# Iterated dynamic thresholding search for packing equal circles into a circular container

Xiangjing Lai [a], Jin-Kao Hao [b,*], Dong Yue [a], Zhipeng Lü [c], Zhang-Hua Fu [d,e]

[a] *Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*

[b] *LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

[c] *SMART, School of Computer Science and Technology, Huazhong University of Science and Technology, 430074 Wuhan, P.R.China*

[d] *Institute of Robotics and Intelligent Manufacturing, The Chinese University of Hong Kong, Shenzhen 518172, China*

[e] *Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen 518172, China*

**European Journal of Operational Research**, August 2021
https://doi.org/10.1016/j.ejor.2021.08.044

**Abstract**

Packing equal circles in a circle is a classic global optimization problem that has a rich research history and a number of relevant applications. The problem is computationally challenging due to the fact that the number of possible packing configurations grows exponentially with the number of circles. In this work, we propose a highly effective iterated dynamic thresholding search algorithm for solving this difficult problem. The algorithm integrates several features including a two-phase local optimization method, a dynamic thresholding search and a container adjustment procedure. Computational experiments on popular benchmark instances with up to $N = 320$ circles show that the algorithm outperforms significantly the state-of-the-art algorithms. In particular, it improves the best-known results for 136 instances, while matching the best-known results for other 175 instances.

*Keywords*: Packing, thresholding search, global optimization, heuristics.

* Corresponding author.
*Email addresses:* laixiangjing@gmail.com (Xiangjing Lai), jin-kao.hao@univ-angers.fr (Jin-Kao Hao), medongy@vip.163.com (Dong Yue), zhipeng.lui@gmail.com (Zhipeng Lü), fuzhanghua@cuhk.edu.cn (Zhang-Hua Fu).

## 1 Introduction

Given $N$ unitary circles $\{c_1, c_2, \ldots, c_N\}$ with $(x_i, y_i)$ being the center of circle $c_i$, the Packing Equal Circles in a Circle (PECC) problem involves packing the $N$ circles into a circular container $c_0$ such that no two circles overlap and no circle exceeds the container, while the radius $R$ of the container is minimized. Formally, the PECC problem can be modeled as a nonlinear constrained optimization problem as follows:

$$\text{Minimize} \quad R \tag{1}$$

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

$$\sqrt{x_i^2 + y_i^2} + 1 \leq R, i = 1, 2, \ldots, N \tag{3}$$

where $R$ is the radius of the container with a center at the origin of the two-dimensional Cartesian coordinate system, $(x_i, y_i)$ and $(x_j, y_j)$ represent respectively the centers of circles $c_i$ and $c_j$, the constraints (2) guarantee that any two circles do not overlap, and the constraints (3) ensure that none of the $N$ circles exceeds the container.

The PECC problem has a number of industrial applications, such as circular cutting, fiber optic cable manufacturing, container loading, cylinder packing and facility and dashboard layout [7]. On the other hand, the problem is known to be NP-hard [11,23] and thus solving the problem is computationally challenging.

In addition, the PECC problem is a natural and remarkable test system for evaluating various general-purpose global optimization techniques due to the following features [27]. First, in spite of its simplicity in form, the number of its local optimum solutions is huge even for medium-sized instances, which increases exponentially as $N$ increases [21]. Second, a solution of the problem has a visual and intuitive graphical representation on a 2-dimensional plane, which facilitates the interpretation of the solution. Third, the absence of particular structural features of the problem makes it very suitable to act as a benchmark for testing general global optimization algorithms.

Since 1967 [28], a large number of studies and algorithms have been reported in the literature for solving the PECC problem. In the early stage of research history, many efforts were devoted to finding the optimum solution for small instances. For example, in 1968 and 1969, Graham and Pirl proved the optimality of solutions for $N \leq 7$ and $N \leq 10$, respectively [18,45]. In 1994, Melissen proved the optimal solution for $N = 11$ [42], while the optimality for $N = 19$ was provided by Fodor in 1999 [15]. Nevertheless, for large instances, no method is known to be able to solve the problem optimally.

Thus, most of later studies were devoted to designing effective heuristic algorithms that are based on the non-convex continuous optimization model and can be summarized as follows. In 1998, Graham et al. proposed a billiards simulation method to solve the PECC problem and published for the first time the putative global optimum solutions for up to $N = 65$ [19]. In 2002, Akiyama et al. investigated a greedy heuristic algorithm and improved the best-known results for a number of instances with $N \leq 100$ [3]. In 2005, Mladenović et al. presented a reformulation descent algorithm, where two different formulations of the PECC problem are alternately used to avoid search stagnations [43]. In 2009, Grosso et al. developed two basin-hopping algorithms and improved the best-known results for 20 instances with $N \leq 100$ [21]. In the same year, Liu et al. proposed an energy landscape paving algorithm and reported competitive results for some instances with $N \leq 100$ [31]. In 2010, Birgin and Gentil investigated the resolution of non-linear equations systems through the Newton-Raphson method and improved the accuracy of previous results attained by continuous optimization approaches [5]. In 2011, Huang and Ye designed a highly efficient quasi-physical global optimization algorithm and further improved the best-known results for 63 instances with $N \leq 200$ [27]. In the same year, López and Beasley developed a heuristic algorithm based on the formulation space search method for the problem of packing equal circles into a variety of containers [36], including the circular container. They showed that their algorithm outperforms the earlier formulation space search method [43]. In 2018, Chen et al. designed a greedy heuristic by means of a greedy corner-occupying placement strategy, and presented competitive results for small instances with $N \leq 100$ [8]. In the same year, He et al. developed a quasi-physical quasi-human algorithm, and improved 66 best-known results for instances with $N \leq 320$ [23]. Recently, Stoyan et al. studied several optimization strategies for a number of circle and sphere packing problems, including PECC, and their computational results show that the adopted optimization strategies are very efficient [49]. In particular, they reported improved results for several large-scale PECC instances with up to 5000 circles.

Additionally, from the well-known Packomania website maintained by Specht [47], we observe that researchers launched an interesting competition on the PECC problem. In this contest, people are invited to solve the PECC problem by their own algorithms and then provide the improved solutions with respect to the best-known result, regardless of the computational resource used. From the continually updated history of the website for the best-known results, we observe that in the range of $101 \leq N \leq 200$, Huang et al [27] and Cantrell report the best-known results for most instances. For a majority of instances of $N > 201$, the best-known results are hold by Specht and Cantrell [47]. For $N > 1000$, the best results are provided by the IIPP-random/lattice-IPOPT algorithm [49] that uses lattice packings as the initial solution of the algorithm for 17 large scale instances with up to 5000 circles. In view of the updating history of the Packomania website and the newest results reported

in [23], the quasi-physical quasi-human algorithm in [23], the quasi-physical global optimization in [27], the IIPP-random/lattice-IPOPT algorithm [49], and Cantrell's unpublished algorithm [47] can be regarded to represent the current state-of-the-art for solving the PECC problem.

In addition to the previous review, there are a large number of studies dedicated to several variants of the PECC problem, including mainly the problem of packing equal circles into a variety of containers, the problem of packing unequal circles into a regular container, and the sphere packing problems in the 3-dimensional space. For these variants, numerous algorithms have been proposed in the literature, which can be mainly divided into the following categories. The first category of algorithms are heuristics based on the non-convex continuous optimization model, such as the quasi-physical quasi-human algorithm [54], monotonic basin-hopping algorithm [2], population basin-hopping algorithm [1], simulated annealing algorithms [56,44], tabu search algorithm [16], greedy vacancy search algorithm [26], local search-based methods [24,25], formulation space search methods [37,38], hybrid heuristic algorithms [6,55], jump algorithm [51,52], and popular Packmol approach which is a package for building initial configurations for molecular dynamics simulations [41]. The second category of algorithms are based on the ideas of discrete optimization [53]. For instance, Lü and Huang proposed a stochastic growth algorithm called PERM in which the circles are packed one by one into the container [39]. Based on the constructed grids, Galiev et al. investigated several integer linear programming models for the approximate solution of the problem of packing equal circles into a given domain [17], where the centers of circles can locate only at the grid points. Using valid inequalities and different grids, Litvinchev et al. studied some integer linear programming-based heuristics [32,33]. The third category of algorithms include a deterministic global optimization approach [34] and deterministic method for high density packing of equal circles in rectangles with variable aspect ratio [46]. In addition, there exist several other optimization algorithms for circle packing problems, such as an interval analysis-based optimization algorithm [40] and heuristic algorithms based on the phi-function technique [20,49,50].

Our literature review indicates that the PECC problem remains a hot research topic due to its computational challenge and relevant applications. According to the update history of the Packomania website [47], the best-known results were continually improved by new algorithms, which implies that there is still room for improvement. In this work, we aim to further advance the state-of-the-art of solving the PECC problem. Specifically, we design a new heuristic algorithm for effectively solving large PECC instances with $N > 100$. The algorithm combines a fast two-phase local optimization method to find high quality local optimum solutions, an original dynamic thresholding search method to explore the search space and a dedicated penalty-based container adjustment procedure to reinforce the search intensification. Computational

experiments on popular benchmark instances with up to $N = 320$ circles show that the algorithm outperforms significantly the state-of-the-art algorithms. In particular, it improves the best-known results (new upper bounds) for 136 instances with $N > 100$, while matching the best-known results for other 175 instances. In addition to these results, the underlying ideas behind the two-phase local optimization and the dynamic thresholding search are of general interest and could be used in other search algorithms for solving more optimization problems.

The rest of paper is organized as follows. In Section 2, the general idea of the proposed algorithm is provided. In Section 3, the proposed algorithm is described in detail. In Section 4, computational results on benchmark instances and comparisons are provided. In Section 5, the impacts of several important ingredients of the algorithm are analyzed. Conclusions are drawn in the last section with discussion of research perspectives.

## 2 General Approach for Solving the PECC Problem

The PECC problem is a nonlinear constrained optimization problem that is difficult to solve directly. To cope with this difficulty, we convert the problem into a series of constraint satisfaction problems which are solved by means of the penalty function approach.

### *2.1 Reformulation of the PECC Problem with Fixed-radius Containers*

The adopted approach reformulates the PECC problem as a series of constraint satisfaction problems where each problem has a fixed-radius container whose goal is to pack the circles in the given container.

***Definition 1:*** Given $N$ unitary circles $\{c_1, \ldots, c_N\}$ with $(x_i, y_i)$ being the center of circle $c_i$, and a circular container $c_0$ of radius $R$ ($R > \sqrt{N}$), the Packing Equal Circles in a Circle problem with a Fixed-Radius container (PECC-FR) is to find a packing configuration that satisfies the constraints that 1) no two circles $c_i$ and $c_j$ overlap (i.e., $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq 2, 1 \leq i \neq j \leq N$), and 2) none of the $N$ circles exceeds the container (i.e., $\sqrt{x_i^2 + y_i^2} + 1 \leq R$ for any circle $c_i$).

For a given container of radius $R$, we use PECC-FR[$R$] to denote the particular PECC-FR problem.

To deal with PECC-FR[$R$], we convert it to an unconstrained nonlinear opti-

mization problem defined on the $2N$-dimensional Euclidean space $\Re^{2N}$, where a candidate solution is represented by the coordinate vector $X = (x_1, y_1, \ldots, x_N, y_N)$ indicating the centers of $N$ circles $c_i = (x_i, y_i)$ $(i = 1, 2, \ldots, N)$.

To assess the quality of a candidate solution $X$ (i.e., a packing configuration) of PECC-FR[$R$], we follow the general penalty function approach (see examples in [26,27,41,48]) and define the penalty-based objective function $E_R(X)$ as follows.

$$E_R(X) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} O_{ij}^2 + \sum_{i=1}^{N} O_{0i}^2 \tag{4}$$

where

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

$$O_{0i} = max\{0, \sqrt{x_i^2 + y_i^2} + 1 - R\} \tag{6}$$

In other words, $O_{ij}$ quantifies the overlap between two circles and $O_{0i}$ quantifies the overlap between any circle and the container. As the result, $E_R(X)$ measures the degree of constraint violation of the candidate packing configuration $X$. $E_R(X) = 0$ (true if $E_R(X) < 10^{-25}$ in this work) indicates that $X$ is a feasible solution to PECC-FR[$R$]; otherwise, $X$ violates non-overlapping constraints and is thus an infeasible or conflicting solution. Given two packing configurations $A$ and $B$, $A$ is better than $B$ if and only if $E_R(A) < E_R(B)$.

Fig. 1 shows a conflicting packing configuration with the two types of overlaps.

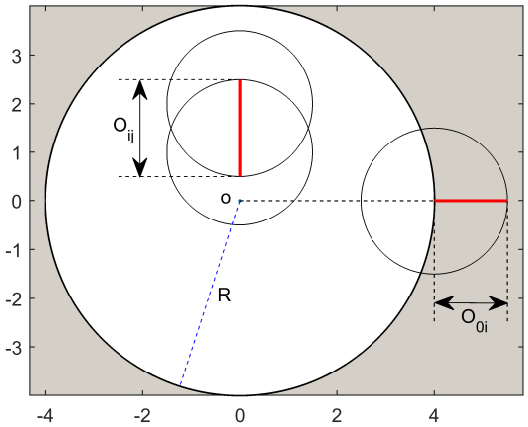



Fig. 1. An illustrative example for a conflicting packing configuration with overlaps.

### *2.2 Solving the PECC Problem based on Constraint Satisfaction*

With the above formulation, we approximate the PECC problem by solving a series of PECC-FR problems with the following procedure.

(1) Set the container radius $R$ to an initial value;
(2) Solve the associated PECC-FR[$R$] problem by minimizing the penalty-based objective function $E_R(X)$ with an unconstrained nonlinear optimization method;
(3) Adjust the solution $X$ and radius $R$ of container from step (2) by a container adjustment method to obtain a feasible packing configuration $(X, R)$ such that $R$ is locally minimized;
(4) Update $R_{min}$ by the adjusted $R$ (i.e., $R_{min} \leftarrow R$) if $R < R_{min}$, where $R_{min}$ is the smallest container radius found so far permitting a feasible solution;
(5) Go to step (2) with $R = R_{min} - \Delta_R$, where $\Delta_R$ is a reduction factor.
(6) Repeat steps (2)–(5) until a stopping condition is met and then returns the last feasible solution found.

The last obtained $R_{min}$, for which a feasible solution for the PECC-FR[$R_{min}$] problem is found, defines an upper bound of the optimal solution of the given PECC problem.

## 3 The Proposed Algorithm

The proposed Iterated Dynamic Thresholding Search (IDTS) algorithm for the PECC problem follows the general procedure outlined in Section 2.2. Its main framework and components are presented in this section.

### *3.1 Framework of Iterated Dynamic Thresholding Search*

The IDTS algorithm employs an initialization procedure to obtain a starting container radius as small as possible, a dynamic thresholding search (DTS) procedure to solve the PECC-FR problem described in Section 2.1, and a container adjustment procedure to further optimize locally the container radius.

As shown in Algorithm 1, IDTS can be viewed as a two-stage search algorithm. At the first stage (lines 2–3), an initialization procedure is used to obtain a feasible packing configuration $X$ with a small container radius $R$, and $X$ and $R$ are recorded as the current best solution $(X^*, R_{min})$. The initialization procedure is based on the popular monotonic basin-hopping (MBH) algorithm and is described in Section 3.3.

At the second stage (lines 6–14), the search iterates an improvement process to find new feasible packing configurations with reduced container radiuses. At each iteration, the current container radius $R$ is set to $R_{min} - \Delta_R$, where

**Algorithm 1:** Framework of iterated dynamic thresholding search

**Input:** Number of circles to be packed ($N$), maximum time limit ($t_{max}$), shrinkage factor ($\theta < 1$)

**Output:** The best packing found ($X^*$) and its container radius ($R_{min}$)

1 **/* First Stage of the Search */**
2 $(X, R) \leftarrow Initialization(N)$ `/* Algorithm 2, generate a feasible packing` $X$ `and obtain an initial sufficiently small radius` $R$ `*/`
3 $X^* \leftarrow X$, $R_{min} \leftarrow R$ `/*` $(X^*, R_{min})$ `records the best packing found */`
4 **/* Second Stage of the Search */**
5 $\Delta_R \leftarrow 0.1$
6 **while** $time() \leq t_{max}$ **do**
7 $\quad \Delta_R \leftarrow max\{\theta * \Delta_R, 10^{-4}\}$ `/* Reduce gradually` $\Delta_R$ `ensuring that` $\Delta_R \geq 10^{-4}$ `*/`
8 $\quad R \leftarrow R_{min} - \Delta_R$ `/* Reduce the radius of container by` $\Delta_R$`, Step(5) of Section 2.2 */`
9 $\quad X \leftarrow DynamicThresholdSearch(R)$ `/* Algorithm 3, find the best packing configuration for the fixed radius container` $R$`, Step(2) of Section 2.2 */`
10 $\quad (X, R) \leftarrow AdjustContainerRadius(X, R)$ `/* Algorithm 4, Step(3) of Section 2.2 */`
11 $\quad$ **if** $R < R_{min}$ **then**
12 $\quad\quad R_{min} \leftarrow R$, $X^* \leftarrow X$ `/* Step(4) of Section 2.2 */`
13 $\quad$ **end**
14 **end**
15 **return** $(X^*, R_{min})$

$\Delta_R$ is a parameter whose value is adaptively tuned by multiplying a factor $\theta \in (0, 1)$. Then, the dynamic thresholding search method (see Section 3.4) is used to solve the corresponding PECC-FR[$R$] problem, i.e, to seek a minimum-overlapping packing configuration with the container radius $R$ by minimizing the objective function $E_R$ defined in Section 2.1. Subsequently, the obtained packing configuration and the container radius are slightly adjusted by the container adjustment procedure (line 10) to obtain a feasible packing configuration in which the container radius is locally minimized. The current best solution $(X^*, R_{min})$ is updated each time a feasible packing with a smaller $R$ is discovered (lines 11–13). The second stage of the algorithm stops when a time limit ($t_{max}$) is reached.

Both search stages above rely on a lower level local optimization procedure which is presented in the next subsection. The other search components are described in the rest of this section.

Given an input solution $X$, the purpose of the local optimization procedure is to reach its nearest local optimum solution with respect to the objective function $E_R(X)$ defined in Section 2.1. Specifically, the local optimization procedure is based on the popular LBFGS algorithm [30] (a limited memory quasi-newton method) that adopts an efficient line search approach proposed in [22]. To ensure a high computational efficiency, the search of local optimization is divided into two phases.

The first phase runs LBFGS to optimize $E_R(X)$ with a relaxed stopping condition of $||g||_\infty < 10^{-2}$ with $||g||_\infty = max\{|g_1|, |g_2|, \ldots, |g_{2N}|\}$ being the maximum norm of the gradient $g$ of $E_R(X)$. Due to the relaxed stopping condition, the first phase is typically very fast and terminates with an approximate local optimum solution where the neighbors of circles generally don't change any more. To further improve the solution, the second phase runs LBFGS with a high precision stopping condition of $||g||_\infty < 10^{-13}$ and only considers the pairwise overlaps between the neighboring circles and the overlaps between the circles and the container. Here, two circles $c_i$ and $c_j$ are considered to be neighboring if the distance between their centers $D(c_i, c_j)$ is less than $d_{cut}$, which is a parameter and can be set according to the demand. In fact, it is generally adequate to set $d_{cut}$ to be slightly larger than 2.0 (i.e., the diameter of unit circles). Nevertheless, for the sake of conservation, $d_{cut}$ is by default set to 4.0 in this work. Fig. 2 gives an illustrative example for the overlaps between the circle $c_i$ and other circles that should be considered in the second phase. As such, the evaluations of $E_R(X)$ and its gradient $g$ of the second phase are much faster than the first phase. This enables the local optimization procedure to perform a high number of iterations with the given time limit. Our experiment indicates this two-phase approach significantly speeds up the local optimization process especially for large-scale instances.

The basic idea of the two-phase strategy is to speed up the search process via the use of the adjacency relationship between circles. The first phase aims to construct the adjacency relationship between circles by a short local optimization with a small number of iterations. Based on the generated adjacency relationship, the second phase aims to reach a local minimum solution with a high precision as fast as possible. It is worth noting that other techniques (e.g., the popular and general Voronoi diagram) can be used to construct the adjacency relationship between circles. In our case, we used the distances between the circles to construct the adjacency relationship, which ensures the simplicity and efficiency of the approach.

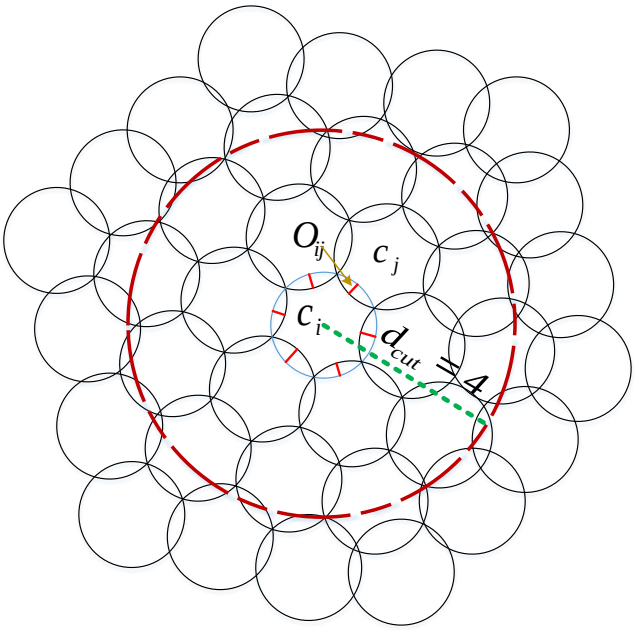



Fig. 2. For the circle $c_i$, the overlaps needed to be considered in the second phase of local optimization are indicated by the red line segments, where $O_{ij}$ indicates the overlap between $c_i$ and $c_j$, and $d_{cut}$ is set to 4. Two circles $c_i$ and $c_j$ are mutually considered as neighbors if the distance between their centers $D(c_i, c_j)$ is less than $d_{cut}$.

### *3.3 Initialization Procedure*

The initialization procedure of the IDTS algorithm aims to obtain rapidly a starting solution with a container radius as small as possible, which is used to seed the second stage of dynamic thresholding search.

As shown in Algorithm 2, a random packing with the container radius $R_0 = \sqrt{\frac{N}{0.85}}$ is first generated by distributing uniformly and randomly the centers $(x_i, y_i)$ of the $N$ circles in the container (line 3) and then the corresponding PECC-FR$[R_0]$ is solved by the popular MBH algorithm [21,29] (line 4), where 0.85 is an estimated packing density for the initial solution generated and is inspired by the best-known results from the Packomania website [47]. Subsequently, the container radius $R$ is adjusted by the container adjustment procedure (Section 3.5) to reach a feasible packing configuration, while $R$ is locally minimized (line 5).

The employed MBH method works as follows. Starting from an input solution $X_0$, it performs a number of iterations to improve the current solution $X$ with respect to the objective function $E_R(\cdot)$. At each iteration, the best solution $X_b$ found so far is slightly modified by a *Perturbation* operator and then improved by the two-phase local optimization procedure described in Section 3.2. The

**Algorithm 2:** Initialization

1 **Function** *Initialization()*
**Input:** The number of circles to be packed ($N$)
**Output:** A feasible packing $X$ with its radius $R$
2 $R_0 \leftarrow \sqrt{\frac{N}{0.85}}$ `/* The first estimated container radius */`
3 $X \leftarrow RandomPacking(R_0)$ `/* Generate a random packing by distributing uniformly randomly the centers of circles in the container with radius` $R_0$ `*/`
4 $X \leftarrow MBH(X, R_0)$ `/* Solving the PECC-FR[`$R_0$`] problem by minimizing the objective function` $E_R$ `*/`
5 $(X, R) \leftarrow AdjustContainerRadius(X, R_0)$ `/* Algorithm 4, adjust the container radius to obtain a feasible packing */`
6 **return** $(X, R)$

improved solution is accepted as the current solution if it is better than $X_b$; otherwise it is discarded. The MBH algorithm stops and returns the best solution found $X_b$ if it has not been updated after $\alpha$ consecutive iterations, where $\alpha$ is a parameter whose default value is set to 100 in this work. To perturb a solution, we shift randomly each coordinate $x_i$ (or $y_i$) of $X$ in a given interval to generate a new solution $X'$, i.e., $x_i \leftarrow x_i + r$ $(1 \leq i \leq 2N)$, where $r$ is a random number in $[-0.8, 0.8]$.

### *3.4 Dynamic Thresholding Search*

The main search component of the proposed IDTS algorithm concerns the dynamic thresholding search procedure, which seeks a high-quality packing configuration for the PECC-FR problem with a given container radius $R$. This DTS procedure follows the general Threshold Accepting heuristic [13] and is described in Algorithm 3.

Starting from a conflicting solution $X$ (lines 2–4), DTS performs a number of iterations to find improved solutions in terms of the objective function $E_R(\cdot)$ (lines 6–25). At each iteration, the current solution $X$ is first perturbed by the perturbation operator (i.e., the random perturbation, Section 3.3) and then improved by the two-phase local optimization procedure of Section 3.2 ($TPLocalOptimization()$, lines 7–8). Subsequently, an accepting criterion is applied to determine whether the newly obtained solution $X_{new}$ is accepted as the current solution (lines 10–15). Specifically, $X_{new}$ is accepted if the objective variation between $X_{new}$ and $X$ (i.e., $\Delta_E = E_R(X_{new}) - E_R(X)$) is smaller than a threshold ($Th_E$); otherwise, $X_{new}$ is discarded. Moreover, the threshold $Th_E$ is adaptively adjusted to maintain an acceptance rate of $\frac{1}{2}$ for the new solutions (lines 16–20). Specifically, $Th_E$ is decreased by multiplying a factor $\mu \in (0, 1)$

**Algorithm 3:** Dynamic-thresholding search for the PECC problem with a fixed radius $R$ of container

**Function** *DynamicThresholdSearch*
**Input:** Maximum of iterations $MaxIter$, fixed container radius $R$, and shrinkage factor $\mu$ ($\mu < 1.0$)
**Output:** The best solution found $(X^*, R)$
$X \leftarrow RandomPacking(R)$ `/* Generate a random packing in the container with radius` $R$ `*/`
$X \leftarrow TPLocalOptimization(X)$ `/* Minimize function` $E_R(\cdot)$ `from` $X$ `*/`
$X^* \leftarrow X$
$Iter \leftarrow 0$, $Th_E \leftarrow 10^{-4}$, $N_{accept} \leftarrow 0$, $N_{reject} \leftarrow 0$ `/* Initialization */`
**while** $(Iter \leq MaxIter) \wedge (E_R(X^*) > 10^{-25})$ **do**
  $X_{new} \leftarrow Perturbation(X)$
  $X_{new} \leftarrow TPLocalOptimization(X_{new})$ `/* Minimize function` $E_R(\cdot)$ `from` $X_{new}$ `using two-phase local optimization method */`
  $\Delta_E \leftarrow E_R(X_{new}) - E_R(X)$
  **if** $(\Delta_E < Th_E) \wedge (\Delta_E \neq 0)$ **then**
    $X \leftarrow X_{new}$
    $N_{accept} \leftarrow N_{accept} + 1$
  **else**
    $N_{reject} \leftarrow N_{reject} + 1$
  **end**
  **if** $N_{accept} > N_{reject}$ **then**
    $Th_E \leftarrow \mu * Th_E$ `/* Decrease the threshold value` $Th_E$ `*/`
  **else**
    $Th_E \leftarrow \frac{1}{\mu} * Th_E$ `/* Increase the threshold value` $Th_E$ `*/`
  **end**
  **if** $E_R(X) < E_R(X^*)$ **then**
    $X^* \leftarrow X$ `/* Save the best solution found */`
  **end**
  $Iter \leftarrow Iter + 1$
**end**
**return** $(X^*, R)$

if the number of acceptances ($N_{accept}$) becomes larger than the number of rejections ($N_{reject}$), and is increased by dividing $\mu$ otherwise. Finally, the DTS procedure stops when the maximum number of iterations ($MaxIter$) has been reached or a feasible solution $X$ (i.e., $E_R(X) < 10^{-25}$) is found, and the best solution found $(X^*, R)$ is returned as the result of the DTS procedure.

It is worth noting that the DTS procedure is of general-purpose and does not require any problem-specific knowledge except the objective function. As such, it can be applied to other unconstrained optimization problems. Moreover, it

provides an original way of adjusting adaptively its threshold based on the current acceptance rate, instead of the objective function which was commonly used in the literature.

### *3.5 Container Adjustment Procedure*

**Algorithm 4:** Container adjustment procedure

**Function** *AdjustContainerRadius*
**Input:** Input solution $s_0 = (X_0, R_0)$, maximum of iterations $Q$ $(= 10)$
**Output:** The local optimum packing $s = (X, R)$
$X \leftarrow X_0$, $R \leftarrow R_0$, $\rho \leftarrow 10^6$
**for** $i \leftarrow 1$ ***to*** $Q$ **do**
    $(X, R) \leftarrow LocalOptimization(X, R)$  `/* Using the LBFGS method to minimize` $U_\rho(X, R)$ `*/`
    $\rho \leftarrow 10 * \rho$
**end**
**return** $(X, R)$

Given a feasible or infeasible packing configuration $(X, R)$, the container adjustment procedure aims to adjust the container radius $R$ and the coordinate vector $X$ of the $N$ circles such that the resulting packing configuration is feasible while the radius $R$ is locally minimized. To do this, starting from the given packing configuration $(X, R)$, we solve locally the initial constrained PECC problem defined by Eqs. (1)–(3) in Section 1.

Due to the computational difficulty of solving directly the constrained optimization problem by means of a local optimization method, we employ in this study the popular sequential unconstrained minimization technique (SUMT) [14] to handle this constrained problem. First, the PECC problem is converted into a series of unconstrained minimization problems that can be described as

$$Minimize \quad U_\rho(X, R) = R^2 + \rho * E(X, R) \tag{7}$$

where $\rho$ is a penalty factor and each value of $\rho$ defines an unconstrained minimization problem, $R$ is a variable representing the container radius. The penalty term $E(X, R)$ with $2N + 1$ variables is defined as follows:

$$E(X, R) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} O_{ij}^2 + \sum_{i=1}^{N} O_{0i}^2 \tag{8}$$

where $O_{ij}$ and $O_{0i}$ represent respectively the overlap between two circles $c_i$ and $c_j$ and the overlap between a circle $c_i$ and the exterior of the container

$c_0$. Then, we solve consecutively a series of unconstrained optimization problems (defined by Eq. (7) ) with the increasing $\rho$ values by an unconstrained optimization method, starting from an input solution and an initial $\rho$ value.

The pseudo-code of the container adjustment procedure is given in Algorithm 4. Starting from the input packing configuration $(X_0, R_0)$ and initial $\rho$ value (empirically set to $10^6$ in this work), the procedure performs $Q$ iterations ($Q = 10$ in this work) to reach a feasible packing configuration. At each iteration, the powerful LBFGS method [30] is used to optimize the function $U_\rho(X, R)$ with $2N + 1$ variables, and then the value of $\rho$ is increased as $\rho \leftarrow 10 * \rho$ and the resulting packing configuration $(X, R)$ is used as the input solution of next iteration. According to Eq. (7), the term $E(X, R)$ is penalized more with a large $\rho$ value than with a small $\rho$ value. As such, as the value of $\rho$ increases to a very large number, the search is forced to converge to a feasible solution $(X, R)$ (i.e., $E(X, R) = 0$) with its container radius $R$ locally minimized.

It is worth noting that we employ $R^2$ in equation (7) instead of $R$ for the following reasons. First, $f(R) = R^2$ is a monotone increasing function with respect to the variable $R$ in the range of $R \geq 0$, thus the minimization of function $f(R) = R^2$ is equivalent to that of $f(R) = R$. Second, $f(R) = R^2$ is more smooth than the function $f(R) = R$, which is beneficial to the optimization with the LBFGS method.

### *3.6 Discussion on the Innovations of the Proposed Algorithm*

Compared with the existing circle packing algorithms in the literature, the proposed IDTS algorithm includes mainly two original features concerning its two-phase local optimization and dynamic thresholding search.

First, IDTS is the first PECC algorithm adopting a two-phase local optimization approach. In fact, unlike other acceleration techniques such as those proposed in [4,23], the proposed local optimization method is composed of two LBFGS procedures, where the first aims to generate a local minimum solution with a low precision and obtain the adjacency relation between the circles, and the second aims to obtain a local minimum solution with a high precision based on the adjacency relation generated by the first LBFGS application. This two-phase optimization approach has the main advantage of accelerating the search without sacrificing solution quality. Second, the dynamic thresholding search is applied for the first time to the PECC problem. This key search component utilizes an original technique to dynamically adjust the thresholding value at each iteration based on the current acceptance rate of the new solutions.

As we show in the next section, the IDTS algorithm integrating these features

performs very well on the tested PECC instances. Moreover, given that the ideas of the two-phase local optimization and the dynamic thresholding search are very general, they can be adapted to a number of other geometry optimization problems, such as unequal circles packing and structural optimization of atomic clusters [12,29,35].

## 4 Computational Experiments and Assessments

In this section, we evaluate the performance of the IDTS algorithm on a large number of benchmark instances commonly used in the literature and make comparisons with state-of-the-art results in the literature.

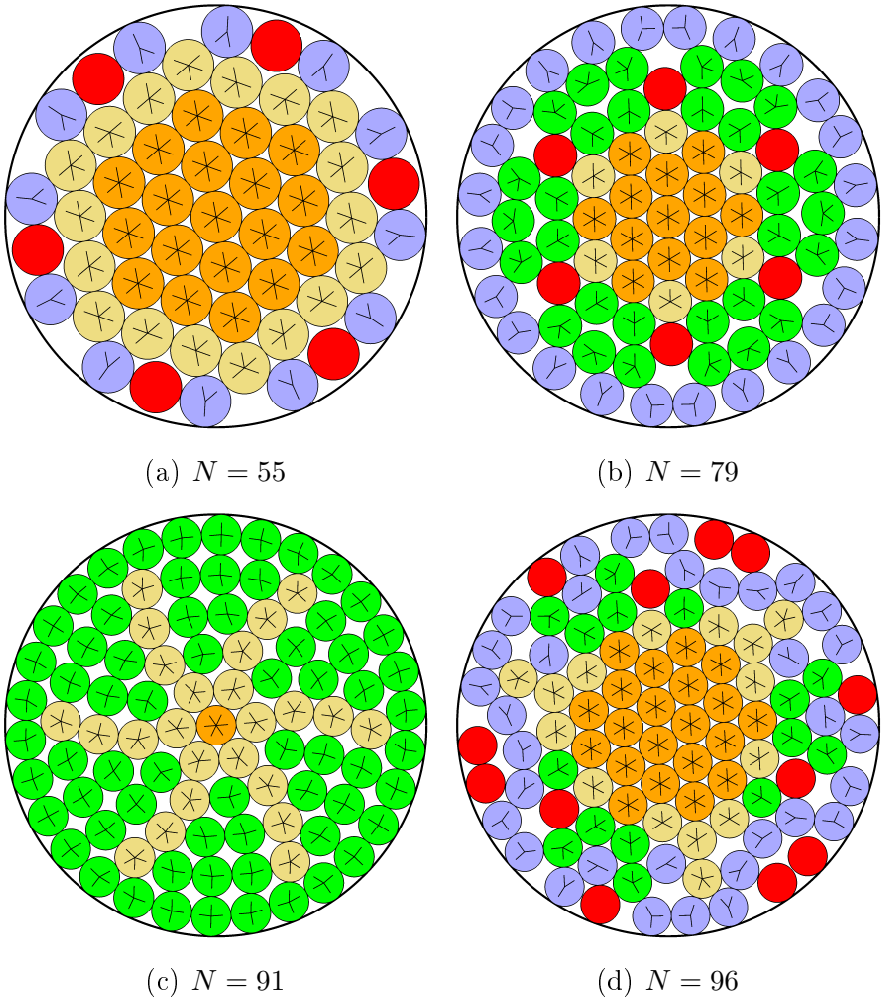
### *4.1 Parameter Settings and Experimental Protocol*

Table 1
Settings of parameters

| Parameters | Section | Description | Values |
|---|---|---|---|
| $\theta$ | 3.1 | coefficient used to adjust the radius of container | 0.7 |
| $\mu$ | 3.4 | coefficient used to tune the threshold value $Th_E$ | 0.75 |
| $MaxIter$ | 3.4 | maximum number of iterations of DTS procedure | $10^3$ |

The proposed IDTS algorithm employs three main parameters whose descriptions and default values are given in Table 1, which were empirically determined via a preliminary experiment. This setting can be considered as the default setting of the algorithm and was consistently used to conduct all computational experiments in Section 4. A sensibility analysis of these parameters is presented in Section 5.1.

The IDTS algorithm was implemented in the C++ language and complied using the g++ compiler with the -O3 option. The computational experiments with IDTS were executed on a computer with an Intel E5-2670 processor (2.5 GHz and 2G RAM), running the Linux operating system. To assess the performance of the algorithm, we employed 320 popular instances with $N \leq 320$. Due to its stochastic feature, the IDTS algorithm was independently performed 20 times with different random seeds (CPU time stamps) to solve each instance. The stopping condition for each run is a maximum time limit $t_{max}$: 2 hours for the instances with $N \leq 100$, 8 hours for $101 \leq N \leq 200$, and 12 hours for $201 \leq N \leq 320$. These cutoff limits are comparable with those used by the state-of-the-art algorithms in the literature. For example, in [27], $t_{max}$ was set to 4 hours for the instances with $N \leq 100$, and 8 hours for $101 \leq N \leq 200$. In [23], $t_{max}$ was set to 4 hours for $N \leq 100$, and 3 days for $N \geq 101$.



(a) $N = 55$ (b) $N = 79$

(c) $N = 91$ (d) $N = 96$

Fig. 3. Best configurations found by IDTS with a high symmetry for 4 representative instances in the range of $1 \leq N \leq 100$. The circles are colored by several colors according to the numbers of their neighbors, where two circles $c_i$ and $c_j$ are mutually considered as neighbors if and only if the distance $D(c_i, c_j)$ between their centers satisfies $D(c_i, c_j) < 2.0 + 10^{-10}$.

The first experiment focuses on the most studied (small) instances with $N \leq 100$ and shows a comparison with two best performing reference algorithms: the quasi-physical global optimization (QPGO) algorithm [27] and the quasi-physical quasi-human (QPQH) algorithm [23]. The results of the proposed IDTS algorithm on the 35 instances with $66 \leq N \leq 100$ are summarized in Table 2, together with the results of the reference algorithms. The instances with $N \leq 65$ are ignored, since they are very easy for the compared algorithms.

Columns 1 and 2 of Table 2 give respectively the number $N$ of circles and the best-known result $R^*$ (i.e., the known smallest container radius) reported

Table 2
Comparison between the IDTS algorithm and two state-of-the-art algorithms in the literature on 35 representative instances with $66 \leq N \leq 100$.

| | | QPGO [27] | | QPQH [23] | | IDTS (this work) | |
|---|---|---|---|---|---|---|---|
| $N$ | $R^*$ [23,47] | SR | $time(s)$ | SR | $time(s)$ | SR | $time(s)$ |
| 66 | 9.096279427 | 10/10 | 720 | 10/10 | 2404 | 20/20 | 20 |
| 67 | 9.168971882 | 10/10 | 1500 | 10/10 | 7492 | 20/20 | 51 |
| 68 | 9.229773747 | 10/10 | 720 | 10/10 | 3252 | 20/20 | 24 |
| 69 | 9.269761267 | 10/10 | <60 | 10/10 | 713 | 20/20 | 19 |
| 70 | 9.345653194 | 10/10 | 1500 | 10/10 | 1752 | 20/20 | 180 |
| 71 | 9.415796897 | 10/10 | 660 | 10/10 | 916 | 20/20 | 254 |
| 72 | 9.473890857 | 10/10 | 120 | 10/10 | 113 | 20/20 | 11 |
| 73 | 9.540346152 | 10/10 | 2520 | 2/20 | 5571 | 20/20 | 226 |
| 74 | 9.589232764 | 10/10 | <60 | 6/20 | 5343 | 20/20 | 292 |
| 75 | 9.672029632 | 10/10 | 2220 | 8/20 | 6671 | 20/20 | 112 |
| 76 | 9.729596802 | 10/10 | 180 | 10/10 | 464 | 20/20 | 22 |
| 77 | 9.798911925 | 10/10 | 840 | 10/10 | 147 | 20/20 | 18 |
| 78 | 9.857709900 | 10/10 | 1320 | 3/10 | 4276 | 20/20 | 63 |
| 79 | 9.905063468 | 10/10 | 240 | 10/10 | 5464 | 20/20 | 100 |
| 80 | 9.968151813 | 10/10 | 3900 | 5/10 | 5145 | 20/20 | 165 |
| 81 | 10.010864241 | 10/10 | 120 | 7/10 | 5446 | 20/20 | 53 |
| 82 | 10.050824224 | 10/10 | 600 | 0/10 | - | 20/20 | 78 |
| 83 | 10.116857875 | 10/10 | 960 | 7/10 | 4821 | 20/20 | 316 |
| 84 | 10.149530867 | 10/10 | 420 | 8/10 | 5364 | 20/20 | 87 |
| 85 | 10.163111466 | 10/10 | 1140 | 10/10 | 377 | 20/20 | 41 |
| 86 | 10.298701053 | 10/10 | <60 | 2/10 | 5867 | 20/20 | 318 |
| 87 | 10.363208505 | 10/10 | <60 | 10/10 | 1148 | 20/20 | 445 |
| 88 | 10.432337693 | 10/10 | 1560 | 10/10 | 4866 | 20/20 | 43 |
| 89 | 10.500491815 | 6/10 | 3600 | 5/10 | 7751 | 20/20 | 449 |
| 90 | 10.546069178 | 10/10 | 600 | 10/10 | 588 | 20/20 | 45 |
| 91 | 10.566772234 | 10/10 | <60 | 10/10 | 40 | 20/20 | 11 |
| 92 | 10.684645848 | 5/10 | 6720 | 4/10 | 4992 | 20/20 | 409 |
| 93 | 10.733352600 | 5/10 | 5880 | 3/10 | 6092 | 20/20 | 175 |
| 94 | 10.778032160 | 10/10 | 180 | 8/10 | 5552 | 20/20 | 135 |
| 95 | 10.840205022 | 7/10 | 7020 | 5/10 | 6056 | 20/20 | 110 |
| 96 | 10.883202760 | 2/10 | 7860 | 4/10 | 5824 | 20/20 | 2349 |
| 97 | 10.938590110 | 1/10 | 10680 | 1/10 | 11540 | 20/20 | 350 |
| 98 | 10.979383128 | 10/10 | 1800 | 9/10 | 2593 | 20/20 | 79 |
| 99 | 11.033141151 | 5/10 | 2640 | 9/10 | 3445 | 20/20 | 124 |
| 100 | 11.082149724 | 1/10 | 4560 | 0/10 | - | 20/20 | 172 |
| #Better | | 0 | 4 | 0 | 0 | | |
| #Equal | | 27 | 0 | 14 | 0 | | |
| #Worse | | 8 | 31 | 21 | 35 | | |

in the literature or shown on the Packomania website [47]. The remaining columns present the results of the three compared algorithms, respectively, including the success rate (SR) (i.e., the number of runs reaching the best-known result over the total number of runs) and the average computation time in seconds ($time(s)$) needed to reach the best-known result [1] . Note that QPGO was performed on a computer with an Intel Xeon processor and 32 Gb RAM and QPQH was performed on the Ali cloud platform (http://www.aliyun.com) with 8 Gb RAM. Since the compared algorithms were run on different computing platforms, timing information is shown only for indicative purposes. The symbol '-' means that the corresponding algorithm failed to reach the best-known result. The last rows '#Better', '#Equal', '#Worse' show the numbers

[1] The best packing configurations found in this work is available at `http://www.info.univ-angers.fr/pub/hao/circlepacking.html` and `https://github.com/XiangjingLai/circle-packing`.

of instances for which the corresponding reference algorithm obtained a better, equal, worse result compared to the proposed IDTS algorithm for each performance indicator. Finally, it is worth noting that the two reference algorithms used the best-known result $R^*$ prior to their work to seed their search, while the proposed IDTS algorithm started its search from scratch.

Table 2 shows that the proposed algorithm performs very well for these small instances with $N \leq 100$. Specifically, IDTS has a success rate of 100% for each tested instance with much shorter computation times. This is in sharp contrast with the reference algorithms whose success rates typically decrease with the increase of the problem size while requiring considerably more computation times to attain the best-known results.

For an intuitive presentation of the computational results, Fig. 3 shows the best packing configurations found by IDTS for four representative instances, disclosing some clear regular patterns (e.g., symmetry) for these instances.

### *4.3 Computational Results for the Larger Instances*

The second experiment aims to assess the proposed IDTS algorithm on large instances with $N \geq 101$ and up to $N = 320$ by making a comparison with the best-known results ($R^*$, the known smallest container radius) reported in the literature or shown on the Packomania website [47]. The results of the IDTS algorithm are summarized in Tables 3–6, respectively for $101 \leq N \leq 150$, $150 \leq N \leq 200$, $201 \leq N \leq 260$ and $261 \leq N \leq 320$.

The first column of each table gives the value of $N$ of the instance, and the second column shows the current best-known result $R^*$, which is compiled from the best results on the Packomania website [47] and the improved results in a recent paper [23]. The third and fourth columns give respectively the results of the Packomania website ($R_1$) and the improved results by QPQH [23] ($R_2$), while '-' means the corresponding result is not available. The results of the IDTS algorithm are reported in the last five columns, including the best result obtained over 20 independent runs ($R_{best}$), the average result ($R_{avg}$), the difference between $R_{best}$ and $R^*$ (i.e., $R_{best} - R^*$, so a negative value indicates an improved new best result), the success rate (SR) of hitting the best result $R_{best}$, and the average computation time for the algorithm to hit its best solution ($time(s)$). The rows '#Improve', '#Equal', '#Worse' at the bottom of tables show the numbers of instances for which the IDTS algorithm obtained an improved, equal, and worse result compared to the best-known result. In addition, to verify whether there exists a significant difference between our results and the best-known results $R^*$ both in terms of $R_{best}$ and $R_{avg}$, the last row of tables indicates the *p-values* from the Wilcoxon signed-rank tests,

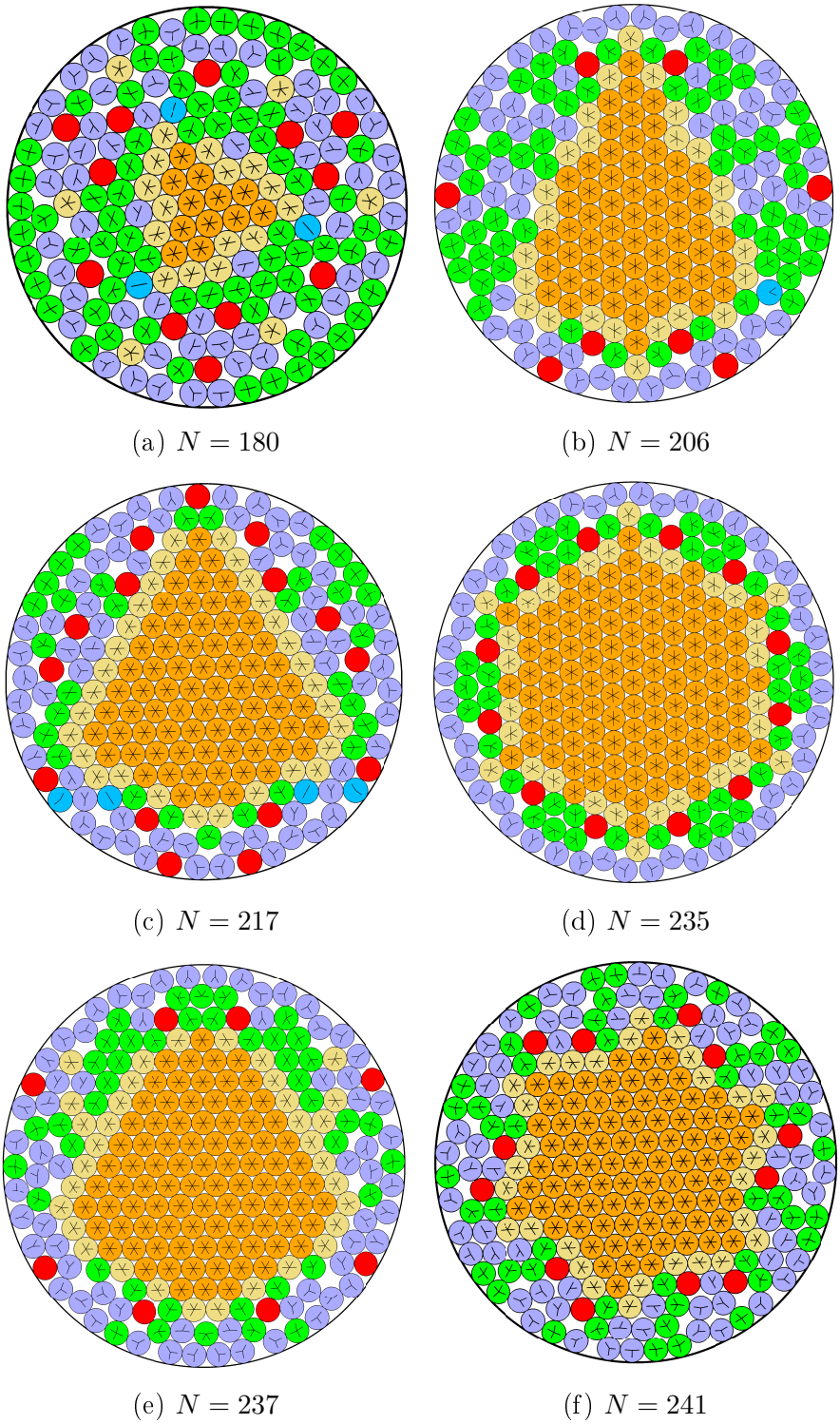
Table 3
Computational results and comparison on the 50 instances with $101 \leq N \leq 150$. The improved results are indicated in bold compared to the best-known results $R^*$ both in terms of $R_{best}$ and $R_{avg}$, and the worse results are indicated in italic.

| | | | | IDTS (this work) | | | | |
|---|---|---|---|---|---|---|---|---|
| $N$ | $R^*$ [23,47] | $R_1$ [47] | $R_2$ [23] | $R_{best}$ | $R_{avg}$ | $R_{best} - R^*$ | SR | $time(s)$ |
| 101 | 11.146933575 | 11.146933575 | - | 11.146933575 | 11.146933575 | 0.0 | 20/20 | 410 |
| 102 | 11.196863473 | 11.196863473 | - | 11.196863473 | 11.196863473 | 0.0 | 20/20 | 108 |
| 103 | 11.265143566 | 11.265143566 | - | 11.265143566 | 11.265143566 | 0.0 | 20/20 | 170 |
| 104 | 11.317658566 | 11.317658566 | - | 11.317658566 | 11.317658566 | 0.0 | 20/20 | 1698 |
| 105 | 11.362659613 | 11.362659613 | - | 11.362659613 | 11.362659613 | 0.0 | 20/20 | 100 |
| 106 | 11.421834366 | 11.421834366 | - | 11.421834366 | 11.421834366 | 0.0 | 20/20 | 466 |
| 107 | 11.472051837 | 11.472051837 | - | 11.472051837 | 11.472051837 | 0.0 | 20/20 | 770 |
| 108 | 11.524016134 | 11.524016134 | - | 11.524016134 | 11.524016134 | 0.0 | 20/20 | 411 |
| 109 | 11.562119071 | 11.562119071 | - | 11.562119071 | 11.562119071 | 0.0 | 20/20 | 2458 |
| 110 | 11.616861550 | 11.616861550 | - | 11.616861550 | 11.616861550 | 0.0 | 20/20 | 831 |
| 111 | 11.662811184 | 11.662811184 | - | 11.662811185 | 11.662811185 | 0.0 | 20/20 | 371 |
| 112 | 11.705274526 | 11.705274526 | - | 11.705274526 | 11.705274526 | 0.0 | 20/20 | 715 |
| 113 | 11.747528122 | 11.747528122 | - | 11.747528123 | 11.747528123 | 0.0 | 20/20 | 157 |
| 114 | 11.795173364 | 11.795173364 | - | 11.795173364 | 11.795173364 | 0.0 | 20/20 | 678 |
| 115 | 11.839474009 | 11.839474009 | - | 11.839474009 | 11.839474009 | 0.0 | 20/20 | 70 |
| 116 | 11.896704500 | 11.896704500 | - | 11.896704500 | 11.896704500 | 0.0 | 20/20 | 557 |
| 117 | 11.943475536 | 11.943475536 | - | 11.943475536 | 11.943475536 | 0.0 | 20/20 | 490 |
| 118 | 11.985551046 | 11.985551046 | - | 11.985551046 | 11.985551046 | 0.0 | 20/20 | 357 |
| 119 | 12.042334444 | 12.042334444 | - | 12.042334444 | 12.042334444 | 0.0 | 20/20 | 512 |
| 120 | 12.085212460 | 12.085212460 | - | 12.085212460 | 12.085212460 | 0.0 | 20/20 | 832 |
| 121 | 12.124803082 | 12.124803082 | - | 12.124803082 | 12.124803082 | 0.0 | 20/20 | 11 |
| 122 | 12.204374812 | 12.204374812 | - | 12.204374812 | 12.204374812 | 0.0 | 20/20 | 65 |
| 123 | 12.276399909 | 12.276399909 | - | 12.276399909 | 12.276399909 | 0.0 | 20/20 | 61 |
| 124 | 12.321708315 | 12.321708315 | - | 12.321708315 | 12.321708315 | 0.0 | 20/20 | 9244 |
| 125 | 12.368225321 | 12.368225321 | - | 12.368225321 | 12.368225321 | 0.0 | 20/20 | 98 |
| 126 | 12.417144417 | 12.417463956 | 12.417144417 | **12.417144392** | **12.417144392** | -2.52E-08 | 20/20 | 1609 |
| 127 | 12.461549515 | 12.461549515 | - | 12.461549515 | 12.461549515 | 0.0 | 20/20 | 2772 |
| 128 | 12.502222199 | 12.502310071 | 12.502222199 | 12.502222199 | 12.502222199 | 0.0 | 20/20 | 1830 |
| 129 | 12.553717819 | 12.553717819 | - | 12.553717819 | 12.553717819 | 0.0 | 20/20 | 2432 |
| 130 | 12.601774612 | 12.602318937 | 12.601774612 | **12.600339970** | **12.600339970** | -1.43E-03 | 20/20 | 3870 |
| 131 | 12.649620461 | 12.649620461 | - | 12.649620462 | 12.649620462 | 0.0 | 20/20 | 980 |
| 132 | 12.687436791 | 12.687436791 | - | 12.687436791 | 12.687436791 | 0.0 | 20/20 | 132 |
| 133 | 12.735273089 | 12.735273089 | - | 12.735273089 | 12.735273089 | 0.0 | 20/20 | 1152 |
| 134 | 12.771446240 | 12.771446240 | - | 12.771446240 | 12.771446240 | 0.0 | 20/20 | 413 |
| 135 | 12.814254771 | 12.814254771 | - | 12.814254772 | 12.814254772 | 0.0 | 20/20 | 1076 |
| 136 | 12.865759551 | 12.865759551 | - | 12.865759551 | 12.865759551 | 0.0 | 20/20 | 2874 |
| 137 | 12.914711247 | 12.914725417 | 12.914711247 | **12.914469900** | **12.914469900** | -2.41E-04 | 20/20 | 4888 |
| 138 | 12.961304417 | 12.962702608 | 12.961304417 | **12.961304311** | **12.961304311** | -1.06E-07 | 20/20 | 1360 |
| 139 | 13.008987241 | 13.008987241 | - | 13.008987241 | 13.008987241 | 0.0 | 20/20 | 1941 |
| 140 | 13.060696617 | 13.061097215 | 13.060696617 | **13.059777506** | **13.059841914** | -9.19E-04 | 4/20 | 15842 |
| 141 | 13.107255295 | 13.107255295 | - | 13.107255295 | 13.107255295 | 0.0 | 20/20 | 1328 |
| 142 | 13.146411626 | 13.146411626 | - | 13.146411626 | 13.146411626 | 0.0 | 20/20 | 314 |
| 143 | 13.197400825 | 13.197400825 | - | 13.197400825 | 13.197400825 | 0.0 | 20/20 | 288 |
| 144 | 13.247789225 | 13.247789225 | - | 13.247789225 | 13.247789225 | 0.0 | 20/20 | 3478 |
| 145 | 13.276668630 | 13.276668630 | - | 13.276668630 | *13.277790851* | 0.0 | 6/20 | 20834 |
| 146 | 13.331233768 | 13.331233768 | - | 13.331233768 | *13.331264325* | 0.0 | 3/20 | 7548 |
| 147 | 13.357112495 | 13.357112495 | - | 13.357112495 | 13.357112495 | 0.0 | 20/20 | 914 |
| 148 | 13.386939355 | 13.386939355 | - | 13.386939355 | 13.386939355 | 0.0 | 20/20 | 608 |
| 149 | 13.435548518 | 13.435548518 | - | 13.435548518 | 13.435548518 | 0.0 | 20/20 | 650 |
| 150 | 13.460806371 | 13.460806371 | - | 13.460806371 | 13.460806371 | 0.0 | 20/20 | 1289 |
| #Improve | | | | 5 | 5 | | | |
| #Equal | | | | 45 | 43 | | | |
| #Worse | | | | 0 | 2 | | | |
| *p-value* | | | | 1.35E-1 | 6.55E-1 | | | |

Table 4
Computational results and comparison on the 50 instances with $151 \leq N \leq 200$. The improved results are indicated in bold compared to the best-known results $R^*$ both in terms of $R_{best}$ and $R_{avg}$, and the worse results are indicated in italic.

| | | | | IDTS (this work) | | | | |
|---|---|---|---|---|---|---|---|---|
| $N$ | $R^*$ [23,47] | $R_1$ [47] | $R_2$ [23] | $R_{best}$ | $R_{avg}$ | $R_{best} - R^*$ | SR | $time(s)$ |
| 151 | 13.476191508 | 13.476191508 | - | 13.476191508 | 13.476191508 | 0.0 | 20/20 | 527 |
| 152 | 13.531758483 | 13.531758483 | - | **13.531748487** | **13.531748487** | -1.00E-05 | 20/20 | 7563 |
| 153 | 13.591536687 | 13.591536687 | - | 13.591536687 | 13.591536687 | 0.0 | 20/20 | 5774 |
| 154 | 13.637418860 | 13.637418860 | - | **13.636897671** | **13.636897671** | -5.21E-04 | 20/20 | 1208 |
| 155 | 13.673708640 | 13.673708640 | - | 13.673708640 | *13.673710758* | 0.0 | 7/20 | 10609 |
| 156 | 13.718404613 | 13.719600040 | 13.718404613 | **13.7163630490** | **13.716830585** | -2.04E-03 | 1/20 | 16652 |
| 157 | 13.772991900 | 13.772991900 | - | 13.772991900 | 13.772991900 | 0.0 | 20/20 | 7172 |
| 158 | 13.823225917 | 13.823225917 | - | 13.823225917 | 13.823225917 | 0.0 | 20/20 | 6504 |
| 159 | 13.864193589 | 13.864193589 | - | **13.864193368** | **13.864193368** | -2.21E-07 | 20/20 | 574 |
| 160 | 13.920451761 | 13.920538614 | 13.920451761 | **13.919646942** | **13.919810338** | -8.05E-04 | 6/20 | 14582 |
| 161 | 13.969400270 | 13.969400270 | - | 13.969400270 | 13.969400270 | 0.0 | 20/20 | 884 |
| 162 | 14.011328518 | 14.011328518 | - | **14.010172332** | **14.010217255** | -1.16E-03 | 3/20 | 14160 |
| 163 | 14.067635971 | 14.069620216 | 14.067635971 | **14.065203711** | **14.065216603** | -2.43E-03 | 5/20 | 12722 |
| 164 | 14.110700047 | 14.110700047 | - | **14.109601373** | **14.109601373** | -1.10E-03 | 20/20 | 6189 |
| 165 | 14.144339292 | 14.144339292 | - | 14.144339292 | *14.144406918* | 0.0 | 19/20 | 11525 |
| 166 | 14.185678420 | 14.185678420 | - | 14.185678420 | 14.185678420 | 0.0 | 20/20 | 3739 |
| 167 | 14.220394493 | 14.220394493 | - | **14.219046588** | **14.219046588** | -1.35E-03 | 20/20 | 4820 |
| 168 | 14.259535536 | 14.259535536 | - | **14.257706915** | **14.257710173** | -1.83E-03 | 17/20 | 11126 |
| 169 | 14.295358228 | 14.295358228 | - | 14.295358228 | 14.295358228 | 0.0 | 20/20 | 2837 |
| 170 | 14.331549723 | 14.331549723 | - | **14.331150274** | **14.331150274** | -3.99E-04 | 20/20 | 1187 |
| 171 | 14.367661448 | 14.367661448 | - | 14.367661448 | 14.367661448 | 0.0 | 20/20 | 2223 |
| 172 | 14.416776084 | 14.416776084 | - | 14.416776084 | 14.416776084 | 0.0 | 20/20 | 9646 |
| 173 | 14.451341436 | 14.451341436 | - | 14.451341436 | 14.451341436 | 0.0 | 20/20 | 1798 |
| 174 | 14.492848839 | 14.492848839 | - | 14.492848839 | 14.492848839 | 0.0 | 20/20 | 2137 |
| 175 | 14.536540602 | 14.536540602 | - | **14.536426508** | **14.536426508** | -1.14E-04 | 20/20 | 1176 |
| 176 | 14.574135655 | 14.574135655 | - | 14.574135655 | 14.574135655 | 0.0 | 20/20 | 189 |
| 177 | 14.614803186 | 14.617194155 | 14.614803186 | **14.614518392** | **14.614521438** | -2.85E-04 | 10/20 | 13471 |
| 178 | 14.655927628 | 14.658814224 | 14.655927628 | 14.655927628 | 14.655927628 | 0.0 | 20/20 | 2053 |
| 179 | 14.699982808 | 14.702293364 | 14.699982808 | **14.699851252** | **14.699851252** | -1.32E-04 | 20/20 | 2901 |
| 180 | 14.739035484 | 14.742878035 | 14.739035484 | **14.739034879** | **14.739034879** | -6.05E-07 | 20/20 | 3806 |
| 181 | 14.780829657 | 14.780829657 | - | 14.780829657 | 14.780829657 | 0.0 | 20/20 | 6091 |
| 182 | 14.818930631 | 14.818930631 | - | 14.818930631 | *14.823505353* | 0.0 | 8/20 | 9953 |
| 183 | 14.865788757 | 14.869399060 | 14.865788757 | **14.865670837** | **14.865670837** | -1.18E-04 | 20/20 | 2051 |
| 184 | 14.905472585 | 14.905472585 | - | **14.905418305** | *14.907479359* | -5.43E-05 | 1/20 | 9094 |
| 185 | 14.938144835 | 14.938144835 | - | 14.938144835 | *14.939607732* | 0.0 | 2/20 | 18103 |
| 186 | 14.961185685 | 14.961185685 | - | 14.961185685 | *14.961419453* | 0.0 | 18/20 | 15077 |
| 187 | 14.989269701 | 14.989269701 | - | **14.989106751** | *14.990022721* | -1.63E-04 | 9/20 | 20789 |
| 188 | 15.028750763 | 15.028782735 | 15.028750763 | **15.028750689** | *15.028752282* | -7.41E-08 | 19/20 | 9983 |
| 189 | 15.069680565 | 15.069680565 | - | **15.069099372** | **15.069404626** | -5.81E-04 | 10/20 | 12266 |
| 190 | 15.103601504 | 15.103601504 | - | 15.103601504 | 415.103601504 | 0.0 | 20/20 | 7537 |
| 191 | 15.142357530 | 15.142357530 | - | **15.142357503** | **15.142357505** | -2.65E-08 | 3/20 | 4865 |
| 192 | 15.169435212 | 15.169435212 | - | 15.169435212 | 15.169435212 | 0.0 | 20/20 | 634 |
| 193 | 15.199374495 | 15.199374495 | - | 15.199374495 | *15.199379498* | 0.0 | 4/20 | 13820 |
| 194 | 15.241428667 | 15.249753585 | 15.241428667 | **15.241428657** | **15.241428657** | -1.00E-08 | 20/20 | 2541 |
| 195 | 15.287172168 | 15.287172168 | - | 15.287172168 | *15.287684736* | 0.0 | 1/20 | 13215 |
| 196 | 15.323079829 | 15.323079829 | - | **15.322999997** | **15.323021340** | -7.98E-05 | 17/20 | 12829 |
| 197 | 15.368518430 | 15.368975295 | 15.368518430 | **15.367497382** | **15.367599851** | -1.02E-03 | 1/20 | 5607 |
| 198 | 15.390410683 | 15.391207855 | 15.390410683 | **15.389903201** | **15.389903201** | -5.07E-04 | 18/20 | 10319 |
| 199 | 15.402260415 | 15.402260415 | - | 15.402260415 | 15.402260415 | 0.0 | 20/20 | 224 |
| 200 | 15.463274879 | 15.463274879 | - | 15.463274879 | 15.463274879 | 0.0 | 20/20 | 1632 |
| #Improve | | | | 25 | 22 | | | |
| #Equal | | | | 25 | 18 | | | |
| #Worse | | | | 0 | 10 | | | |
| *p-value* | | | | 9.34E-6 | 1.66E-1 | | | |

where a *p-value* less than 0.05 means that there exists a statistically significant difference between the compared results.



(a) $N = 180$ (b) $N = 206$

(c) $N = 217$ (d) $N = 235$

(e) $N = 237$ (f) $N = 241$

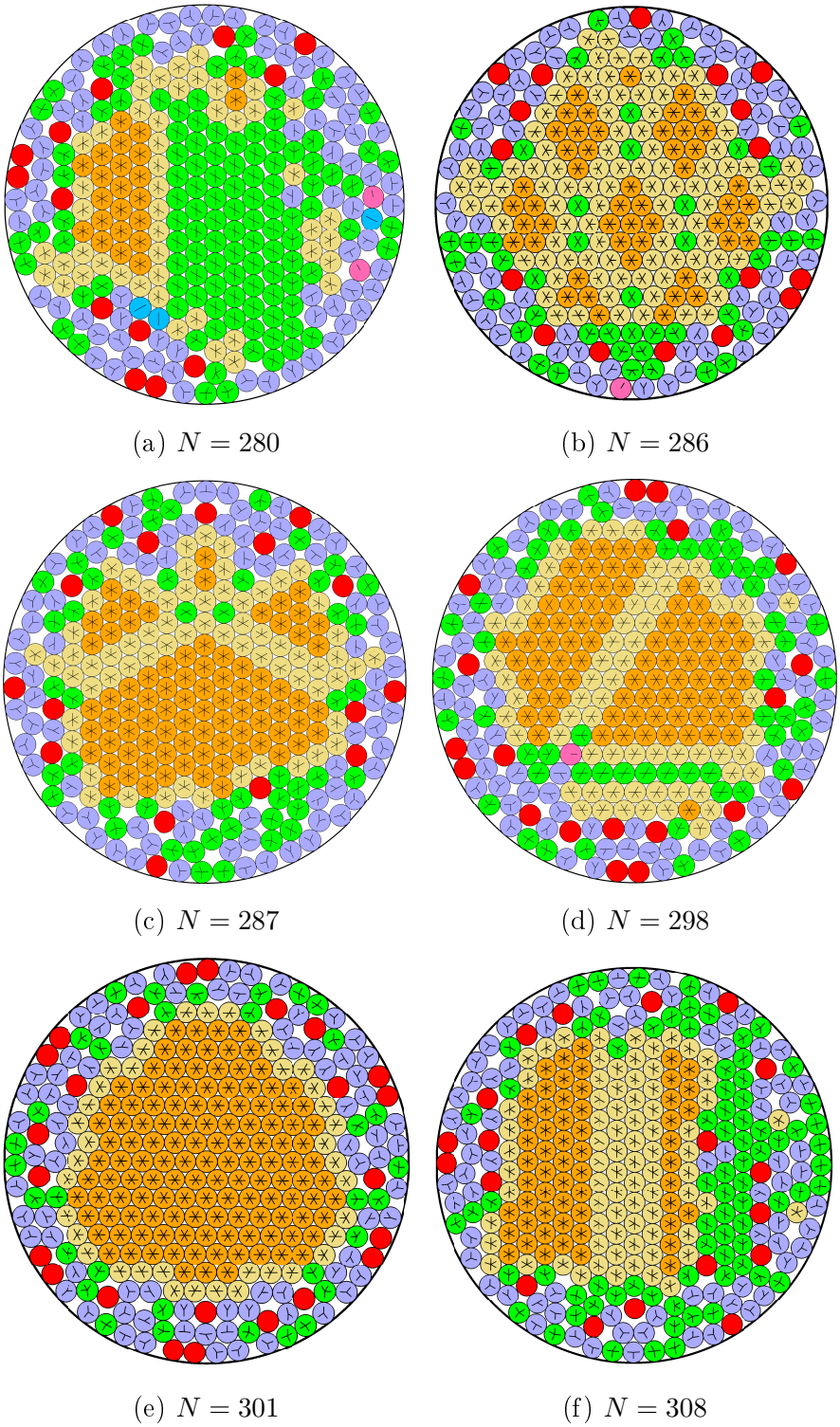
Fig. 4. Packing configurations with a high symmetry for some selected instances in the range of $101 \leq N \leq 320$.

Table 5
Computational results and comparison on the 60 instances with $201 \leq N \leq 260$. The improved results are indicated in bold compared to the best-known results $R^*$ both in terms of $R_{best}$ and $R_{avg}$, and the worse results are indicated in italic.

| | | | | IDTS (this work) | | | | |
|---|---|---|---|---|---|---|---|---|
| $N$ | $R^*$ [23,47] | $R_1$ [47] | $R_2$ [23] | $R_{best}$ | $R_{avg}$ | $R_{best} - R^*$ | SR | $time(s)$ |
| 201 | 15.520419047 | 15.520419047 | - | **15.520225293** | **15.520225293** | -1.94E-04 | 20/20 | 8240 |
| 202 | 15.569845914 | 15.569845914 | - | **15.568028837** | **15.568090487** | -1.82E-03 | 9/20 | 23342 |
| 203 | 15.612567145 | 15.612567145 | - | **15.611235518** | **15.611235518** | -1.33E-03 | 20/20 | 9479 |
| 204 | 15.644738394 | 15.652649998 | 15.644738394 | **15.642484085** | **15.642522181** | -2.25E-03 | 1/20 | 18445 |
| 205 | 15.695538521 | 15.709043666 | 15.695538521 | **15.690988269** | **15.691079511** | -4.55E-03 | 5/20 | 22144 |
| 206 | 15.736799931 | 15.736799931 | - | **15.734557511** | *15.736968076* | -2.24E-03 | 1/20 | 22321 |
| 207 | 15.770190573 | 15.770271664 | 15.770190573 | **15.770190569** | *15.773476577* | -3.80E-09 | 3/20 | 21559 |
| 208 | 15.811985301 | 15.811985301 | - | **15.811764086** | **15.811852669** | -2.21E-04 | 12/20 | 20214 |
| 209 | 15.844267752 | 15.844267752 | - | **15.839871358** | **15.840916273** | -4.40E-03 | 3/20 | 30942 |
| 210 | 15.879201277 | 15.879201277 | - | **15.878779424** | **15.878800075** | -4.22E-04 | 18/20 | 9303 |
| 211 | 15.902512487 | 15.902512487 | - | 15.902512487 | 15.902512487 | 0.0 | 20/20 | 6491 |
| 212 | 15.937421635 | 15.937421635 | - | **15.935369099** | **15.935369099** | -2.05E-03 | 20/20 | 5263 |
| 213 | 15.969855988 | 15.970256294 | 15.969855988 | **15.969855941** | **15.969855941** | -4.70E-08 | 20/20 | 3536 |
| 214 | 16.018386421 | 16.018763220 | 16.018386421 | **16.017995815** | **16.017995815** | -3.91E-04 | 20/20 | 18952 |
| 215 | 16.050828504 | 16.050828504 | - | **16.049306210** | **16.050205425** | -1.52E-03 | 3/20 | 26274 |
| 216 | 16.087017095 | 16.087407652 | 16.087017095 | **16.085154499** | **16.085298637** | -1.86E-03 | 8/20 | 21517 |
| 217 | 16.118237367 | 16.119370348 | 16.118237367 | **16.116693296** | **16.117369938** | -1.54E-03 | 6/20 | 21763 |
| 218 | 16.151019810 | 16.151019810 | - | **16.149552938** | **16.149653633** | -1.47E-03 | 12/20 | 24221 |
| 219 | 16.169155113 | 16.169155113 | - | **16.169155098** | *16.171217331* | -1.49E-08 | 1/20 | 26209 |
| 220 | 16.225373549 | 16.225373549 | - | **16.224094825** | **16.224134520** | -1.28E-03 | 18/20 | 13400 |
| 221 | 16.261873763 | 16.261873985 | 16.261873763 | **16.258834346** | **16.259426290** | -3.04E-03 | 1/20 | 26332 |
| 222 | 16.299162417 | 16.299696226 | 16.299162417 | **16.298789347** | **16.298897565** | -3.73E-04 | 1/20 | 23040 |
| 223 | 16.337822305 | 16.337822305 | - | **16.337166598** | **16.337327474** | -6.56E-04 | 3/20 | 19201 |
| 224 | 16.369591221 | 16.371899241 | 16.369591221 | **16.368831685** | **16.369065419** | -7.60E-04 | 1/20 | 20358 |
| 225 | 16.408410528 | 16.409054279 | 16.408410528 | **16.403603782** | **16.403983211** | -4.81E-03 | 8/20 | 19930 |
| 226 | 16.449824611 | 16.450019502 | 16.449824611 | **16.448318718** | **16.448320646** | -1.51E-03 | 18/20 | 18076 |
| 227 | 16.489753739 | 16.494400009 | 16.489753739 | **16.488369922** | **16.488433032** | -1.38E-03 | 10/20 | 23388 |
| 228 | 16.527071189 | 16.527340870 | 16.527071189 | **16.524998071** | **16.525000425** | -2.07E-03 | 14/20 | 21565 |
| 229 | 16.564350195 | 16.566377981 | 16.564350195 | **16.562664605** | **16.562714547** | -1.69E-03 | 7/20 | 22247 |
| 230 | 16.596246697 | 16.596430072 | 16.596246697 | **16.592564989** | **16.594176903** | -3.68E-03 | 3/20 | 26229 |
| 231 | 16.631031907 | 16.640078327 | 16.631031907 | **16.629060933** | **16.629325490** | -1.97E-03 | 1/20 | 25846 |
| 232 | 16.669457196 | 16.669457196 | - | **16.664704645** | **16.668814250** | -4.75E-03 | 1/20 | 23617 |
| 233 | 16.694829942 | 16.694829942 | - | *16.696155400* | *16.700043062* | 1.33E-03 | 1/20 | 27503 |
| 234 | 16.706335883 | 16.706335883 | - | *16.706344401* | *16.712776770* | 8.52E-06 | 1/20 | 31503 |
| 235 | 16.712575786 | 16.712575786 | - | 16.712575786 | *16.728429055* | 0.0 | 1/20 | 33291 |
| 236 | 16.774738750 | 16.774738750 | - | **16.774392488** | **16.774678757** | -3.46E-04 | 2/20 | 26494 |
| 237 | 16.802145503 | 16.802145503 | - | **16.801582860** | *16.802332489* | -5.63E-04 | 1/20 | 18079 |
| 238 | 16.827113437 | 16.827113437 | - | **16.826908995** | **16.826910824** | -2.04E-04 | 19/20 | 14576 |
| 239 | 16.863843301 | 16.863843301 | - | **16.863505622** | **16.863507770** | -3.38E-04 | 18/20 | 12708 |
| 240 | 16.897165895 | 16.897165895 | - | **16.894472076** | **16.894472076** | -2.69E-03 | 20/20 | 7648 |
| 241 | 16.915064584 | 16.915064584 | - | 16.915064584 | *16.915349265* | 0.0 | 1/20 | 17897 |
| 242 | 16.961287246 | 16.962132986 | 16.961287246 | **16.960708659** | **16.960708660** | -5.79E-04 | 9/20 | 17587 |
| 243 | 17.001947599 | 17.004065250 | 17.001947599 | **17.000247953** | **17.000248010** | -1.70E-03 | 10/20 | 12620 |
| 244 | 17.039559451 | 17.039719464 | 17.039559451 | **17.034806378** | **17.034896401** | -4.75E-03 | 4/20 | 25380 |
| 245 | 17.078003394 | 17.079365818 | 17.078003394 | **17.074689206** | **17.074689206** | -3.31E-03 | 20/20 | 5873 |
| 246 | 17.113112319 | 17.113998222 | 17.113112319 | **17.107922480** | **17.107922480** | -5.19E-03 | 20/20 | 5679 |
| 247 | 17.132526683 | 17.141082424 | 17.132526683 | **17.132068945** | **17.132378242** | -4.58E-04 | 12/20 | 21726 |
| 248 | 17.182444113 | 17.184447103 | 17.182444113 | **17.182167927** | **17.182167998** | -2.76E-04 | 18/20 | 17176 |
| 249 | 17.220431735 | 17.220431735 | - | **17.219002124** | **17.219002124** | -1.43E-03 | 20/20 | 5728 |
| 250 | 17.262962239 | 17.262962239 | - | **17.261954907** | **17.262287793** | -1.01E-03 | 1/20 | 21907 |
| 251 | 17.297607156 | 17.305245616 | 17.297607156 | **17.295389898** | **17.295601630** | -2.22E-03 | 1/20 | 22029 |
| 252 | 17.326883903 | 17.331245569 | 17.326883903 | **17.325195900** | **17.325608574** | -1.69E-03 | 2/20 | 21310 |
| 253 | 17.345956323 | 17.345956323 | - | *17.346089752* | *17.346171694* | 1.33E-04 | 1/20 | 17275 |
| 254 | 17.400641308 | 17.400734609 | 17.400641308 | **17.393763187** | **17.394169348** | -6.88E-03 | 1/20 | 25791 |
| 255 | 17.454058463 | 17.454200693 | 17.454058463 | **17.444011350** | **17.444080914** | -1.00E-02 | 16/20 | 20552 |
| 256 | 17.493101490 | 17.494310641 | 17.493101490 | **17.486251304** | **17.487129555** | -6.85E-03 | 3/20 | 24549 |
| 257 | 17.523849714 | 17.523849714 | - | **17.517964918** | **17.519827002** | -5.88E-03 | 1/20 | 27861 |
| 258 | 17.547085193 | 17.547880529 | 17.547085193 | **17.538235776** | **17.539101690** | -8.85E-03 | 3/20 | 33153 |
| 259 | 17.580456185 | 17.580456185 | - | **17.573954655** | **17.575365809** | -6.50E-03 | 1/20 | 30320 |
| 260 | 17.604955193 | 17.604955193 | - | **17.600835484** | **17.600906404** | -4.12E-03 | 6/20 | 25377 |
| #Improve | | | | 54 | 51 | | | |
| #Equal | | | | 3 | 1 | | | |
| #Worse | | | | 3 | 8 | | | |
| *p-value* | | | | 2.61E-10 | 4.78E-6 | | | |

Table 6
Computational results and comparison on the 60 instances with $261 \leq N \leq 320$. The improved results are indicated in bold compared to the best-known results $R^*$ both in terms of $R_{best}$ and $R_{avg}$, and the worse results are indicated in italic.

| | | | | IDTS (this work) | | | | |
|---|---|---|---|---|---|---|---|---|
| $N$ | $R^*$ [23,47] | $R_1$ [47] | $R_2$ [23] | $R_{best}$ | $R_{avg}$ | $R_{best} - R^*$ | SR | $time(s)$ |
| 261 | 17.634766878 | 17.634862233 | 17.634766878 | **17.627240672** | **17.628625372** | -7.53E-03 | 1/20 | 28424 |
| 262 | 17.662924961 | 17.662924961 | - | **17.657906787** | **17.661282499** | -5.02E-03 | 1/20 | 27050 |
| 263 | 17.693359550 | 17.693359550 | - | **17.688613836** | **17.689565543** | -4.75E-03 | 1/20 | 24287 |
| 264 | 17.701373926 | 17.701373926 | - | 17.701373926 | *17.705237062* | 0.0 | 12/20 | 23728 |
| 265 | 17.741538902 | 17.741538902 | - | **17.741224763** | *17.741706383* | -3.14E-04 | 7/20 | 30347 |
| 266 | 17.778852591 | 17.778852591 | - | **17.776086768** | **17.777217033** | -2.77E-03 | 1/20 | 24801 |
| 267 | 17.797657056 | 17.797657056 | - | **17.797361304** | **17.797538147** | -2.96E-04 | 1/20 | 25569 |
| 268 | 17.832086714 | 17.832086714 | - | **17.832068508** | *17.8320934 28* | -1.82E-05 | 1/20 | 22276 |
| 269 | 17.863552746 | 17.863552746 | - | **17.862579267** | **17.862674626** | -9.73E-04 | 10/20 | 25078 |
| 270 | 17.887265668 | 17.887265668 | - | 17.887265668 | *17.890234807* | 0.0 | 1/20 | 27231 |
| 271 | 17.930192221 | 17.930192221 | - | **17.929693883** | **17.929914822** | -4.98E-04 | 3/20 | 23896 |
| 272 | 17.961933270 | 17.961933270 | - | **17.959189722** | **17.959347069** | -2.74E-03 | 6/20 | 22175 |
| 273 | 17.996459047 | 17.996459047 | - | **17.995299347** | **17.995315566** | -1.16E-03 | 15/20 | 24635 |
| 274 | 18.034743806 | 18.034743806 | - | **18.033698365** | **18.033832315** | -1.05E-03 | 1/20 | 19498 |
| 275 | 18.064257143 | 18.064257143 | - | **18.062653689** | **18.062808429** | -1.60E-03 | 4/20 | 25389 |
| 276 | 18.106737167 | 18.106737167 | - | **18.103416474** | **18.103756760** | -3.32E-03 | 3/20 | 24124 |
| 277 | 18.139629995 | 18.142486517 | 18.139629995 | **18.136079464** | **18.136758887** | -3.55E-03 | 1/20 | 22376 |
| 278 | 18.187675017 | 18.189804499 | 18.187675017 | **18.180555975** | **18.180925160** | -7.12E-03 | 1/20 | 26926 |
| 279 | 18.221572693 | 18.224068629 | 18.221572693 | **18.217568448** | **18.218195726** | -4.00E-03 | 1/20 | 29421 |
| 280 | 18.247226743 | 18.247226743 | - | **18.245927474** | **18.246904727** | -1.30E-03 | 1/20 | 30016 |
| 281 | 18.281539686 | 18.284504843 | 18.281539686 | **18.280408576** | **18.280832129** | -1.13E-03 | 1/20 | 28344 |
| 282 | 18.309334921 | 18.315754346 | 18.309334921 | **18.308789945** | *18.310816728* | -5.45E-04 | 1/20 | 22753 |
| 283 | 18.340942447 | 18.340942447 | - | **18.340236575** | *18.341407581* | -7.06E-04 | 1/20 | 31114 |
| 284 | 18.359456641 | 18.359456641 | - | *18.361020810* | *18.366038322* | 1.56E-03 | 1/20 | 34313 |
| 285 | 18.414548614 | 18.414548614 | - | **18.402678595** | **18.405461625** | -1.19E-02 | 1/20 | 25875 |
| 286 | 18.434157792 | 18.434315259 | 18.434157792 | **18.429241763** | **18.431535841** | -4.92E-03 | 1/20 | 30801 |
| 287 | 18.468823749 | 18.468823749 | - | **18.468212077** | *18.468883851* | -6.12E-04 | 1/20 | 23803 |
| 288 | 18.494844941 | 18.494844941 | - | **18.494381728** | *18.496038151* | -4.63E-04 | 1/20 | 28802 |
| 289 | 18.511602090 | 18.511602090 | - | *18.511810011* | *18.513773621* | 2.08E-04 | 1/20 | 28047 |
| 290 | 18.549375070 | 18.549375070 | - | **18.548707587** | **18.549023499** | -6.67E-04 | 2/20 | 19030 |
| 291 | 18.567351454 | 18.567351454 | - | **18.566314463** | **18.567300968** | -1.04E-03 | 2/20 | 27074 |
| 292 | 18.597179792 | 18.597179792 | - | **18.594501090** | **18.595234278** | -2.68E-03 | 1/20 | 29822 |
| 293 | 18.623366700 | 18.623366700 | - | *18.623599730* | *18.624720565* | 2.33E-04 | 1/20 | 20991 |
| 294 | 18.646248051 | 18.646248051 | - | **18.644849697** | **18.645597816** | -1.40E-03 | 1/20 | 28715 |
| 295 | 18.655217522 | 18.655217522 | - | *18.655996666* | *18.658527712* | 7.79E-04 | 1/20 | 24794 |
| 296 | 18.703338597 | 18.704963368 | 18.703338597 | **18.702750006** | **18.703029204** | -5.89E-04 | 1/20 | 24419 |
| 297 | 18.731053485 | 18.731053485 | - | **18.729781499** | **18.730183831** | -1.27E-03 | 2/20 | 30727 |
| 298 | 18.759535176 | 18.759535176 | - | **18.759344750** | *18.759574994* | -1.90E-04 | 4/20 | 23969 |
| 299 | 18.786282897 | 18.786282897 | - | **18.785427812** | **18.785802880** | -8.55E-04 | 3/20 | 27798 |
| 300 | 18.813583364 | 18.813583364 | - | **18.813153706** | **18.813180138** | -4.30E-04 | 6/20 | 21553 |
| 301 | 18.843675327 | 18.843979274 | 18.843675327 | **18.843463507** | **18.843586628** | -2.12E-04 | 4/20 | 24968 |
| 302 | 18.892376640 | 18.895682962 | 18.892376640 | **18.891782255** | **18.892069366** | -5.94E-04 | 1/20 | 26223 |
| 303 | 18.935589502 | 18.936286113 | 18.935589502 | **18.929749153** | **18.930127041** | -5.84E-03 | 1/20 | 25122 |
| 304 | 18.972060389 | 18.973327183 | 18.972060389 | **18.964441751** | **18.964787942** | -7.62E-03 | 1/20 | 24562 |
| 305 | 19.008914111 | 19.010908936 | 19.008914111 | **19.001754565** | **19.002296693** | -7.16E-03 | 1/20 | 26941 |
| 306 | 19.033069424 | 19.033069424 | - | **19.030389407** | **19.031089900** | -2.68E-03 | 1/20 | 20928 |
| 307 | 19.063881602 | 19.063881602 | - | **19.060160922** | **19.061046757** | -3.72E-03 | 1/20 | 22978 |
| 308 | 19.120984457 | 19.121680199 | 19.120984457 | **19.104991437** | **19.108690710** | -1.60E-02 | 1/20 | 23202 |
| 309 | 19.146749239 | 19.146749239 | - | **19.142573165** | **19.144543174** | -4.18E-03 | 1/20 | 32491 |
| 310 | 19.184859463 | 19.184859463 | - | **19.178928265** | **19.181213383** | -5.93E-03 | 1/20 | 28042 |
| 311 | 19.211386405 | 19.211386405 | - | *19.212365036* | *19.214696924* | 9.79E-04 | 1/20 | 20635 |
| 312 | 19.234773095 | 19.234773095 | - | **19.233585653** | *19.236023545* | -1.19E-03 | 1/20 | 31080 |
| 313 | 19.259264009 | 19.259264009 | - | **19.257103014** | *19.259829708* | -2.16E-03 | 1/20 | 28332 |
| 314 | 19.288083255 | 19.288083255 | - | **19.286195141** | **19.286677783** | -1.89E-03 | 2/20 | 31615 |
| 315 | 19.302300677 | 19.302300677 | - | **19.302288067** | *19.303378383* | -1.26E-05 | 2/20 | 24985 |
| 316 | 19.335029425 | 19.335029425 | - | **19.334041754** | *19.335453673* | -9.88E-04 | 2/20 | 30840 |
| 317 | 19.375211471 | 19.375211471 | - | **19.367595672** | *19.368188858* | -7.62E-03 | 1/20 | 20441 |
| 318 | 19.396538989 | 19.407004242 | 19.396538989 | **19.391566091** | **19.392633602** | -4.97E-03 | 2/20 | 30334 |
| 319 | 19.432246442 | 19.432992566 | 19.432246442 | **19.424277830** | **19.425905156** | -7.97E-03 | 6/20 | 24676 |
| 320 | 19.451309906 | 19.456230764 | 19.451309906 | *19.451649630* | *19.453763706* | 3.40E-04 | 1/20 | 24894 |
| #Improve | | | | 52 | 40 | | | |
| #Equal | | | | 2 | 0 | | | |
| #Worse | | | | 6 | 20 | | | |
| *p-value* | | | | 3.39E-9 | 2.09E-3 | | | |



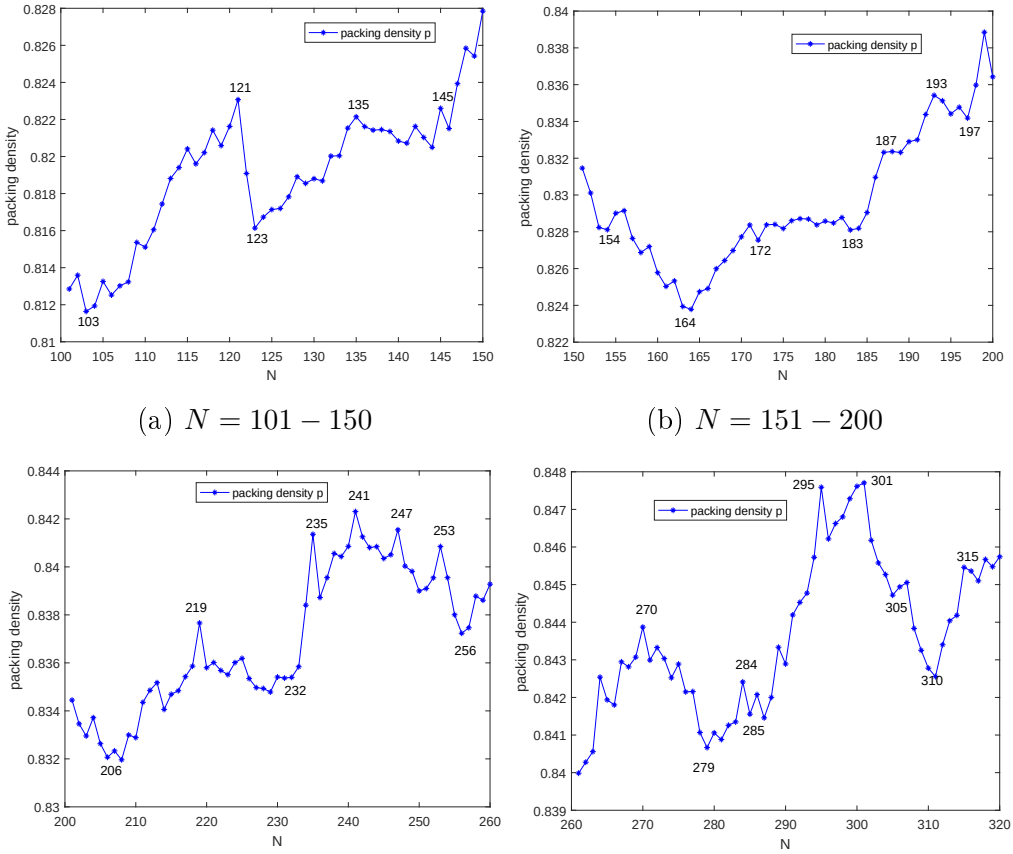
(a) $N = 280$ (b) $N = 286$

(c) $N = 287$ (d) $N = 298$

(e) $N = 301$ (f) $N = 308$

Fig. 5. Best packing configurations found in this work for some representative instances in the range of $101 \leq N \leq 320$.

From these results, we can make the following observations.




(a) $N = 101 - 150$ (b) $N = 151 - 200$

(c) $N = 201 - 260$ (d) $N = 261 - 320$

Fig. 6. Packing density plots for the current best known solutions.

Table 7
Summary of computational results

| $N$ | #Instance | #Improve | #Equal | #Worse |
|---|---|---|---|---|
| 101-150 | 50 | 5 | 45 | 0 |
| 151-200 | 50 | 25 | 25 | 0 |
| 201-260 | 60 | 54 | 3 | 3 |
| 261-320 | 60 | 52 | 2 | 6 |
| Total | 220 | 136 | 75 | 9 |

(1) For the 50 instances with $101 \leq N \leq 150$ (Table 3), IDTS improved and matched the best-known results for 5 and 45 instances, respectively. The algorithm obtained its best results ($R_{best}$) with a perfect success rate of 100% except for 3 instances.

(2) For the 50 instances with $151 \leq N \leq 200$ (Table 4), IDTS improved the best-known results for half of these instances while matching the best-known results for the other instances. The small *p-value* (9.43e-6) means that the difference between the best results of the IDTS algorithm and the best-known results is significant. On the other hand, the algorithm reached a perfect success rate of 100% for 29 instances. For 12 instances, the success rate for the algorithm to hit its best results drops to less than

10/20, indicating the increasing difficulty of these instances. In addition, it is worth noting that even the average result $R_{avg}$ of the IDTS algorithm is better than the best-known result $R^*$ for 22 out of 50 instances, implying a strong searching ability of the algorithm.

(3) For the 120 largest instances with $201 \leq N \leq 320$ (Tables 5 and 6), IDTS improved the best-known results for 106 instances, matched the best-known result for 5 instances, and missed the best-known result only for 9 instances. Moreover, the average result of the IDTS algorithm ($R_{avg}$) is better than the best-known result ($R^*$) for 91 out 120 instances. The small *p-values* show that both in terms of $R_{best}$ and $R_{avg}$ there exist a significant difference between the results of the IDTS algorithm and the best-known results. Moreover, for these largest instances, the success rate for the algorithm to hit its best results continues to decrease, confirming that they are the most difficult to solve among all tested instances.

Table 7 summarizes the results achieved by the IDTS algorithm on the 4 sets of instances, where the first column indicates the range of instances, the second column gives the number of instances in the associated set, and the last three columns show the numbers of instances for which the IDTS algorithm obtained an improved, equal, worse result compared with the best-known result $R^*$. The last row shows the total number of instances for each column. We observe from Table 7 that the proposed IDTS algorithm improved 136 best-known results for the 220 instances with $101 \leq N \leq 320$, matched the best-known result for 75 instances, and missed the best-known result only for 9 instances. These results demonstrate the high competitiveness of the proposed IDTS algorithm compared with the state-of-the-art methods.

For an intuitive presentation of the computational results, we illustrate the best packing configurations found for some representative instances in the range of $101 \leq N \leq 320$ in Fig. 4 (with clear regular patterns) and Fig. 5 (with less clear regular patterns).

Furthermore, to observe the packing density of the neighboring instances in size (i.e., the number of circles $N$), we plotted in Fig. 6 the evolution of packing density of the current best known configurations as a function of the instance size ($N$) for the 4 sets of instances. The packing density $p$ of a feasible packing configuration is calculated as $p = \frac{N\pi r^2}{\pi R^2}$, where $R$ represents the container radius and $r$ is the radius of packed circle ($r = 1$ for unit circles). One observes from Fig. 6 that there exit a number of instances for which the best-known solution has a particularly high or low packing density compared with the neighboring instances in size.

Finally, to have an idea about the capacity of our approach for dealing with very large instances ($N > 1000$), we tested IDTS on 17 instances with $N = 1077-1080, 1090-1092, 1094, 1096, 1099, 1100, 1200, 1300, 1500, 3000, 4000, 5000$.

Interestingly, compared to the best-known results for these instances at the Packomania website [47], which were achieved by the IIPP-random/lattice-IPOPT algorithm [49] under unknown conditions, IDTS can improve 7 best-known results ($N = 1080, 1094, 1096, 1099, 1100, 1200, 1300$). Meanwhile, we recognize that IDTS requires a very high computation time (2 to 5.7 days) to converge to its best solutions for these instances. In [49], it is indicated that the lattice-based initialization is critical for their IIPP-random/lattice-IPOPT algorithm. Thus, as a research perspective, it would be interesting to investigate this idea as well as other local optimization methods within our IDTS approach to accelerate the algorithm.

## 5 Analysis

In this section, we analyze several important elements of the proposed algorithm including the two-phase local optimization method (Section 3.2) and the sensitivity of the settings of parameters on the performance of the algorithm.

### *5.1 Sensitivity Analysis of Parameters*

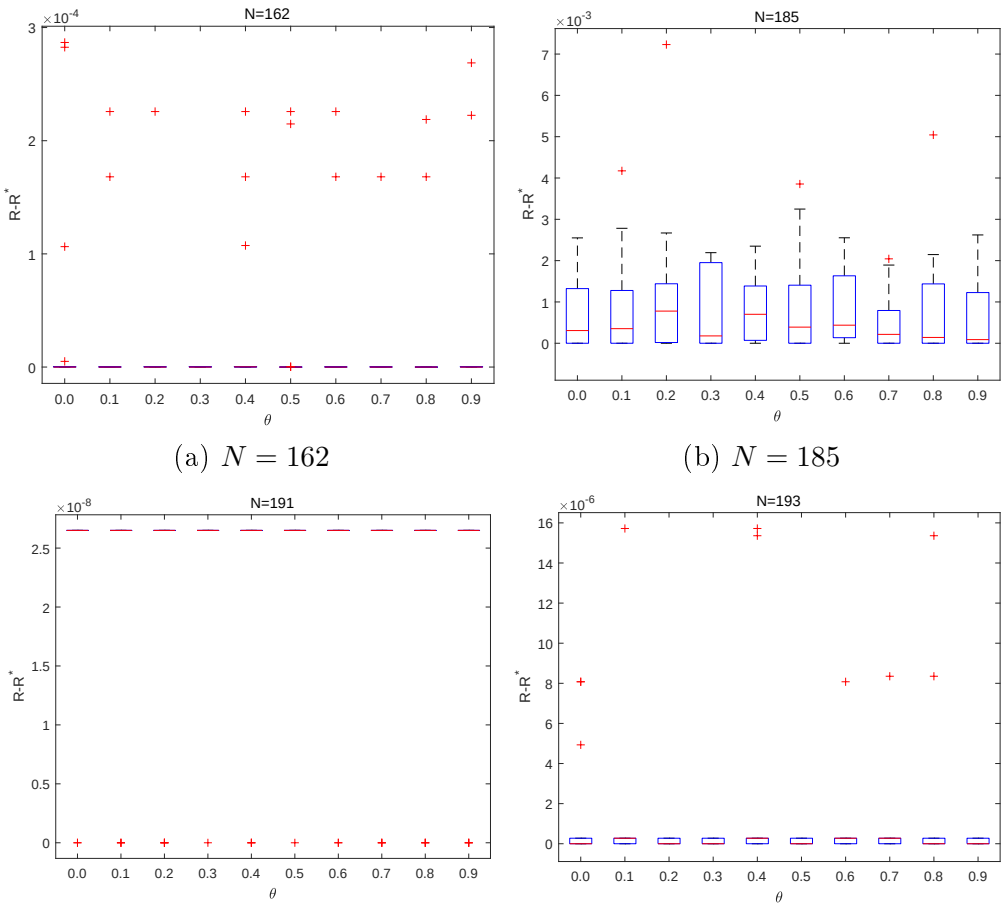
The proposed algorithm employs three main parameters ($\theta$, $MaxIter$ and $\mu$). This section discusses their influences on the algorithm.

#### *5.1.1 Sensitivity Analysis of Parameters $\theta$ and $MaxIter$*

The scaling factor $\theta$ of $\Delta_R$ (line 7 of Algorithm 1) and the maximum number of iterations ($MaxIter$) for each run of the dynamic thresholding search procedure are two important parameters of the IDTS algorithm. To show their sensitivity on the algorithm, we carried out two additional experiments on four representative instances with $N = 162$, 185, 191 and 193.

The first experiment aims to analyze the sensitivity of parameter $\theta$. For this, we varied $\theta$ in the range of $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, and performed the IDTS algorithm 20 times with each $\theta$ value to solve each instance. The results are summarized in Fig. 7 using the popular box and whisker plots, where the X-axis indicates the values of parameter $\theta$ and the Y-axis indicates the gap between the obtained objective value $R$ and the current best-known result $R^*$. Fig. 7 shows that all tested settings of $\theta$ lead to very similar results for the 20 runs and tested instances, which means that the algorithm is not sensitive to the setting of $\theta$.

The second experiment aims to investigate the sensitivity of parameter $MaxIter$.




(a) $N = 162$ (b) $N = 185$

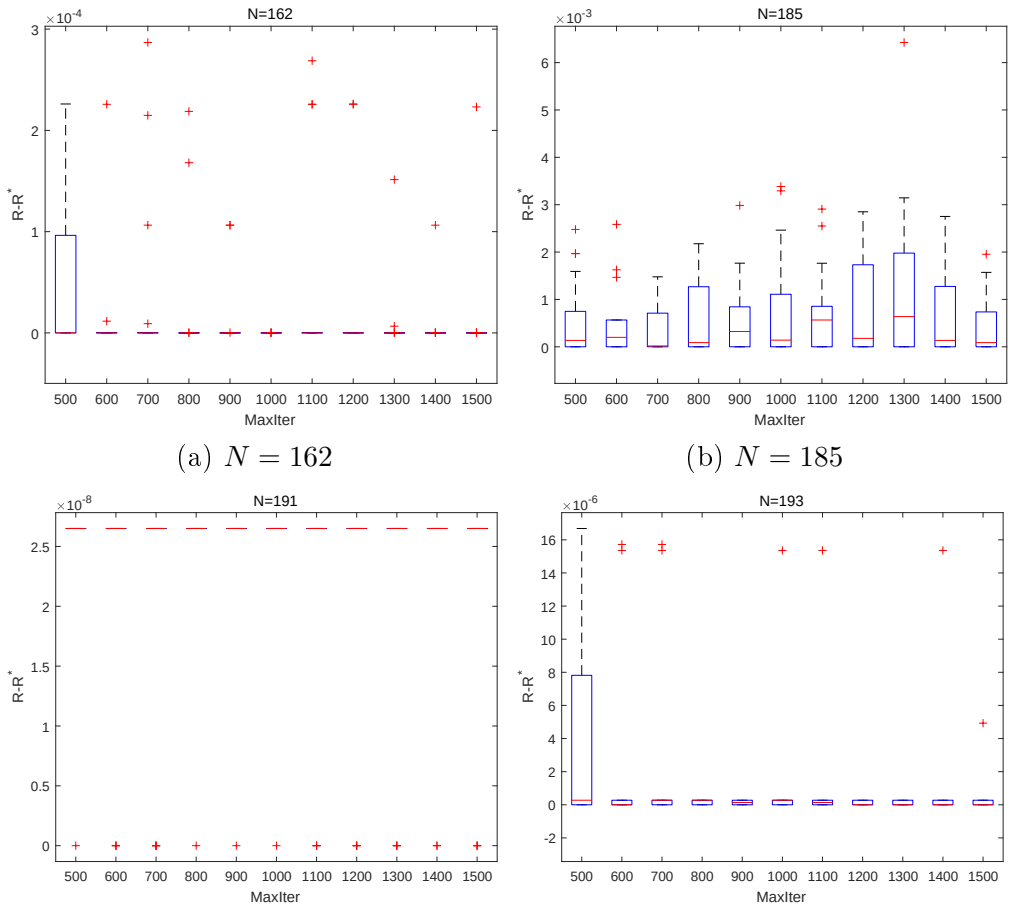
(c) $N = 191$ (d) $N = 193$

Fig. 7. Influence of parameter $\theta$ on the performance of the algorithm for 4 representative instances.

Like in the previous experiment, we varied the value of $MaxIter$ in the range of $[500, 1500]$ by an interval of 100, and then performed the IDTS algorithm 20 times with each $MaxIter$ value to solve each instance. The results are summarized in Fig. 8, where the X-axis indicates the values of $MaxIter$ and the Y-axis indicates the gap between the objective value $R$ and the current best-known result $R^*$. One observes from Fig. 8 that most $MaxIter$ values lead to very similar results for the tested instances, which means that the IDTS algorithm is not sensitive to the setting of parameter $MaxIter$.

### *5.1.2 Sensitivity Analysis of Parameter $\mu$*

The dynamic thresholding search procedure described in Algorithm 3 is one main component of the proposed IDTS algorithm. It employs a key parameter $\mu$ to dynamically adjust the threshold to determine whether a new solution should be accepted as the current solution or not. To check the influence of this parameter, we tested the IDTS algorithm on 15 selected difficult instances




(a) $N = 162$ (b) $N = 185$

(c) $N = 191$ (d) $N = 193$

Fig. 8. Influence of parameter $MaxIter$ on the performance of the algorithm for 4 representative instances.

with $101 \leq N \leq 200$, by varying $\mu$ in the range $\{0.75, 0.8, 0.85, 0.9, 0.95, 0.98\}$. Experimental results are summarized in Table 8, where the first column gives the number of circles in the instance ($N$) and columns 2–7 show respectively the average results ($R_{avg}$) of the algorithm over 20 runs for each tested $\mu$ value. The rows Avg. and #Best indicate respectively the average value for each column and the numbers of instances for which the corresponding $\mu$ value leads to the best result in terms of $R_{avg}$ among all the tested values.

Table 8 shows that the algorithm is statistically sensitive to the setting of this parameter. Specifically, the algorithm with $\mu = 0.75$ obtained the best performance in terms of $R_{avg}$ for 8 out of the tested 15 instances, a much higher number than with the other 5 $\mu$ values. On the other hand, the effectiveness of the parameter setting depends also on the instance to be solved. For example, for $N = 146$ and 156, the algorithm with $\mu = 0.98$ leads to the best result, but for $N = 140$ and 193, $\mu = 0.8$ is the most appropriate. As a result, the default value of $\mu$ is set to 0.75 in this work, given that this setting generated the global best results both in terms of Avg. and #Best.

Table 8
Influence of the parameter $\mu$ on the average results ($R_{avg}$) for 15 selected difficult instances in the range of $101 \leq N \leq 200$, where the best results obtained among the tested parameter values are indicated in bold.

| | $R_{avg}$ | | | | | |
|---|---|---|---|---|---|---|
| $N/\mu$ | 0.75 | 0.8 | 0.85 | 0.90 | 0.95 | 0.98 |
| 140 | 13.059841914 | **13.059817453** | 13.059823270 | 13.059869899 | 13.059888973 | 13.059835537 |
| 145 | **13.277790851** | 13.279272357 | 13.279111852 | 13.331265740 | 13.279055403 | 13.278586300 |
| 146 | 13.331264325 | 13.331264474 | 13.331270819 | 13.279138230 | 13.331277731 | **13.331261815** |
| 155 | **13.673710758** | 13.673711182 | 13.673711394 | 13.673713813 | 13.673713389 | 13.673713813 |
| 156 | 13.716830585 | 13.716872226 | 13.716815459 | 13.716939820 | 13.716864271 | **13.716725394** |
| 160 | **13.919810338** | 13.919881950 | 13.919825081 | 13.919900855 | 13.919864987 | 13.919819064 |
| 162 | **14.010217255** | 14.010262026 | 14.010301040 | 14.010339043 | 14.010382355 | 14.010458278 |
| 163 | **14.065216603** | 14.065224398 | 14.065240188 | 14.065243626 | 14.065249862 | 14.065378644 |
| 182 | 14.823505353 | 14.823788704 | **14.822847381** | 14.823913472 | 14.822465558 | 14.822936227 |
| 185 | **14.939607732** | 14.941075551 | 14.939665117 | 14.940375880 | 14.940583282 | 14.940206054 |
| 187 | 14.990022721 | 14.990192813 | 14.989746864 | 14.989629606 | **14.989615934** | 14.989776456 |
| 191 | **15.142357505** | 15.142360122 | 15.142357509 | 15.142357573 | 15.142357526 | 15.142357509 |
| 193 | 15.199379498 | **15.199379252** | 15.199382766 | 15.199380312 | 15.199382248 | 15.199383836 |
| 195 | 15.287684736 | 15.287625886 | 15.287717064 | 15.287688537 | **15.287611777** | 15.287697020 |
| 198 | **15.389903201** | **15.389903201** | **15.389903201** | **15.389903201** | 15.389903288 | 15.389903245 |
| Avg. | **14.321809558** | 14.322042106 | 14.321847933 | 14.321977307 | 14.321881105 | 14.321869279 |
| #Best | **8** | 3 | 2 | 1 | 2 | 2 |

In addition, it is worth mentioning that we tested several dynamic strategies to adjust the value of $\mu$. However, no significant improvement was observed with these strategies, which indicates a good robustness of the adopted strategy.

### *5.2 Effectiveness of the Two-phase Local Optimization*

The two-phase local optimization procedure described in Section 3.2 is one of the basic components of the proposed algorithm. It runs the LBFGS algorithm in two phases instead of one single phase. To verify whether such a two-phase approach is more efficient than the popular one-phase approach, we carried out an experiment based on 96 PECC-FR instances with $N \in \{50, 60, \ldots, 990, 1000\}$, where for each $N$ the container radius $R$ was set to the best result published at the well-known Packomania website [47]. In this experiment, the two-phase local optimization method and the standard LBFGS method were respectively performed 100 times to solve each instance, and at each time the initial solution was generated by uniformly and randomly distributing the centers $(x_i, y_i)$ $(i = 1, 2, \ldots, N)$ of $N$ circles in the given container. Moreover, to make a fair comparison, two local optimization methods employed the same stopping condition which is $||g||_\infty < 10^{-13}$. The experimental results are summarized in Fig. 9, where the X-axis denotes the number of circles ($N$) and the Y-axis denotes the average run times needed.

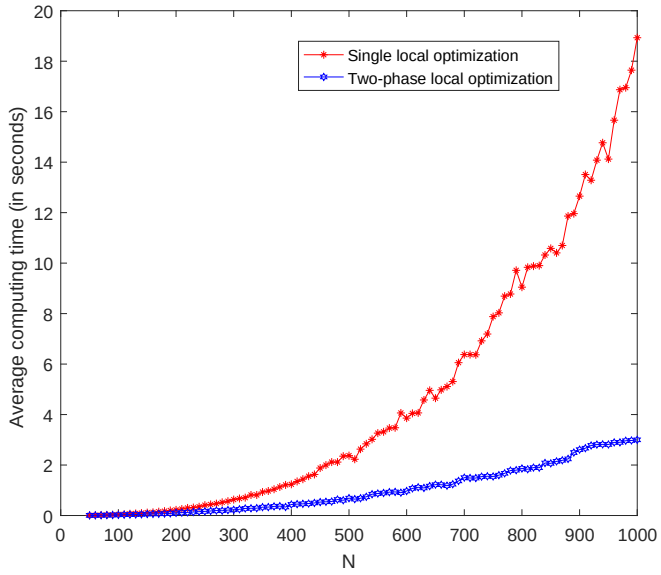



Fig. 9. Comparison between the standard one-phase local optimization method and the two-phase local optimization method used in the proposed algorithm.

One observes from Fig. 9 that the two-phase method is much more efficient compared to the popular one-phase method. For the one-phase method (i.e., the standard LBFGS method), the run time increases almost quadratically with the number of circles ($N$). However, for the two-phase method, the time increases almost linearly as $N$ increases. As a result, the two-phase local optimization method is much more faster than the one-phase method on the large instances. For example, for the instance with $N = 1000$, these two methods consumed respectively 3 and 19 seconds for one run. This observation implies that at the later stage of local optimizations the search process can be sped up significantly if the computational complexity for evaluating the objective function $E_R(X)$ is reduced by considering only the overlaps between neighboring circles especially for the large instances. Indeed, the two-phase local optimization procedure greatly contributed to the excellent performance reported in the last section especially on large instances.

## 6 Conclusions and Future Work

We presented an effective search algorithm for the packing equal circles in a circle problem, which is computationally challenging in terms of global optimization and relevant in terms of important applications. The proposed iterated dynamic thresholding search algorithm originally integrates three main complementary components (i.e., two-phase local optimization, dynamic thresholding search and container adjustment). The excellent performance of the algorithm was demonstrated on well-known benchmark instances with up to

320 circles. Indeed, the algorithm updated 136 best-known results (improved upper bounds) and matched the best-known results for other 175 instances. Given that the studied problem has a number of practical applications, the code of our algorithm, which we make publicly available, can be used to solve some of these applications where the run time is not a strong constraint.

On the other hand, the proposed algorithm can be further improved from the following directions. First, the popular penalty function method used in the container adjustment procedure leads to results of low precision for some instances. To deal with this limitation, other approaches such as the augmented Langerian multiplier method can be applied for container adjustment to improve the precision of the results. Second, to further improve the search capacity, the dynamic thresholding search proposed in this study can serve as the key intensification component of a population-based hybrid evolutionary method in combination with a suitable crossover operator. Third, this work demonstrates that even though the best-known results are already of very high quality, it is still possible to make improvements. However, this is at the price of high computational efforts. Therefore, it would be of great interest to improve the computation efficiency of the method. For instance, lattice packing can be used for solution initialization (e.g., [49]) and block-coordinate descent methods can serve as the local optimization method of the IDTS algorithm. Such fast methods would be useful to deal with very-large and super-large instances (e.g., $N > 500$), and contribute to better solve practical applications where the run time is a critical constraint.

Finally, the underlying ideas of the two-phase optimization strategy and the dynamic thresholding search method are of general nature. As such, they can be applied to other related problems such as equal circle packing on a sphere [9], equal sphere packing in a regular container [4], and covering a convex polygon region by equal circles [10].

## Acknowledgments

We are grateful to the reviewers for their valuable comments and suggestions which helped us to improve the paper. This work was partially supported by the National Natural Science Foundation of China (Grant No. 61703213), six talent peaks project in Jiangsu Province (Grant No. RJFW-011), and Shenzhen Science and Technology Innovation Commission under Grant Nos. JCYJ20180508162601910 and 2019-INT003.

## References


[1] Addis B., Locatelli M., Schoen F., 2008, Efficiently packing unequal disks in a circle. *Operations Research Letters*, 36, 37–42.

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

[3] Akiyama J., Mochizuki R., Mutoh N., Nakamura G., 2002, Maximin distance for n points in a unit square or a unit circle. *In: Akiyama, J., Kano, M. (Eds.)*, Japanese Conference on Discrete and Computational Geometry, JCDCG 2002, Tokyo, Japan, pp 9–13.

[4] Birgin E.G., Sobral F.N.C., 2008, Minimizing the object dimensions in circle and sphere packing problems. *Computers & Operations Research*, 35, 2357–2375.

[5] Birgin E.G., Gentil J.M., 2010, New and improved results for packing identical unitary radius circles within triangles, rectangles and strips. *Computers & Operations Research*, 37, 1318–1327.

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

[7] Castillo I., Kampas F.K., Pintér J.D., 2008, Solving circle packing problems by global optimization: Numerical results and industrial applications. *European Journal of Operational Research*, 191, 786–802.

[8] Chen M., Tang X.Y., Song T., Zeng Z.Z., Peng X.C, Liu S.Y., 2018, Greedy heuristic algorithm for packing equal circles into a circular container. *Computers & Industrial Engineering*, 119, 114–120.

[9] Clare B.W., Kepert D.L., 1991, The optimal packing of circles on a sphere. *Journal of Mathematical Chemistry*, 6, 325–349.

[10] Das G.K., Das S., Nandy S.C., Sinha B.P., 2006, Efficient algorithm for placing a given number of base stations to cover a convex region. *Journal of Parallel and Distributed Computing*, 66, 1353–1359.

[11] Demaine E.D., Fekete S.P., Lang R.J., 2010, Circle packing for origami design is hard. *CoRR*, arXiv:1008.1224.

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

[13] Dueck, G., Scheuer, T., 1990, Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90, 161–175.

[14] Fiacco A.V., GP McCormick G.P., 1964, Computational algorithm for the sequential unconstrained minimization technique for nonlinear programming. *Management Science*, 10, 601–617.

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

[16] Fu Z.H., Huang W.Q., Lü Z.P., 2013, Iterated tabu search for the circular open dimension problem. *European Journal of Operational Research*, 225(2), 236–243.

[17] Galiev S.I., Lisafina M.S., 2013, Linear models for the approximate solution of the problem of packing equal circles into a given domain. *European Journal of Operational Research*, 230, 505–514.

[18] Graham R.L., 1968. Sets of points with given minimum separation (solution to problem El921).*American Mathematical Monthly*, 75, 192–193.

[19] Graham R.L., Lubachevsky B.D., Nurmela K.J., Östergard P.R.J., 1998, Dense packings of congruent circles in a circle. *Discrete Mathematics*, 181, 139–154.

[20] Grebennik I.V., Kovalenko A.A., Romanova T.E., Urniaieva I.A., Shekhovtsov S.B., 2018, Combinatorial configurations in balance layout optimization problems, *Cybernetics and Systems Analysis*, 54, 221–231.

[21] Grosso A., Jamali A.R.M.J.U., Locatelli M., Schoen F., 2010, Solving the problem of packing equal and unequal circles in a circular container. *Journal of Global Optimization*, 47, 63–81.

[22] Hager W.W., Zhang H.C, 2005, A new conjugate gradient method with guaranteed descent and an efficient line search, *SIAM Journal on Optimization*, 16, 170–192.

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

[24] Hifi M., M' Hallah R., 2007, A dynamic adaptive local search algorithm for the circular packing problem. *European Journal of Operational Research*, 183, 1280–1294.

[25] Hifi M., Yousef L., 2019, A local search-based method for sphere packing problems. *European Journal of Operational Research*, 274, 482–500.

[26] Huang W.Q., Ye T., 2010, Greedy vacancy search algorithm for packing equal circles in a square. *Operations Research Letters*, 38, 378–382.

[27] Huang W.Q., Ye T., 2011, Global optimization method for finding dense packing of equal circle in a circle. *European Journal of Operational Research*, 210, 474–481.

[28] Kravitz S., 1967, Packing cylinders into cylindrical containers. *Mathematics Magazine*, 40, 65–71.

[29] Leary R., 2000, Global optimization on funneling landscapes. *Journal of Global Optimization*, 18, 367–383

[30] Liu D.C., Nocedal J., 1989, On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45, 503–528.

[31] Liu J.F., Xue S.J., Liu Z.X., Xu D.H., 2009, An improved energy landscape paving algorithm for the problem of packing circles into a larger containing circle. *Computers & Industrial Engineering*, 57, 1144–1149.

[32] Litvinchev I., Ozuna E. L., 2014, Approximate packing circles in a rectangular container: valid inequalities and nesting. *Journal of Applied Research and Technology*, 12,716–723.

[33] Litvinchev I., Infante L., Espinosa E.L.Q., 2016, Using valid inequalities and different grids in LP-based heuristic for packing circular objects, *Lecture Notes in Computer Science*, 9622, 681-690

[34] Locatelli M., Raber U., 2002, Packing equal circles in a square: a deterministic global optimization approach. *Discrete Applied Mathematics*, 122, 139–166.

[35] Locatelli M., Schoen F., 2002, Fast global optimization of difficult Lennard-Jones clusters. *Computational Optimization and Applications*, 21, 55–70.

[36] López C.O., Beasley J.E., 2011, A heuristic for the circle packing problem with a variety of containers. *European Journal of Operational Research*, 214, 512–525.

[37] López C.O., Beasley J.E., 2013, Packing unequal circles using formulation space search. *Computers & Operations Research*, 40, 1276–1288.

[38] López C.O., Beasley J.E., 2016, A formulation space search heuristic for packing unequal circles in a fixed size circular container. *European Journal of Operational Research*, 251, 64–73.

[39] Lü Z.P., Huang W.Q., 2008, PERM for solving circle packing problem. *Computers & Operations Research*, 35(5), 1742–1755.

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

[41] Martínez L., Andrade R., Birgin E.G., Martínez J.M., 2009, Packmol: A package for building initial configurations for molecular dynamics simulations. *Journal of Computational Chemistry*, 30(13):2157–2164.

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

[43] Mladenović N., Plastria K., Urošević D., 2005, Reformulation descent applied to circle packing problems. *Computers & Operations Research*, 32, 2419–2434.

[44] Müller A., Schneider J.J., Schömer E., 2009, Packing a multidisperse system of hard disks in a circular environment. *Physical Review E*, 79, 021102.

[45] Pirl U., 1969, Der Mindestabstand von n in der Einheitskreisscheibe gelegenen Punkten. *Mathematische Nachrichten*, 40(1–3), 111-124.

[46] Specht E., 2013, High density packings of equal circles in rectangles with variable aspect ratio. *Computers & Operations Research*, 40, 58–69.

[47] Specht E., November 2020 (Accessed), Packomania website: `http://www.packomania.com`.

[48] Stetsyuk P.I., Romanova T.E., Scheithauer G., 2016, On the global minimum in a balanced circular packing problem. *Optimization Letters*, 10, 1347–1360.

[49] Stoyan Y., Yaskov G., Romanova T., Litvinchev I., Yakovlev S., Cantú J.M.V., 2020, Optimized packing multidimensional hyperspheres: a unified approach, *Mathematical Biosciences and Engineering*, 76(6): 6601–6630.

[50] Stoyan Y., Zlotnik M., Chugay A., 2012, Solving an optimization packing problem of circles and non-convex polygons with rotations into a multiply connected region. *Journal of the Operational Research Society*, 63: 379–391.

[51] Stoyan Y., Yaskov G., 2014, Packing unequal circles into a strip of minimal length with a jump algorithm. *Optimization Letters*, 8: 949–970.

[52] Stoyan Y., Scheithauer G., Yaskov G.N., 2016, Packing unequal spheres into various containers. *Cybernetics and Systems Analysis*, 52: 419–426.

[53] Toledo F.M.B., Carravilla M.A., Ribeiro C., Oliveira J.F., Gomes A.M., 2013, The dotted-board model: a new MIP model for nesting irregular shapes. *International Journal of Production Economics*, 45(2), 478–487.

[54] Wang H.Q., Huang W.Q., Zhang Q., Xu D.M., 2002, An improved algorithm for the packing of unequal circles within a larger containing circle. *European Journal of Operational Research*, 141, 440–453.

[55] Zeng Z.Z., Yu X.G., He K., Huang W.Q., Fu Z.H., 2016, Iterated tabu search and variable neighborhood descent for packing unequal circles into a circular container. *European Journal of Operational Research*, 250, 615–627.

[56] Zhang D.F., Deng A., 2005, An efficient hybrid algorithm for the problem of packing the circles into a larger containing circle. *Computers & Operations Research*, 32(8), 1941–1951.