	1445
183	15.01512999	15.01430450	15.01430450	15.01430450	-8.25E-04	20/20	1424
184	15.08030005	15.06994998	15.07051651	15.07242450	-1.04E-02	12/20	3771
185	15.12527995	15.11321619	15.11359850	15.11470166	-1.21E-02	13/20	3405
186	15.15833014	15.15648427	15.15661522	15.15833014	-1.85E-03	16/20	4457
187	15.19243873	15.19086317	15.19251845	15.19670624	-1.58E-03	2/20	5347
188	15.23751318	15.22914287	15.23008216	15.23262453	-8.37E-03	7/20	5835
189	15.25734952	15.25273776	15.25301365	15.25430693	-4.61E-03	15/20	4301
190	15.28931004	15.28772107	15.28772107	15.28772107	-1.59E-03	20/20	2328
191	15.31991771	15.31932982	15.31934376	15.31960860	-5.88E-04	19/20	2912
192	15.36584005	15.36519642	15.36521146	15.36529671	-6.44E-04	17/20	3878
193	15.40042766	15.39963616	15.39963616	15.39963616	-7.92E-04	20/20	3837
194	15.43981814	15.43914722	15.43914722	15.43914722	-6.71E-04	20/20	2561
195	15.48373234	15.48168862	15.48168862	15.48168862	-2.04E-03	20/20	2072
196	15.52705499	15.52404128	15.52407856	15.52441036	-3.01E-03	17/20	3194
197	15.56150329	15.55794670	15.55794914	15.55799555	-3.56E-03	19/20	3089
198	15.60368610	15.59902157	15.59902446	15.59905321	-4.66E-03	2/20	4122
199	15.62855640	15.62855640	15.62855640	15.62855640	0.0	20/20	3812
200	15.68010718	15.67975636	15.67980896	15.68060061	-3.51E-04	17/20	4670
#Improve		46	44	41			
#Equal		4	3	4			
#Worse		0	3	5			



(a) N = 170 (b) N = 195 (c) N = 200

(d) N = 233 (e) N = 305 (f) N = 320

(g) N = 160 (h) N = 180 (i) N = 190

(j) N = 168 (k) N = 187 (l) N = 193

Figure 4: The improved solutions of 12 representative instances for the equal circle packing problems in four different containers, i.e., a circular quadrant, a circle, a semicircle and a regular hexadecagon, where the circles are colored according to their number of neighbors.

5 Analysis of the Key Algorithmic Components

In this section, we turn to an analysis and discussion of two essential components of the proposed algorithm, i.e., the optima exploitation based adjustment method to adjust the minimum distance between points and the DIH method to optimize the subproblems.

5.1 Effectiveness of the optima exploitation based adjustment method

Table 10: Comparison between the OEB adjustment method and the standard SUMT adjustment method on a well-known equal circle packing problem (i.e., the problem of packing equal circles in a circular container), where the better results between two compared methods are indicated in bold.

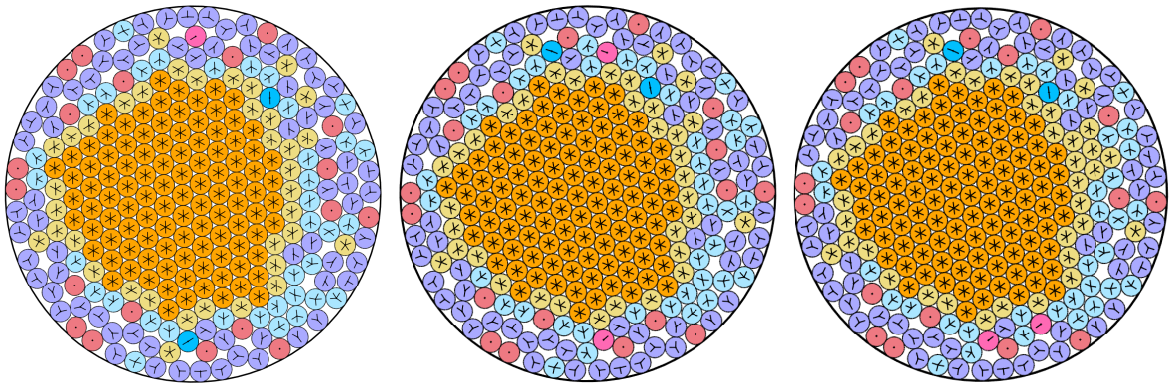
	R_{best}		R_{avg}		SR		$time(s)$	
N	HAMSP*	HAMSP	HAMSP*	HAMSP	HAMSP*	HAMSP	HAMSP*	HAMSP
262	17.65746970	17.65746970	17.65847621	17.65760132	1/20	19/20	11934	8309
267	17.79735106	17.79735106	17.79739701	17.79735157	2/20	19/20	9813	6357
268	17.83196595	17.83187501	17.83204077	17.83190837	0/20	14/20	7253	11358
271	17.92969125	17.92967411	17.92970886	17.92967411	0/20	20/20	12154	6038
274	18.03371525	18.03358504	18.03384668	18.03358504	0/20	20/20	9067	5202
277	18.13607836	18.13607836	18.13689254	18.13608058	1/20	10/20	11584	8925
278	18.18010080	18.18002601	18.18088264	18.18012574	0/20	2/20	9593	10675
279	18.21757827	18.21736923	18.21778483	18.21756168	0/20	1/20	7812	8879
280	18.24592742	18.24592308	18.24595821	18.24592308	0/20	20/20	9742	4057
281	18.28031024	18.28020456	18.28041047	18.28028991	0/20	3/20	10652	10778
282	18.30734603	18.30727366	18.30885008	18.30809173	0/20	8/20	13995	12341
283	18.34027382	18.34008331	18.34085322	18.34027483	0/20	11/20	11579	11067
284	18.35942819	18.35942819	18.36078644	18.35953414	1/20	18/20	16129	10407
285	18.40071061	18.40071061	18.40220652	18.40122240	1/20	8/20	14617	11641
286	18.42924174	18.42924174	18.42999250	18.42924174	7/20	20/20	10961	3644
287	18.46743979	18.46743979	18.46776284	18.46748244	5/20	19/20	11019	8048
288	18.49428619	18.49428619	18.49433566	18.49428619	3/20	20/20	13948	6817
289	18.51148362	18.51148362	18.51151638	18.51148362	8/20	20/20	13224	9068
290	18.54862498	18.54862498	18.54880632	18.54862498	1/20	20/20	8295	9384
291	18.56604139	18.56604139	18.56617971	18.56604139	8/20	20/20	13576	6564
292	18.59440852	18.59433631	18.59459066	18.59440060	0/20	10/20	13258	8005
293	18.62320045	18.62278847	18.62379224	18.62338391	0/20	4/20	9653	11150
294	18.64472770	18.64427425	18.64495967	18.64427827	0/20	15/20	13550	9956
303	18.92974929	18.92970239	18.92990859	18.92970239	0/20	20/20	13439	5917
304	18.96389896	18.96273958	18.96441090	18.96307327	0/20	4/20	12387	10726
305	19.00124350	19.00123339	19.00172921	19.00135969	0/20	15/20	11067	12136
307	19.06016876	19.05981236	19.06088124	19.05981236	0/20	20/20	9690	3259
308	19.10678912	19.10468186	19.10782138	19.10484180	0/20	15/20	12106	8151
309	19.14133079	19.14132915	19.14235732	19.14132915	0/20	20/20	14606	5045
310	19.17800359	19.17791232	19.17900317	19.17791453	0/20	17/20	14777	8767
311	19.21066978	19.21052802	19.21151914	19.21104031	0/20	4/20	14835	12815
320	19.45158294	19.45120573	19.45173284	19.45120573	0/20	20/20	10131	11018
#Better	0	21	0	32	0	32	8	24
#Equal	11	11	0	0	0	0	0	0
#Worse	21	0	32	0	32	0	24	8

The optima exploitation based (OEB) adjustment method is one of the main components of the HAMSP algorithm, whose goal is to maximize the common circle radius or the minimum distance between the points. To check its effectiveness, we carried out an experiment based on 32 selected hard instances with respect to the well-known equal circle packing problem in a circular container. It should be noted that these instances are very hard to solve for the state-of-art algorithms according to the update history of the best-known solutions reported on the Packomania website (Specht, 2023). In this experiment, we first created a

variant HAMSP* of the HAMSP algorithm by replacing the OEB adjustment method with the standard SUMT method that is shown to be the best performing adjustment method for the circle packing problem (Lai et al., 2022), while keeping other components of the algorithm unchanged. Then, the HAMSP* and HAMSP algorithms were respectively run 20 times with different random seeds to solver each of the 32 selected instances. The experimental results of the two algorithms are summarized in Table 10, including the best objective value R_{best} ($= 2.0/d_{best}$) over 20 runs, the average objective value (R_{avg}), the success rate of hitting the current best solutions (SR), and the average computational time ($time(s)$) in seconds to reach the final result. The better results between the two compared algorithms are indicated in bold. Moreover, the last three rows '#Better', '#Equal' and '#Worse' respectively indicate the numbers of instances for which the corresponding algorithm obtains a better, equal or worse result for the corresponding performance indicator.

Table 10 shows that the HAMSP algorithm significantly outperforms the HAMSP* algorithm for each performance indicator. For R_{best}, the HAMSP algorithm obtains a better result on 21 out of the 32 instances, while matching the results of HAMSP* for the remaining instances. In terms of both R_{ave} and success rate, HAMSP outperforms HAMSP* on all the instances. In terms of computational time, HAMSP obtains a better and worse result for 24 and 8 instances, respectively. The outcome of this experiment clearly shows that for these hard instances the OEB adjustment method proposed in this work significantly outperforms the standard SUMT method, clearly showing the effectiveness of the OEB adjustment method.

In addition, to have an adequate understanding for the reason why the OEB adjustment method significantly outperforms the standard SUMT method for these hard instances, we carried out a case study based on an representative instance with $N = 271$. In the case study, to observe the differences between the solutions returned by the SUMT method, we ran HAMSP* several times and collected a number of solutions returned by the SUMT method in the search process of the algorithm. Figure 5 gives the packing configurations of three different local optimal solutions collected.



(a) $R = 17.92969388$ (b) $R = 17.92969125$ (c) $R = 17.92967411$

Figure 5: The packing configurations of three different local optimal solutions returned by the SUMT method for the problem of packing $N = 271$ circles in a circular container, where the circles are colored according to their number of neighbors.

One observes from Figure 5 that despite having different objective values, the geometrical

configurations of three local optimal solutions are very similar such that it is not easy for the naked eyes to observe the differences between them. This observation indicates a very interesting phenomenon that for the circle packing problems there exist a large number of different local optimal solutions or saddle points sharing very similar geometrical configurations, which largely increases the difficulty of global optimization. Furthermore, it should be pointed out that this is a very common phenomenon especially for the hard instances according to our computational experiments.

Due to this phenomenon, the adjustment methods aiming at local optimization are doomed to be ineffective for the hard instances, which explains why the existing adjustment methods like the standard SUMT method (Lai et al., 2022) and the binary search method (Huang and Ye, 2011) perform poorly for these hard instances. Specifically, the adjustment methods aiming at local optimization return only one local optimal solution at each time, and thus often fail to reach high-quality packing configurations among a large number of similar ones, thus limiting largely their search capacity. However, compared to these local optimization approaches, our OEB adjustment method can be regarded as a global optimization approach. In particular, it consists of the standard SUMT method and a MBH method with multi-scale perturbations ($Weak\_MBH$ in Algorithm 4), where the MBH method is a global optimization method aiming to find a high-quality feasible solution by jumping from one local optimal solution to another one and the SUMT method is a local optimization method aiming to locally improve the solution returned by the MBH method. Thus, the combined use of the SUMT and MBH methods allows the OEB adjustment method to visit a large number of local minimum solutions with very similar packing configurations and find high-quality local minimum solutions.

5.2 Effectiveness of the diversification and intensification based heuristic method

The diversification and intensification based heuristic (DIH) method used to optimize the unconstrained function $E_d(X)$ (defined in Section 2.1) is one of main components of proposed HAMSP algorithm, and its goal is to reach a desirable tradeoff between the intensification and diversification search by alternately performing the MBH method and a stochastic diversification procedure. To show its effectiveness, we carried out another experiment based on 32 representative instances mentioned above. In this experiment, we first created two variants HAMSP$_1$ and HAMSP$_2$ of HAMSP algorithm by replacing the DIH method with the MBH method with different search depths, respectively. In HAMSP$_1$, the DIH method is replaced by the MBH method with a small search depth of $MaxNoImprove = 20$, and thus HAMSP$_1$ has a weak intensification search ability and a strong diversification search ability compared to the HAMSP algorithm. On the contrary, in HAMSP$_2$, the DIH method is replaced by the MBH method with a large search depth of $MaxNoImprove = 200$, which leads to a strong intensification search ability of the algorithm. Then, the HAMSP$_1$, HAMSP$_2$ and HAMSP algorithms were independently run 20 times on each instance, respectively, and the computational results are summarized in Table 11, where '#Best' denotes the numbers of instances for which the corresponding algorithm obtains the best result among the compared algorithms and other symbols are the same as in Table 10.

We observe from Table 11 that the HAMSP algorithm significantly outperforms the HAMSP$_1$ and HAMSP$_2$ algorithms on all the considered performance indicators. Specifically, in terms of R_{ave}, HAMSP, HAMSP$_1$ and HAMSP$_2$ respectively obtain the best results on 27, 0 and 9 instances. In terms of the success rate of hitting the current best-known solu-

Table 11: Comparison of HAMSP_1, HAMSP_2 and HAMSP in terms of R_{ave}, success rate (SR) to hit the current best-known solution, and average computational time ($time(s)$). The best results among the three algorithms are indicated in bold in terms of R_{ave}, SR and the computational time.

	R_{ave}			SR			$time(s)$		
N	HAMSP_1	HAMSP_2	HAMSP	HAMSP_1	HAMSP_2	HAMSP	HAMSP_1	HAMSP_2	HAMSP
262	17.65960785	17.65773293	17.65760132	6/20	9/20	19/20	12868	12062	8309
267	17.79735669	17.79735311	17.79735157	9/20	8/20	19/20	9052	8977	6357
268	17.83209481	17.83189848	17.83190837	0/20	7/20	14/20	5559	13115	11358
271	17.92971492	17.92967411	17.92967411	12/20	20/20	20/20	7864	6179	6038
274	18.03360854	18.03358504	18.03358504	17/20	20/20	20/20	6762	6904	5202
277	18.13608830	18.13609301	18.13608058	4/20	3/20	10/20	7060	9746	8925
278	18.18015389	18.18020441	18.18012574	0/20	0/20	2/20	9734	13029	10675
279	18.21764200	18.21759286	18.21756168	1/20	1/20	1/20	7976	9799	8879
280	18.24592330	18.24592308	18.24592308	19/20	20/20	20/20	8345	9697	4057
281	18.28034950	18.28031514	18.28028991	0/20	1/20	3/20	13633	13165	10778
282	18.30970663	18.30868923	18.30809173	0/20	5/20	8/20	9292	12983	12341
283	18.34045670	18.34028669	18.34027483	6/20	11/20	11/20	9981	8728	11067
284	18.36077337	18.36089674	18.35953414	9/20	7/20	18/20	14497	16012	10407
285	18.40216457	18.40302101	18.40122240	3/20	2/20	8/20	12810	12059	11641
286	18.42930425	18.42945027	18.42924174	19/20	19/20	20/20	6921	10510	3644
287	18.46776682	18.46770038	18.46748244	10/20	15/20	19/20	9154	12256	8048
288	18.49430858	18.49434317	18.49428619	17/20	18/20	20/20	12386	9780	6817
289	18.51172660	18.51149183	18.51148362	8/20	17/20	20/20	10778	15841	9068
290	18.54871729	18.54868674	18.54862498	11/20	15/20	20/20	9271	9527	9384
291	18.56629865	18.56624000	18.56604139	12/20	16/20	20/20	10394	12255	6564
292	18.59451635	18.59437445	18.59440060	3/20	11/20	10/20	11351	8747	8005
293	18.62403455	18.62325319	18.62338391	0/20	8/20	4/20	10190	10018	11150
294	18.64459086	18.64429020	18.64427827	4/20	15/20	15/20	11960	10835	9956
303	18.92971365	18.92971972	18.92970239	8/20	12/20	20/20	11768	13794	5917
304	18.96335411	18.96299465	18.96307327	1/20	2/20	4/20	10331	10240	10726
305	19.00161173	19.00123339	19.00135969	5/20	20/20	15/20	8808	8162	12136
307	19.05981241	19.05981236	19.05981236	18/20	20/20	20/20	6262	10560	3259
308	19.10559347	19.10487262	19.10484180	3/20	11/20	15/20	10808	12596	8151
309	19.14212861	19.14134333	19.14132915	4/20	18/20	20/20	13226	11624	5045
310	19.17849825	19.17798487	19.17791453	9/20	14/20	17/20	10288	12616	8767
311	19.21158251	19.21214989	19.21104031	1/20	1/20	4/20	11256	13360	12815
320	19.45156768	19.45159036	19.45120573	2/20	0/20	20/20	9227	13275	11018
#Best	0	9	27	1	10	29	8	4	20

tion, HAMSP, HAMSP$_1$ and HAMSP$_2$ respectively obtain the best results on 29, 1, and 10 instances. As for the average computational time, the HAMSP algorithm consumes the least time for 20 out of the 32 instances. The outcome of this experiment implies that a good tradeoff between the intensification and the diversification of search plays an very important role in ensuring the performance of the proposed algorithm and that the proposed DIH method is able to provide such a tradeoff and thus guarantees a high performance of the HAMSP algorithm in terms of both solution quality and computational time.

6 Conclusions and Future Work

In this work, we investigate the classic equal circle packing problem and the point arrangement problem in a variety of convex containers, which have many real-world applications in different domains. Due to the short interactions between the circles or points, the number of local optimal solutions of these problems is huge and the corresponding optimization is very challenging. To solve efficiently these problems, we proposed a unified HAMSP algorithm by integrating several complementary search components, including the optima exploitation based adjustment method to adjust the minimum distance between points (or the circle centers) and the diversification and intensification based heuristic (DIH) method to achieve a desirable tradeoff between search intensification and diversification.

The computational results on a large number of popular benchmark instances show that the proposed algorithm is efficient and significantly outperforms the state-of-the-art algorithms in the literature, especially for the hard instances. In particular, for the equal circle packing problem in a circular container, which is the most studied circle packing problem, the proposed algorithm improves the best-known results for 69 out of 96 selected hard instances, while matching the best-known results for most of the remaining instances with a high success rate. For the remaining instances, the proposed algorithm also performs well for every performance metric considered. The experimental analysis shows that both the DIH method and the distance adjustment method used contribute significantly to the high performance of the algorithm. Particularly, unlike the existing distance adjustment methods that visit only one local optimum solution for the corresponding constrained optimization problem, our optima exploitation based adjustment method is able to explore a large number of local optima and then find high-quality solutions, which significantly enhances the search capacity of the proposed algorithm.

The present study can be extended from the following directions. First, the basic idea of the optima exploitation based adjustment method is very general and can be applied to other related max-min (or min-max) constrained optimization problems, such as covering a complicated region by the equal circles or spheres (Birgin et al., 2024). Second, the idea of the multi-scale perturbation strategy is also very general and it is very interesting to check its effectiveness on other optimization problems with a large number of local optima or saddle points. Third, to handle super-large scale instances with more than 10^3 points or circles, it would be interesting to integrate decomposition methods and dimension reduction strategies into the proposed algorithm to further improve its performance. Fourth, the proposed algorithm can be applied to higher dimensional problems with a convex container, such as the point arrangement problem on a unit sphere in k-dimensional ($k \geq 3$) space (i.e., the classic spherical code problem (https://spherical-codes.org)) and the k-dimensional ($k \geq 3$) sphere packing problems with a regular convex container, such as a spherical container. Finally,

by adding some specialized move operators, such as the insertion operator, which moves a high-energy circle from its current position to a vacancy position, the proposed algorithm can be adapted to the problems with a non-convex container as well.

Acknowledgments

We are grateful to the reviewers for their valuable comments and suggestions, which helped us to improve the paper. We thank the Beijing Beilong Super Cloud Computing Co., Ltd (http://www.blsc.cn/) for providing HPC resources that have contributed to the computational experiments reported in this work. This work was partially supported by the National Natural Science Foundation of China (Grant No. 61703213).

References

B. Addis, M. Locatelli, and F. Schoen. Disk packing in a square: a new global optimization approach. INFORMS Journal on Computing, 20(4):516–524, 2008.

H. Akeb, M. Hifi, and R. M'Hallah. A beam search algorithm for the circular packing problem. Computers & Operations Research, 36(5):1513–1528, 2009.

J. Akiyama, R. Mochizuki, N. Mutoh, and G. Nakamura. Maximin distance for n points in a unit square or a unit circle. In Japanese Conference on Discrete and Computational Geometry, pages 9–13. Springer, 2002.

P. Amore. Circle packing in regular polygons. Physics of Fluids, 35(2), 2023.

P. Amore and T. Morales. Efficient algorithms for the dense packing of congruent circles inside a square. Discrete & Computational Geometry, 70(1):249–267, 2023.

E. Basurto, P. Gurin, E. Specht, and G. Odriozola. Searching for the maximal packing fraction of hard disks confined by a circular cavity through replica exchange/event-chain monte carlo. The Journal of Chemical Physics, 161(4), 2024.

C. Baur and S. P. Fekete. Approximation of geometric dispersion problems. Algorithmica, 30:451–470, 2001.

E. G. Birgin and J. M. Gentil. New and improved results for packing identical unitary radius circles within triangles, rectangles and strips. Computers & Operations Research, 37(7): 1318–1327, 2010.

E. G. Birgin and F. Sobral. Minimizing the object dimensions in circle and sphere packing problems. Computers & Operations Research, 35(7):2357–2375, 2008.

E. G. Birgin, J. M. Martínez, F. Nishihara, and D. P. Ronconi. Orthogonal packing of rectangular items within arbitrary convex regions by nonlinear optimization. Computers & Operations Research, 33(12):3535–3548, 2006.

E. G. Birgin, J. L. Gardenghi, and A. Laurain. Bounds on the optimal radius when covering a set with minimum radius identical disks. Mathematics of Operations Research, 49(3): 1303–2047, 2024.

D. W. Boll, J. Donovan, R. L. Graham, and B. D. Lubachevsky. Improving dense packings of equal disks in a square. The Electronic Journal of Combinatorics, 7:R46–R46, 2000.

L. G. Casado, I. García, P. G. Szabo, and T. Csendes. Packing equal circles packing in square. II. new results for up to 100 circles using the TAMSASS-PECS algorithm. Optimization Theory: Recent Developments from Mátraháza, pages 207–224, 1998.

I. Castillo, F. J. Kampas, and J. D. Pintér. Solving circle packing problems by global optimization: numerical results and industrial applications. European Journal of Operational Research, 191(3):786–802, 2008.

M. Chen, Y. Yang, Z. Zeng, X. Tang, X. Peng, and S. Liu. A filtered beam search based heuristic algorithm for packing unit circles into a circular container. Computers & Operations Research, page 106636, 2024.

A. Costa, P. Hansen, and L. Liberti. On the impact of symmetry-breaking constraints on spatial branch-and-bound for circle packing in a square. Discrete Applied Mathematics, 161(1-2):96–106, 2013.

Z. Dai, K. Xu, and M. Ornik. Repulsion-based p-dispersion with distance constraints in non-convex polygons. Annals of Operations Research, pages 1–17, 2021.

E. D. Demaine, S. P. Fekete, and R. J. Lang. Circle packing for origami design is hard. arXiv preprint, arXiv:1008.1224v2, 2010.

A. Dimnaku, R. Kincaid, and M. W. Trosset. Approximate solutions of continuous dispersion problems. Annals of Operations Research, 136(1):65–80, 2005.

J. P. Doye, R. H. Leary, M. Locatelli, and F. Schoen. Global optimization of Morse clusters by potential energy transformations. INFORMS Journal on Computing, 16(4):371–379, 2004.

Z. Drezner and E. Erkut. Solving the continuous p-dispersion problem using non-linear programming. Journal of the Operational Research Society, 46:516–520, 1995.

A. V. Fiacco and G. P. McCormick. Computational algorithm for the sequential unconstrained minimization technique for nonlinear programming. Management Science, 10(4):601–617, 1964.

F. Fodor. The densest packing of 19 congruent circles in a circle. Geometriae Dedicata, 74: 139–145, 1999.

S. I. Galiev and M. S. Lisafina. Linear models for the approximate solution of the problem of packing equal circles into a given domain. European Journal of Operational Research, 230 (3):505–514, 2013.

M. Goldberg. The packing of equal circles in a square. Mathematics Magazine, 43(1):24–30, 1970.

R. Graham and B. Lubachevsky. Repeated patterns of dense packings of equal disks in a square. The Electronic Journal of Combinatorics, 3(R16):2, 1996.

R. L. Graham, B. D. Lubachevsky, K. J. Nurmela, and P. R. Östergård. Dense packings of congruent circles in a circle. Discrete Mathematics, 181(1-3):139–154, 1998.

A. Grosso, A. Jamali, M. Locatelli, and F. Schoen. Solving the problem of packing equal and unequal circles in a circular container. Journal of Global Optimization, 47(1):63–81, 2010.

K. He, H. Ye, Z. Wang, and J. Liu. An efficient quasi-physical quasi-human algorithm for packing equal circles in a circular container. Computers & Operations Research, 92:26–36, 2018.

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

W. Huang and T. Ye. Greedy vacancy search algorithm for packing equal circles in a square. Operations Research Letters, 38(5):378–382, 2010.

W. Huang and T. Ye. Global optimization method for finding dense packings of equal circles in a circle. European Journal of Operational Research, 210(3):474–481, 2011.

X. Lai, J.-K. Hao, D. Yue, Z. Lü, and Z.-H. Fu. Iterated dynamic thresholding search for packing equal circles into a circular container. European Journal of Operational Research, 299(1):137–153, 2022.

X. Lai, J.-K. Hao, R. Xiao, and F. Glover. Perturbation based thresholding search for packing equal circles and spheres. INFORMS Journal on Computing, 35(4):725–746, 2023.

X. Lai, Z. Lin, J.-K. Hao, and Q. Wu. An efficient optimization model and tabu search-based global optimization approach for the continuous p-dispersion problem. INFORMS Journal on Computing, 2024. doi: 10.1287/ijoc.2023.0089.

R. H. Leary. Global optimization on funneling landscapes. Journal of Global Optimization, 18(4):367–383, 2000.

I. Litvinchev and E. Ozuna. Approximate packing circles in a rectangular container: valid inequalities and nesting. Journal of Applied Research and Technology, 12(4):716–723, 2014.

I. Litvinchev, L. Infante, and E. L. O. Espinosa. Using valid inequalities and different grids in lp-based heuristic for packing circular objects. In Intelligent Information and Database Systems: 8th Asian Conference, ACIIDS 2016, Da Nang, Vietnam, March 14–16, 2016, Proceedings, Part II 8, pages 681–690. Springer, 2016.

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. Mathematical Programming, 45(1):503–528, 1989.

J. Liu, S. Xue, Z. Liu, and D. Xu. An improved energy landscape paving algorithm for the problem of packing circles into a larger containing circle. Computers & Industrial Engineering, 57(3):1144–1149, 2009.

M. Locatelli and U. Raber. Packing equal circles in a square: a deterministic global optimization approach. Discrete Applied Mathematics, 122(1-3):139–166, 2002.

C. O. López and J. E. Beasley. A heuristic for the circle packing problem with a variety of containers. European Journal of Operational Research, 214(3):512–525, 2011.

B. D. Lubachevsky. How to simulate billiards and similar systems. Journal of Computational Physics, 94(2):255–283, 1991.

C. D. Maranas, C. A. Floudas, and P. M. Pardalos. New results in the packing of equal circles in a square. Discrete Mathematics, 142(1-3):287–293, 1995.

M. C. Markót. Improved interval methods for solving circle packing problems in the unit square. Journal of Global Optimization, 81(3):773–803, 2021.

M. C. Markót and T. Csendes. A new verified optimization technique for the "packing circles in a unit square" problems. SIAM Journal on Optimization, 16(1):193–219, 2005.

H. Melissen. Densest packings of congruent circles in an equilateral triangle. The American Mathematical Monthly, 100(10):916–925, 1993.

H. Melissen. Densest packings of eleven congruent circles in a circle. Geometriae Dedicata, 50:15–25, 1994.

N. Mladenović, F. Plastria, and D. Urošević. Reformulation descent applied to circle packing problems. Computers & Operations Research, 32(9):2419–2434, 2005.

K. J. Nurmela and P. R. Östergård. Packing up to 50 equal circles in a square. Discrete & Computational Geometry, 18(1):111–120, 1997.

E. Specht. Packomania website. http://packomania.com, 2023.

Y. Stoyan, G. Yaskov, T. Romanova, I. Litvinchev, S. Yakovlev, and J. M. V. Cantú. Optimized packing multidimensional hyperspheres: A unified approach. Mathematical Biosciences and Engineering, 17(6):6601–6630, 2020.

P. G. Szabó and E. Specht. Packing up to 200 equal circles in a square. In Models and Algorithms for Global Optimization, volume 4 of Optimization and Its Applications, pages 141–156. Springer, 2007.

E. R. Van Dam, B. Husslage, D. Den Hertog, and H. Melissen. Maximin Latin hypercube designs in two dimensions. Operations Research, 55(1):158–169, 2007.

J. Zhou, K. He, J. Zheng, and C.-M. Li. Geometric batch optimization for packing equal circles in a circle on large scale. Expert Systems with Applications, 250:123952, 2024.