Iterated dynamic neighborhood search for packing equal circles on a sphere

Xiangjing Lai[a], Dong Yue[a], Jin-Kao Hao[b,*], Fred Glover[c], Zhipeng Lü[d]

[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]*Entanglement, Inc., Boulder, Colorado, USA*
[d]*SMART, School of Computer Science and Technology, Huazhong University of Science and Technology, 430074 Wuhan, China*

Computers and Operations Research, December 2022
https://doi.org/10.1016/j.cor.2022.106121

Abstract

In this work, we investigate the equal circle packing problem on a sphere (ECPOS), which consists in packing N equal non-overlapping circles on a unit sphere such that the radius of circles is maximized. The problem is of great interest in biology, engineering and operations research and thus has a rich research history both from theoretical and computational aspects. We propose from the point of view of computational research an effective iterated dynamic neighborhood search (IDNS) algorithm for the ECPOS problem. The algorithm includes a multiple-stage local optimization method, a general dynamic neighborhood search method and an adjustment method of the minimum distance between the points on the unit sphere. Extensive experiments are conducted with the proposed algorithm on 205 instances commonly used in the literature. Computational results show that the algorithm is highly effective by improving the best-known results for 42 instances and matching the best-known results for other 116 instances, while missing the best-known results for only 5 instances. For the remaining 42 instances, the best-known results are reported for the first time by the IDNS algorithm.

*Corresponding author.
Email addresses: laixiangjing@gmail.com (Xiangjing Lai), medongy@vip.163.com (Dong Yue), jin-kao.hao@univ-angers.fr (Jin-Kao Hao), fred@entanglement.ai (Fred Glover), zhipeng.lv@hust.edu.cn (Zhipeng Lü)

Keywords: Circle packing; Tammes problem; Global optimization; Metaheuristics; Nonlinear programming.

1. Introduction

Given N equal non-overlapping circles and a unit sphere with the surface S^2, the circle packing problem studied in this work consists in packing these N circles on the surface such that the radius (or the angular diameter) of the circles is maximized, where each circle corresponds to a spherical cap defined as the inside of the circumference of a circle on S^2 [43]. This problem is also known as the Tammes problem in the literature [42] and is equivalent to the problem of maximizing the minimum distance between N points on S^2, where each point corresponds to the center of a spherical cap on S^2. Throughout this paper, these two equivalent descriptions of the problem will be indifferently used according to the context.

Unlike other circle packing problems, such as packing circles into a regular container (e.g., circle or square) [1, 5, 16, 24, 25, 28, 35, 36, 49, 50], packing equal circles on the sphere does not involve the container boundary and thus is very interesting. It can act as a remarkable test system to evaluate various global optimization techniques due to its NP-hard feature [9] and the fact that the number of locally optimal solutions increases exponentially as the number of circles N increases. On the other hand, this problem is a well-known global optimization model with a large number of applications in biology, engineering, operations research and information theory [3, 15, 17, 21, 26, 27, 44]. For instance, the globally optimal solution of the problem corresponds to the ground-state structure of atomic clusters [40], while the optimal solutions of some instances correspond to the structures of spherical pollen-grains, and the problem with $N = 13$ is the famous 13-sphere problem in mathematics [29, 30]. Recently, this problem was applied to find diversified neurons in the design of neural networks [48] and to distribute a group of N agents on a sphere while maximizing the minimum inter-agent distance [2]. Interested readers are referred to [40] for more application examples.

Due to its practical importance and theoretical significance, the ECPOS problem has received a lot of attention since 1930 and a large number of related researches have been reported in the literature examining both theoretical and computational aspects. The earliest work about this problem stems from the botanist Tammes who studied the distribution of hollows on

the surface of spherical pollen-grains [37]. Since then, a number of approaches were proposed to find or prove the optimality of solutions to the problem.

Up to now, the proven optimal solutions were found only for very small instances with $N = 1 - 14$ and 24 [8, 18, 29, 30, 31, 32]. For example, Musin and Tarasov proved the optimality of solutions by mathematical methods for $N = 13$ and 14 [29, 30]. For larger instances, the optimality proofs of solutions are difficult for mathematical methods and exact algorithms. Thus, for large-scale instances, many researchers turned to methods of finding high-quality suboptimal solutions, instead of proving the solution optimality. The employed approaches mainly include mathematical methods, construction methods based on prior knowledge of the problem, and numerical global optimization methods.

In 1983, based on the theory of bar structures, Taenia and Gáspár improved the best-known configurations by a mathematical method for instances with $N = 18$, 27, 34, 35 and 40 [41], without giving an optimality proof.

After that, many construction methods were proposed to predict the optimal solutions by utilizing prior knowledge of the problem. In 1987, inspired by the structure of virus coats, Taenia and Gáspár [42] investigated four packing sequences of circles on the sphere by taking into account of rotational symmetry of the regular tetrahedron, octahedron and icosahedron, and obtained some multi-symmetric packing configurations by a construction method for $N = 78$, 96, 108, 144, 150, 192, 198, 270, 360, 372, 480, 492. Subsequently, Gáspár further extended one of these sequences to several large instances with $N = 150$, 216, 300, 432, 750 and 1080 [12]. These highly symmetrical configurations were shown to be very promising candidates for optimal solutions for the special sizes. Between 1994 and 2000, Hardin et al. solved the ECPOS problem by some construction methods and presented for the first time the putatively optimal solutions for all instances in the range of $N \leq 130$. All the best solutions obtained are available online in [34], together with some other best-known results collected from other researchers. Moreover, the authors also generated high-quality solutions with icosahedral symmetry for a number of large-scale instances between $N = 60$ and $N = 33002$ and made them available online in [14]. At the present time, these results can be regarded as the best-known results for the ECPOS problem. In 2000, Teshima and Ogawa proposed a novel construction method named the minimum-zenith method and tested their method on all instances in the range of $N \leq 150$. This method starts from an initially constructed

partial solution, and then sequentially packs the remaining circles on the surface of the unit sphere such that the zenith angle is as small as possible.

Besides these mathematical and construction methods, some numerical global optimization methods were proposed in the literature, without utilizing prior knowledge of the problem. In 1977, Mackay et al. solved the ECPOS problem by an iterative numerical optimization algorithm and then listed for the first time the putatively optimal solutions for all the instances with up to $N = 27$ [26]. In 1986, Clare and Kepert improved the best-known results for a number of instances in the range of $N = 20 - 40$ by minimizing orderly a series of energy functions $E_m(X)$ $(m \geq 1)$ defined on the unit sphere:

$$E_m(X) = \sum_{1 \leq i < j \leq N} (\frac{1}{d_{ij}})^m \tag{1}$$

where X is a configuration of N points on the unit sphere, m is a positive integer and d_{ij} denotes the distance between points i and j [7]. It is worth noting that the local minimum solutions of $E_m(X)$ will converge to a local minimum solution of the ECPOS problem as the value of m increases to a very large number. The optimization process was conducted in a multi-start fashion, each rerun starting from an initial random solution. In 1991, the same authors applied their approach to instances with $N = 19 - 80$ and improved again the best-known results for many of them. Their computational results showed that the best-known configurations are generally of low symmetry, differing from the constructed solutions. In the same year, using a variant of this approach, Kottwitz further improved the best-known results for a number of instances in the range of $N = 15 - 90$ [17]. It should be noted that the algorithm proposed in the present work also falls into this category of optimization methods, where a potential function is minimized via a series of random perturbations (or restarts) and gradient-based local optimizations [7, 17]. This type of optimization methods were also widely used to solve the related circle or sphere packing problems in the literature [1, 4, 5, 13, 19]

In addition, a number of studies were dedicated to some variants of the ECPOS problem. For example, Appelbaum and Weiss investigated the problem of packing equal circles on a hemisphere [3]. Tarnai et al. and Fowler et al. studied respectively the problems of packing regular triplets or tetrahedral quartets of circles on a sphere [11, 39, 40].

Our motivation is twofold in this work. First, we undertake to devise a highly effective heuristic algorithm for the ECPOS problem due to its

important applications and computational challenges. Second, we propose a general-purpose approach for global optimization of non-convex continuous functions, thus providing more tools for global optimization.

The contributions of this work can be summarized as follows. First, we propose the iterated dynamic neighborhood search (IDNS) global optimization algorithm for solving the ECPOS problem. Computational results show that the proposed algorithm performs very well and improves the best-known results for a number of instances widely tested in the literature. Second, the two main components (i.e., the dynamic neighborhood search and the multiple-stage local optimization) of the IDNS algorithm are of general nature. The dynamic neighborhood search can be applied to perform global optimization of any non-convex differentiable function, while the multiple-stage local optimization is applicable to geometry optimization problems in which the system is composed of a number of particles interacting via a short-ranged potential.

The rest of paper is organized as follows. In Section 2, the mathematical formulations of the ECPOS problem and our proposed algorithm are described. In Section 3, the performance of our proposed algorithm is assessed based on a large number of instances and the structural features of putatively optimal solutions are investigated. Section 4 analyzes the key algorithmic components, including the parameters and the local optimization method. Finally, the last section provides several research perspectives.

2. Global optimization method

We first introduce the non-linear optimization formulations of the ECPOS problem, and then describe the iterated dynamic neighborhood search (IDNS) global optimization algorithm for solving it. The proposed algorithm can be viewed as an iterated neighborhood search method whose main idea is to control dynamically the size of neighborhood to reach a good tradeoff between the intensification and diversification of the search process. In this sense, the proposed algorithm shares the same idea with some popular global optimization algorithms, such as the basin-hopping algorithm and its variants [20, 45]. However, to deal with our particular problem, the proposed IDNS algorithm integrates some innovations in terms of global optimization as explained below.

2.1. Formulations of the ECPOS problem

Given a positive number N, the ECPOS problem aims to distribute N points (or circles) on the unit sphere such that the minimum distance D between points is maximized. In three-dimensional Cartesian coordinate system, the ECPOS problem can be described as follows:

$$Maximize \quad D \tag{2}$$

$$s.t. \quad \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \geq D, 1 \leq i, j \leq N; \tag{3}$$

$$\sqrt{x_i^2 + y_i^2 + z_i^2} = 1, 1 \leq i \leq N; \tag{4}$$

where (x_i, y_i, z_i) and (x_j, y_j, z_j) respectively represent the Cartesian coordinates of points r_i and r_j. Constraints (3) ensure that the distance between any two points is larger than D, and constraints (4) ensure that all N points are confined on the surface of the unit sphere S^2.

The problem defined by Eqs. (2)-(4) is a constrained optimization problem and intractable for the popular local optimization methods.

To make local optimization methods applicable to this constrained optimization problem, we first convert the ECPOS problem into a series of constraint satisfaction subproblems by successively fixing the value of minimum allowed distance D to a constant number, where the goal of each subproblem is to find a solution for which the minimum distance between points is larger than or equal to the given D value. Then, we solve each subproblem by a stochastic optimization approach called the dynamic neighborhood search method. Given a set of points $X = \{r_1, r_2, \ldots, r_N\}$ on the unit sphere S^2 and a fixed minimum distance D (> 0) between points, the constraint satisfaction subproblem can be converted into a minimization problem as follows by means of the penalty function method:

$$Miminize \quad E_D^c(X) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} max^2\{0, D - d(i,j)\} \tag{5}$$

$$s.t. \quad r_i, r_j \in S^2, i, j = 1, 2, \ldots, N \tag{6}$$

where $d(i,j)(= \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2})$ represents the Euclidean distance between the points r_i and r_j, and $max\{0, D - d(i,j)\}$ represents the overlap depth between two spherical caps with the centers at points r_i and

r_j. $E_D^c(X) = 0$ means that X is feasible, and infeasible otherwise. Thus, the goal of subproblem is to find a feasible solution X with $E_D^c(X) = 0$ for the given D value.

The subproblem defined by Eqs. (5) and (6) is still a constrained optimization problem which is not easy to handle by popular local optimization methods like the LBFGS method. To perform the local optimization, we convert the problem further to an unconstrained optimization problem by using the spherical coordinate transformation of points on S^2:

$$\begin{cases} x = sin\varphi cos\theta; & (7) \\ y = sin\varphi sin\theta; & (8) \\ z = cos\varphi; & (9) \end{cases}$$

where (x, y, z) and (φ, θ) represent respectively the Cartesian coordinates and the spherical coordinates of a point r on the unit sphere.

Thus, under the spherical coordinate system of points, a candidate solution can be indicated as (X, D) in which $X = (\theta_1, \varphi_1, \ldots, \theta_N, \varphi_N)$ and D denotes the minimum allowed distance between points, and the subproblem defined by Eqs. (5) and (6) can be equivalently expressed as an unconstrained optimization problem as follows:

$$E_D(\varphi, \theta) = \sum_{1 \le i \le j \le N} max^2\{0, D - \sqrt{2 - 2sin\varphi_i sin\varphi_j cos(\theta_i - \theta_j) - 2cos\varphi_i cos\varphi_j}\} \quad (10)$$

where $\varphi = (\varphi_1, \varphi_2, \ldots, \varphi_N) \in R^N$, $\theta = (\theta_1, \theta_2, \ldots, \theta_N) \in R^N$, since both the functions $sin(x)$ and $cos(x)$ are a periodic function with respect to the variable x in R $(= (-\infty, +\infty))$.

2.2. General procedure of iterated dynamic neighborhood search

The IDNS algorithm is a trajectory-based stochastic optimization approach whose pseudo-code is given in Algorithm 1, where X^* and D^* respectively denote the best solution found so far and the corresponding minimum distance between points, while X and D respectively denote the current solution and corresponding minimum allowed distance between points. The algorithm includes three components, a multiple-stage local optimization method, a dynamic neighborhood search (DNS) method and an adjustment procedure of the minimum distance between points.

Algorithm 1: General procedure of the Iterated dynamic Neighborhood Search (IDNS) algorithm

Input: Number of points to be distributed (N), time limit (t_{max})
Output: The best configuration found (X^*, D^*)

$\delta \leftarrow 0.7$
$D \leftarrow 4 \times \sqrt{\frac{\delta}{N}}$
$X \leftarrow RandomSolution(N)$ /* Generate an initial solution */
$X \leftarrow LocalOptimization(X, D)$
/* $E_D(X) < 10^{-25}$ means that X is a feasible solution */
while $E_D(X) < 10^{-25} \wedge time() \leq t_{max}$ **do**
    $\delta \leftarrow \delta + 0.001$
    $D \leftarrow 4 \times \sqrt{\frac{\delta}{N}}$ /* Increase the value of D */
    $X \leftarrow DNS(X, D)$ /* Minimize the function $E_D(X)$ defined in Eq.(10) by the DNS method */
end
$(X^*, D^*) \leftarrow MinDistanceAjustment(X, D)$ /* Adjust the minimum distance D between N points */
while $time() \leq t_{max}$ **do**
    $D \leftarrow D^*$
    $X \leftarrow RandomSolution(N)$ /* A random solution */
    $X \leftarrow DNS(X, D)$
    if $E_D(X) < 10^{-25}$ **then**
        $(X, D) \leftarrow MinDistanceAjustment(X, D)$
        if $D > D^*$ **then**
            $D^* \leftarrow D$
            $X^* \leftarrow X$ /* Save the best solution found */
        end
    end
end
return (X^*, D^*)

The algorithm is performed in a two-phase fashion. At the first phase, from an initial value of D empirically estimated (lines 1–2), an initial solution X is generated by randomly distributing N points on the surface S^2 of the unit sphere and is improved by the minimization of the function $E_D(X)$. Subsequently, the algorithm enters a 'while' loop and performs a number of iterations until an infeasible solution with $E_D(X) > 10^{-25}$ is reached (lines 6–10). At each iteration, the value of D is first progressively increased (lines 7–8) and then the DNS procedure is used to find a feasible solution under the current D value by minimizing the function $E_D(X)$ (line 9). After that, the result of the DNS procedure is used as the input of the next iteration. The algorithm enters the second phase once the DNS procedure fails to find a feasible solution for the current D value and the given input solution.

At the second phase of the search, the algorithm performs a number of iterations until the time limit (t_{max}) is reached (lines 12-23). At each iteration, starting from a solution generated randomly, the DNS procedure is performed to optimize the objective function $E_D(X)$ with D^* set to be the value of D (lines 13–15). Then, the minimum distance adjustment procedure is used to maximize the value of D once a feasible solution X with $E_D(X) < 10^{-25}$ is found, while maintaining the feasibility of solution (line 17). After that, the best solution X^* found so far and the corresponding D value (i.e., D^*) are updated if an improved solution is found (lines 18–21).

2.3. Dynamic neighborhood search method

For the trajectory-based global optimization, one important issue concerns the strategy used to accept the new solution as the current solution, and different strategies will result in various global optimization algorithms. For example, the basin-hopping algorithm employs the Metropolis acceptance rule, which depends on the change in the objective value and a parameter T called the temperature, to accept the new solution [45].

As an initial step, we design a new metaheuristic approach called dynamic neighborhood search (DNS) to search for the global optimum of an unconstrained continuous optimization problem with the first-order derivative. The pseudo-code of the DNS method is given in Algorithm 2, where X_{cur} and X^b respectively denote the current solution and the best solution found by the current DNS run. Starting from an input solution X_0, the DNS method performs a number of iterations to improve its quality until an optimal solution X (i.e., $E_D(X) = 0$ for the present study) is encountered or

Algorithm 2: Dynamic neighborhood search (DNS) for the minimization of the function $E_D(X)$

Function *DNS()*
Input: Input solution X_0, minimum distance allowed D between points
Output: The best solution found (X^b)
$X_{cur} \leftarrow LocalOptimization(X_0, E_D(\cdot))$
$X^b \leftarrow X_{cur}$ /* X_{cur} denotes the current solution */
$\beta \leftarrow 0$
while $(\beta \leq \beta_{max}) \wedge (E_D(X^b) > 10^{-25})$ **do**
  $\eta \leftarrow \eta_f(\beta)$ /* Determine the strength of perturbation */
  $M \leftarrow M_f(\beta)$ /* Determine the maximum size of current neighborhood */
  /* Construct a neighborhood with a maximum cardinality M for X_{cur} */
  $E_D(X_{nbest}) \leftarrow +\infty$ /* X_{nbest} denotes the best solution in the neighborhood */
  for $k \leftarrow 1$ ***to*** M **do**
    $X_{neighbor} \leftarrow Perturbation(X_{cur}, \eta)$
    $X_{neighbor} \leftarrow LocalOptimization(X_{neighbor}, E_D(\cdot))$
    if $E_D(X_{neighbor}) < E_D(X_{nbest})$ **then**
      $X_{nbest} \leftarrow X_{neighbor}$
    end
    if $E_D(X_{neighbor}) < E_D(X_{cur})$ **then**
      $X_{nbest} \leftarrow X_{neighbor}$
      break
    end
  end
  $X_{cur} \leftarrow X_{nbest}$ /* update the current solution */
  if $E_D(X_{cur}) < E_D(X^b)$ **then**
    $X^b \leftarrow X_{cur}$ /* Save the best solution found */
    $\beta \leftarrow 0$
  else
    $\beta \leftarrow \beta + 1$
  end
end
return X^b

the best solution X^b cannot be improved during β_{max} consecutive iterations (lines 5–28), where β_{max} is a parameter called the maximum search depth.

At each iteration, the DNS method first constructs a neighborhood $N(X_{cur})$ for the current solution X_{cur} (lines 10–20) and then selects a best solution from $N(X_{cur})$ to replace the current solution (line 21), where the size of the constructed neighborhood is dynamically controlled by a function (line 7). Figure 1 provides a schematic diagram to illustrate the search process of the DNS method. One can observe from the figure that the size of the neighborhood varies dynamically during the search and that the DNS method is allowed to accept deteriorated solutions to escape from local optima.

To construct a neighborhood of the current solution, the DNS method employs a perturbation operator and a subsequent local optimization procedure to generate each neighboring solution (lines 11–12). Specifically, a random perturbation operator with a strength η is first applied to the current solution X_{cur} and then a local optimization procedure is applied to the perturbed solution to improve its quality, and the resulting solution is used as a member of the current neighborhood. This process is repeated at most M times to generate the members of the current neighborhood $N(X_{cur})$, where M is the maximum size of the neighborhood. The neighborhood construction process stops once a better neighborhood solution than the current solution X_{cur} is encountered (lines 16–19).

To reach a suitable tradeoff between diversification and intensification of the search, the DNS algorithm determines dynamically the perturbation strength η in an interval $[\eta_{min}, \eta_{max}]$ by using a periodic function $\eta_f(\cdot)$ described by Fig. 2 (b), given that a smaller perturbation strength implies a more intensified search and a larger perturbation strength implies a more diversified search. On the other hand, the neighborhood size plays also a key role for the tradeoff between diversification and intensification, since a larger neighborhood allows a more intensified search, but requires a higher computational effort, while a smaller neighborhood allows a more diversified search. Thus, to reach a suitable tradeoff, the DNS method employs a periodic function $M_f(\cdot)$ described in Fig. 2 (a) to dynamically determine the value of M, making it vary periodically in an interval $[M_{min}, M_{max}]$.

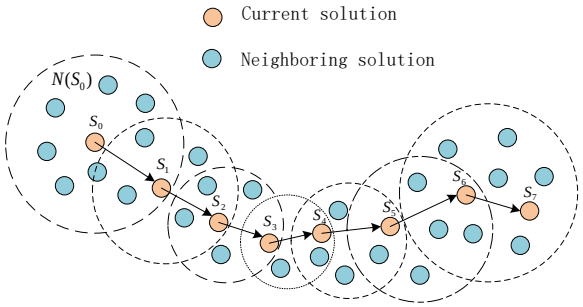
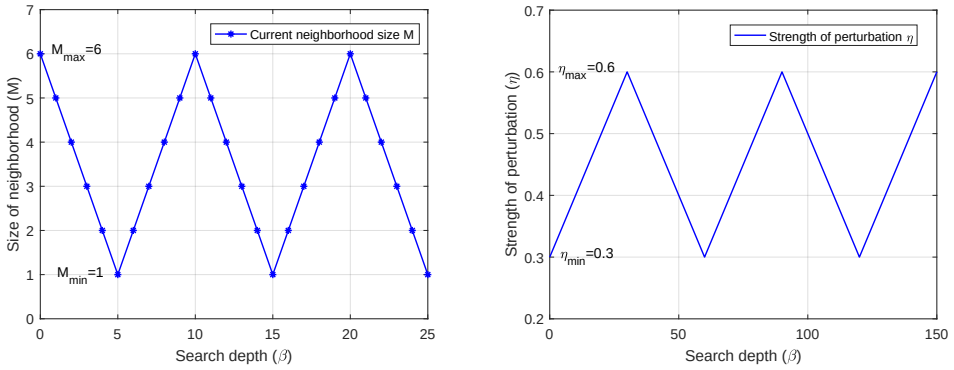


Figure 1: A schematic diagram illustrating the dynamic neighborhood search (DNS) approach, where the size of neighborhood varies dynamically according to a function $M_f(\cdot)$ with respect to the search depth β.



(a) The function for determining the size of neighborhood M_f

(b) The function for determining the strength of perturbation η_f

Figure 2: The variations of the strength of current perturbation (η) and the size of current neighborhood (M) as a function of the search depth (β).

Fig. 2 indicates that the neighborhood size M and the perturbation strength η will respectively reach their maximum value and minimum value at the beginning of the search or when an improved solution is found (i.e., $\beta = 0$) to reinforce search intensification, and then vary dynamically to reinforce search diversification. Eventually, the combined use of dynamic strategies for M and η will cause the DNS algorithm to reach a desirable diversification and intensification tradeoff.

2.4. Multiple-stage Local optimization method

Algorithm 3: Multiple-Stage local optimization

1 **Function** *LocalOptimization*()
Input: Input solution X, the minimum distance allowed (D) between the points, the precisions $\{\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3\} = \{0.1D, 0.01D, 10^{-4}D, 10^{-12}D\}$, the cutoff distances for generating the lists of neighbors of circles $\{\Delta_1, \Delta_2, \Delta_3\} = \{3D, 2D, 1.2D\}$
Output: local minimum solution X
2 $X \leftarrow LBFGS(X, \epsilon_0)$ `/* The first stage */`
3 $L_1 \leftarrow ListOfNeighbors(X, \Delta_1)$
4 $X \leftarrow LBFGS(X, L_1, \epsilon_1)$ `/* The second stage */`
5 $L_2 \leftarrow ListOfNeighbors(X, \Delta_2)$
6 $X \leftarrow LBFGS(X, L_2, \epsilon_2)$ `/* The third stage */`
7 $L_3 \leftarrow ListOfNeighbors(X, \Delta_3)$
8 $X \leftarrow LBFGS(X, L_3, \epsilon_3)$ `/* The fourth stage */`
9 **return** X

The local optimization procedure is the most time-consuming component of our IDNS algorithm and plays a crucial role for its performance, where most of the computational time is consumed by the evaluations of the objective function $E_D(X)$ and its gradient $g(X)$. In the local optimization, given that the overlaps occur only between the neighboring circles in the candidate solution X, we speed up the evaluations of the objective function $E_D(X)$ and its gradient $g(X)$ by considering the possible adjacency relations between circles in the current solution.

To do this, the local optimization method whose pseudo-code is given in Algorithm 3 is performed in multiple stages, where each stage is composed of a construction procedure that involves a list of neighboring circles and a local search method based on the neighboring list.

Specifically, the local optimization method employed by the algorithm is divided into four stages. At the first stage (line 2), the LBFGS method [22] with a low stopping precision of $\epsilon_0 = 0.1D$ is performed to improve the input solution, where D is a given and allowed distance between the centers of circles and the overlap is calculated for each pair of circles in the evaluation of $E_D(X)$ (i.e., each pair of circles is considered to be adjacent in this stage). For most circles, the adjacency relations indicating whether two circles are adjacent in the resulting solution will be determined in the first stage and do

not change in the following stages. The number of iterations needed at this stage is very small due to the low stopping precision, and the computational complexity of each iteration is high (i.e., $O(N^2)$), which is the same as the complexity of a single function evaluation or gradient evaluation.

At the second stage (lines 3–4), the adjacency list L_1 between circles is first generated by considering a cutoff distance Δ_1, and two circles r_i and r_j are identified as neighbors if the distance between their centers on the unit sphere is less than Δ_1. Then, the second LBFGS method is performed utilizing the result of the first stage as the starting point, where only those overlaps between circles identified by the adjacency list L_1 are considered in the evaluation of objective function $E_D(X)$ and its gradient $g(X)$. Thus, the time complexity of function and gradient evaluations is only $O(N)$, which is much cheaper than the first stage. Subsequently, similar to the second stage, the third and fourth stages are performed by constructing the adjacency lists L_2 and L_3 based on the result of the previous stage and performing the corresponding LBFGS procedure (lines 5–8), where the cutoff distances Δ_i $(i = 2, 3)$ decrease gradually to reduce the sizes of the constructed adjacency lists L_2 and L_3.

The idea of this multiple-stage local optimization method is very general and can be applied to a number of geometry optimization problems in which the interaction between two particles can be described by a short-ranged potential with respect to their distance, such as circle and sphere packing problems and the structural optimization of Lennard-Jones clusters [45].

2.5. Perturbation Operator

Algorithm 4: Perturbation operator

1 **Function** *Perturbation*()
  Input: Input solution $X = (\theta_1, \varphi_1, \ldots, \theta_N, \varphi_N)$, parameter η_1
  Output: The perturbed solution X
2 `/*` $rand(-\eta_1, \eta_1)$ `is a random number in` $(-\eta_1, \eta_1)$ `*/`
3 **for** $i \leftarrow 1$ ***to*** N **do**
4 | $\theta_i \leftarrow \theta_i + rand(-\eta_1, \eta_1)$
5 | $\varphi_i \leftarrow \varphi_i + rand(-\eta_1, \eta_1)$
6 **end**
7 **return** $X = (\theta_1, \varphi_1, \ldots, \theta_N, \varphi_N)$

To optimize globally the objective function $E_D(\cdot)$ defined in Eq.(10), the DNS method employs a perturbation operator to escape from the current

local minimum. The pseudo-code of the perturbation operator is given in Algorithm 4. Given a candidate solution $X = (\theta_1, \varphi_1, \ldots, \theta_N, \varphi_N)$, the perturbation operator shifts each coordinate of the solution X in the interval $[-\eta_1, \eta_1]$ to obtain a new solution, where the value of η_1 is determined as $\eta_1 = \eta \times \theta_0$ in which η is a number in $(0, 1)$ and is called the perturbation strength and θ_0 represents the minimum angle between any two points on the unit sphere and is determined as $\theta_0 = arccos(\frac{2-D^2}{2})$.

2.6. Minimum distance adjustment method

Algorithm 5: Adjustment method of minimum distance (D) between points

Function *MinDistanceAdjustment()*
Input: Input solution (X_0, D_0), maximum number of iterations K $(= 15)$
Output: The feasible local optimum configuration (X, D)
$X \leftarrow X_0$, $D \leftarrow D_0$, $\rho \leftarrow 10^2$
for $i \leftarrow 1$ ***to*** K **do**
    $(X, D) \leftarrow LocalOptimization(U_\rho, X, D)$ `/* Minimize` $U_\rho(X, D)$ `using the LBFGS method */`
    $\rho \leftarrow 5 \times \rho$
end
return (X, D)

Given a configuration (X, D) on the sphere, the minimum distance adjustment method aims to slightly modify the coordinates of N points (i.e., the centers of spherical caps), such that no overlap occurs between any two spherical caps in the resulting configuration, while the minimum distance D between points is maximized. The adjustment of minimum distance D is equivalent to obtaining a local solution to a constrained optimization problem.

As in [19], to locally optimize the constrained optimization problem, we employ the sequential unconstrained minimization technique (SUMT) [10]. First, we convert the constrained optimization problem into a series of unconstrained optimization problems which can be written as follows:

$$Minimize \quad U_\rho(X, D) = -D^2 + \rho \times E(X, D) \tag{11}$$

where ρ is a penalty factor and each given ρ value defines a unconstrained optimization problem, D is a variable representing the minimum allowed distance between points, and $E(X, D)$ is a penalty term with $2N+1$ variables

which measures the degree of constraint violation in (X, D), and $E(X, D) = 0$ if and only if (X, D) is a feasible solution.

$$E(X, D) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} max^2\{0, D - d(i, j)\} \tag{12}$$

where $max\{0, D-d(i, j)\}$ represents the overlap depth between two spherical caps and $d(i, j)$ is the Euclidean distance between two points (i.e., the centers of two spherical caps) and can be written as:

$$d(i, j) = \sqrt{2 - 2sin\varphi_i sin\varphi_j cos(\theta_i - \theta_j) - 2cos\varphi_i cos\varphi_j} \tag{13}$$

Then, the adjustment method (see Algorithm 5) solves in order a series of unconstrained optimization problems defined by Eq.(11) with increasing ρ values. Starting from a configuration (X_0, D_0) to be adjusted and an initial ρ value ($\rho_0 = 10^2$), the adjustment procedure performs K iterations. At each iteration, the procedure locally optimize the function defined in Eq.(11) by a LBFGS method [22] and the resulting solution is used as the input solution of the next iteration, followed by increasing the value of ρ by setting $\rho \leftarrow 5 \times \rho$. As ρ increases to a very large value, the adjustment method converges to a feasible solution with $E(X, D) = 0$ in which the value of D is maximized.

3. Computational results and comparisons

To evaluate the performance of the IDNS algorithm and predict the globally optimal configuration of the ECPOS problem, we conducted extensive experiments whose experimental conditions and computational results are given in the following subsections.

3.1. Parameter settings and experimental protocol

Table 1: Settings of parameters

Parameters	Section	Description	Values
M_{min}	2.3	minimum size of neighborhood	1
M_{max}	2.3	maximum size of neighborhood	6
η_{min}	2.3	minimum strength of perturbation	0.3
η_{max}	2.3	maximum strength of perturbation	0.6
η_s	2.3	incremental change of strength of perturbation	0.01
β_{max}	2.3	maximum depth of search	500

The algorithm adopts several parameters whose default settings are given in Table 1, which were determined by a preliminary experiment described in Section 4. In this study, all computational experiments were carried out with the parameter default settings. It is worth noting that these settings are not optimal and fine-tuning some parameters could help the algorithm to find improved results [1]

The IDNS algorithm was implemented in the C++ language and all computational experiments were carried out on a computer with an Intel(R) Xeon (R) Platinum 9242 CPU (2.3 GHz)[2], running a Linux operating system. Given its stochastic feature, the IDNS algorithm was run 10 times on each instance in the range of $5 \leq N \leq 200$ and on several selected large-scale instances with $201 \leq N \leq 1080$ to assess its average performance. The stopping criterion of the algorithm is a maximum time limit t_{max} which was set according to the size of instances. Considering that the ECPOS problem is very hard to solve especially for the large-scale instances and that the time complexity of evaluating the objective function in Eq.(10) is high, t_{max} was set to $5N$ minutes.

3.2. Computational results and comparison on the instances with $N \leq 200$

This section aims to assess the performance of the IDNS algorithm on the instances with $N \leq 200$ and show the structural features of best configurations found. Subsection 3.2.1 shows the computational results and makes a comparison with the best-known results in the literature. Subsection 3.2.3 plots the packing densities and presents some representative configurations for the best solutions found in this work.

[1] We indeed obtained a number of better results than those reported in this paper by using alternative parameters values.

[2] The executable code of the IDNS algorithm and the best solutions reported in this work are online available at `https://github.com/XiangjingLai/Tammes-problem`

3.2.1. Comparison with the best-known results

Table 2: Computational results and comparison on small instances in the range of $6 \leq N \leq 50$.

N	BKR (deg)	d_{best}(deg)	d_{avg} (deg)	d_{worst} (deg)	σ	time(s)
6	90.000000000	90.000000000	90.000000000	90.000000000	0.0	0
7	77.869542155	77.869542155	77.869542155	77.869542155	0.0	0
8	74.858492186	74.858492186	74.858492186	74.858492186	0.0	1
9	70.528779366	70.528779366	70.528779366	70.528779366	0.0	1
10	66.146821988	66.146821988	66.146821988	66.146821988	0.0	1
11	63.434948823	63.434948823	63.434948823	63.434948823	0.0	1
12	63.434948823	63.434948823	63.434948823	63.434948823	0.0	1
13	57.136703078	57.136703078	57.136703078	57.136703078	0.0	2
14	55.670569996	55.670569996	55.670569996	55.670569996	0.0	3
15	53.657850130	53.657850130	53.657850130	53.657850130	0.0	5
16	52.244395753	52.244395753	52.244395753	52.244395753	0.0	3
17	51.090328552	51.090328552	51.090328552	51.090328552	0.0	5
18	49.556654768	49.556654768	49.556654768	49.556654768	0.0	7
19	47.691914109	47.691914109	47.691914109	47.691914109	0.0	9
20	47.431036227	47.431036227	47.431036227	47.431036227	0.0	4
21	45.613223106	45.613223106	45.613223106	45.613223106	0.0	14
22	44.740161167	44.740161167	44.740161167	44.740161167	0.0	16
23	43.709964205	43.709964205	43.709964205	43.709964205	0.0	13
24	43.690767108	43.690767108	43.690767108	43.690767108	0.0	4
25	41.634461260	41.634461260	41.634461260	41.634461260	0.0	17
26	41.037661607	41.037661607	41.037661607	41.037661607	0.0	14
27	40.677600685	40.677600685	40.677600685	40.677600685	0.0	15
28	39.355143569	39.355143569	39.355143569	39.355143569	0.0	17
29	38.713651194	38.713651194	38.713651194	38.713651194	0.0	20
30	38.597115954	38.597115954	38.597115954	38.597115954	0.0	26
31	37.709829144	37.709829144	37.709829144	37.709829144	0.0	25
32	37.475213975	37.475213975	37.475213975	37.475213975	0.0	17
33	36.254552976	36.254552976	36.254552976	36.254552976	0.0	22
34	35.807784396	35.807784396	35.807784396	35.807784396	0.0	28
35	35.319807591	35.319807591	35.319807591	35.319807591	0.0	29
36	35.189732258	35.189732258	35.189732258	35.189732258	0.0	31
37	34.422408009	34.422408009	34.422408009	34.422408009	0.0	38
38	34.250660672	34.250660672	34.250660672	34.250660672	0.0	22
39	33.489046580	33.489046580	33.489046580	33.489046580	0.0	40
40	33.158356264	33.158356264	33.158356264	33.158356264	0.0	31
41	32.729094415	32.729094415	32.729094415	32.729094415	0.0	36
42	32.506386350	32.506386350	32.506386350	32.506386350	0.0	26
43	32.090624406	32.090624406	32.090624406	32.090624406	0.0	47
44	31.983423033	31.983423033	31.983423033	31.983423033	0.0	47
45	31.323081434	31.323081434	31.323081434	31.323081434	0.0	67
46	30.959163488	30.959163488	30.959163488	30.959163488	0.0	57
47	30.781815961	30.781815961	30.781815961	30.781815961	0.0	41
48	30.762785551	30.762785551	30.762785551	30.762785551	0.0	17
49	29.923585114	29.923585114	29.923585114	29.923585114	0.0	62
50	29.752956397	29.752956397	29.752956397	29.752956397	0.0	55
#Improved		0	0	0		
#Equal		45	45	45		
#Worse		0	0	0		

The computational results of our IDNS algorithm on the instances with $N \leq 200$ are summarized in Tables 2–5. The first and second columns of the tables give the sizes of instances (N) and the best-known results (BKR) in the literature in terms of the angular diameter $d \in [0, 360]$ (in degree) of packed circles, where the notation 'N/A' means that the corresponding result is not available. It should be noted that these best-known results were generated by previous researchers using various approaches, such as mathematical methods, construction algorithms and global optimization algorithms. Most of them were collected over past 30 years by Sloane et al. and are available online at a website maintained by Sloane [34]. The computational results of our IDNS algorithm are reported in the last five columns, including the

largest angular diameter of circles found over 10 independent runs (d_{best}), the average angular diameter (d_{avg}), the smallest angular diameter (d_{worst}), the standard deviation of angular diameter (σ), and the average computational time in seconds to reach its final result for each run of IDNS ($time$). In addition, the last rows '#Improved' '#Equal' and '#Worse' in Tables 2–4 show the number of instances for which the IDNS algorithm obtained a better, equal or worse result compared to the best-known result in the literature in terms of d_{best}, d_{avg} and d_{worst}.

Tables 2 and 3 show that the IDNS algorithm is very efficient for the instances in the range of $6 \leq N \leq 100$, which are widely studied in the literature. For 92 out of 95 instances, the proposed IDNS algorithm matched the best-known result with a success rate of 100%. Moreover, for the instance with $N = 97$, the IDNS algorithm consistently improves the best-known result. For other two instances, the IDNS algorithm matched the best-known results in terms of d_{best}, but the average result and worst result are worse than the best-known result with a success rate of less than 100%. In terms of computational efficiency, the IDNS algorithm is very fast for most instances. Specifically, the average computational time is less than 500 seconds except for 3 instances with $N \in \{81, 90, 94\}$.

Table 4 gives the computational results on the 50 instances with $101 \leq N \leq 150$, where the first 30 instances with $101 \leq N \leq 130$ were widely studied in the literature and the next 20 instances with $131 \leq N \leq 150$ were mainly studied in [43]. One observes that the IDNS algorithm improved the best-known results for 26 out of these 50 instances in this range, where 6 instances lie in the range of $N \in [101, 130]$ and 20 instances lie in the range of $N \in [131, 150]$. Nevertheless, the IDNS algorithm missed the best-known results for 3 instances, implying that they are hard instances for the IDNS algorithm. In terms of d_{avg}, our IDNS algorithm obtains a better or equal result with respect to the best-known result for 24 and 21 instances, respectively, indicating a strong search ability of the algorithm. Moreover, the standard deviation of the objective values is very small for most instances, implying a good robustness of the algorithm. The computational time which reflects the hardness of instances, varies drastically in the interval $[200, 30000]$ depending on the instances to be solved.

Table 3: Computational results and comparison on the instances in the range of $51 \leq N \leq 100$. The improved results are indicated in bold compared with the best known results in the literature in terms of d_{best}, d_{avg} and d_{worst}.

N	BKR (deg)	d_{best} (deg)	d_{avg} (deg)	d_{worst} (deg)	σ	time(s)
51	29.368406881	29.368406881	29.368406881	29.368406881	0.0	59
52	29.194757905	29.194757905	29.194757905	29.194757905	0.0	43
53	28.813897205	28.813897205	28.813897205	28.813897205	0.0	63
54	28.716920530	28.716920530	28.716920530	28.716920530	0.0	81
55	28.262791418	28.262791418	28.262791418	28.262791418	0.0	60
56	28.148046651	28.148046651	28.148046651	28.148046651	0.0	51
57	27.826675948	27.826675948	27.826675948	27.826675948	0.0	61
58	27.556415956	27.556415956	27.556415956	27.556415956	0.0	69
59	27.394975670	27.394975670	27.394975670	27.394975670	0.0	72
60	27.192830003	27.192830003	27.192830003	27.192830003	0.0	65
61	26.873277866	26.873277866	26.873277866	26.873277866	0.0	80
62	26.683996996	26.683996996	26.683996996	26.683996996	0.0	83
63	26.486922511	26.486922511	26.486922511	26.486922511	0.0	66
64	26.235043312	26.235043312	26.235043312	26.235043312	0.0	80
65	26.069829948	26.069829948	26.069829948	26.069829948	0.0	102
66	25.947443691	25.947443691	25.947443691	25.947443691	0.0	80
67	25.683981345	25.683981345	25.683981345	25.683981345	0.0	101
68	25.463824458	25.463824458	25.463824458	25.463824458	0.0	69
69	25.333636438	25.333636438	25.333636438	25.333636438	0.0	271
70	25.170919984	25.170919984	25.170919984	25.170919984	0.0	146
71	24.987938062	24.987938062	24.987938062	24.987938062	0.0	424
72	24.926486081	24.926486081	24.926486081	24.926486081	0.0	131
73	24.553779249	24.553779249	24.553779249	24.553779249	0.0	177
74	24.420939780	24.420939780	24.420939780	24.420939780	0.0	121
75	24.301722513	24.301722513	24.301722513	24.301722513	0.0	192
76	24.128194442	24.128194442	24.128194442	24.128194442	0.0	263
77	24.001283683	24.001283683	24.001283683	24.001283683	0.0	180
78	23.931025420	23.931025420	23.931025420	23.931025420	0.0	275
79	23.623991696	23.623991696	23.623991696	23.623991696	0.0	134
80	23.553067202	23.553067202	23.553067202	23.553067202	0.0	159
81	23.347637682	23.347637682	*23.34 7637632*	*23.34 7637599*	4.09E-08	6557
82	23.194607406	23.194607406	23.194607406	23.194607406	0.0	468
83	23.082997639	23.082997639	23.082997639	23.082997639	0.0	123
84	23.051730642	23.051730642	23.051730642	23.051730642	0.0	152
85	22.779162071	22.779162071	22.779162071	22.779162071	0.0	207
86	22.674369389	22.674369389	22.674369389	22.674369389	0.0	179
87	22.546657426	22.546657426	22.546657426	22.546657426	0.0	163
88	22.467881045	22.467881045	22.467881045	22.467881045	0.0	119
89	22.316602355	22.316602355	22.316602355	22.316602355	0.0	243
90	22.154023258	22.154023258	22.154023258	22.154023258	0.0	1002
91	22.051796329	22.051796329	22.051796329	22.051796329	0.0	197
92	22.027581468	22.027581468	22.027581468	22.027581468	0.0	174
93	21.810380126	21.810380126	21.810380126	21.810380126	0.0	244
94	21.723713484	21.723713484	*21.723709135*	*21.723708651*	1.45E-06	1320
95	21.594550060	21.594550060	21.594550060	21.594550060	0.0	226
96	21.520609925	21.520609925	21.520609925	21.520609925	0.0	183
97	21.400619755	**21.400650276**	**21.400650276**	**21.400650276**	0.0	309
98	21.371060736	21.371060736	21.371060736	21.371060736	0.0	153
99	21.135967381	21.135967381	21.135967381	21.135967381	0.0	248
100	21.031202000	21.031202000	21.031202000	21.031202000	0.0	229
#Improved		1	1	1		
#Equal		49	47	47		
#Worse		0	2	2		

Table 5 shows the computational results on the instances in the range of $151 \leq N \leq 200$ which were rarely studied in the literature, where the available best-known results for several instances are taken from the website maintained by Sloane [14] and two previous papers [12, 42]. For most instances in this range, the putatively optimum solutions are generated for the first time by us. One can observe from Table 5 that for most instances, the standard deviation of the objective values obtained by the IDNS algorithm is very small, indicating a good algorithmic robustness. For the eight instances whose best-known results have been reported in the previous studies, the best, average and worst objective values of the IDNS algorithm are superior

to the best-known results except for $N = 200$. On the other hand, the table shows that the computational time of the IDNS algorithm on these large instances is usually much longer than on the smaller instances in Tables 2–4, which means the difficulty increases significantly as the size N increases in the general cases.

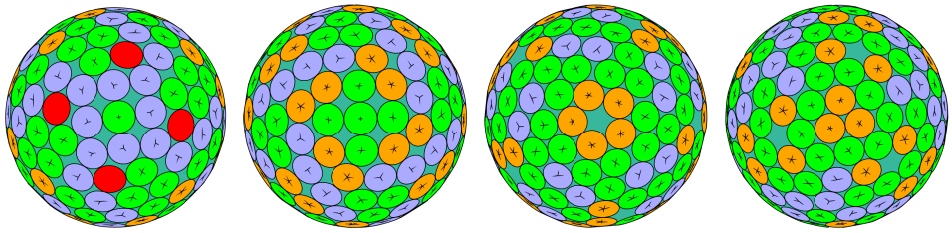
Table 4: Computational results and comparison on the instances in the range of $101 \leq N \leq 150$. The improved results are indicated in bold compared with the best known results in the literature in terms of d_{best}, d_{avg} and d_{worst}, and the worse results are indicated in italic.

N	BKR (deg)	d_{best} (deg)	d_{avg}(deg)	d_{worst}(deg)	σ	time(s)
101	20.928683418	20.928683418	20.928683418	20.928683418	0.0	231
102	20.855688715	20.855688715	20.855688715	20.855688715	0.0	1029
103	20.738269268	20.738269268	20.738269268	20.738269268	0.0	444
104	20.656620961	20.656620961	20.656620961	20.656620961	0.0	332
105	20.538852367	20.538852367	20.538852367	20.538852367	0.0	245
106	20.439408913	20.439408913	20.439408913	20.439408913	0.0	357
107	20.361203471	20.361203471	*20.36119994 7*	*20.361198437*	2.31E-06	5073
108	20.304444715	20.304444715	20.304444715	20.304444715	0.0	324
109	20.149319591	20.149319591	20.149319591	20.149319591	0.0	579
110	20.111327602	*20.096120236*	*20.096117678*	*20.096117039*	1.28E-06	4025
111	19.982476901	**19.993724800**	*19.975084380*	*19.968707889*	1.02E-02	6454
112	19.891304375	19.891304375	19.891304375	19.891304375	0.0	8305
113	19.805601302	19.805601302	19.805601302	19.805601302	0.0	287
114	19.745009357	19.745009357	19.745009357	19.745009357	0.0	309
115	19.623993121	19.623993121	19.623993121	19.623993121	0.0	1854
116	19.549796869	19.549796869	19.549796869	19.549796869	0.0	378
117	19.461291100	19.461291100	19.461291100	19.461291100	0.0	284
118	19.389349705	**19.389510434**	**19.389510434**	**19.389510434**	0.0	3504
119	19.325751352	*19.314184273*	*19.314061399*	*19.313118887*	3.18E-04	21104
120	19.324020069	*19.264740121*	*19.264740121*	*19.264740121*	0.0	251
121	19.135729782	**19.146164522**	**19.146161475**	**19.146151839**	4.20E-06	21761
122	19.070036856	**19.075939806**	**19.071748646**	19.070036856	2.18E-03	4927
123	19.006389067	19.006389067	19.006389067	19.006389067	0.0	678
124	18.953911647	18.953911647	18.953911647	18.953911647	0.0	633
125	18.844815070	**18.844831207**	*18.836924324*	*18.832117202*	5.93E-03	12947
126	18.781585614	18.781585614	18.781585614	18.781585614	0.0	302
127	18.690056810	**18.690313825**	**18.690133280**	18.690056810	1.13E-04	9012
128	18.634972596	18.634972596	18.634972596	18.634972596	0.0	1050
129	18.563472647	18.563472647	18.563472647	18.563472647	0.0	359
130	18.510352167	18.510352167	18.510352167	18.510352167	0.0	295
131	18.2831860	**18.420047209**	**18.420047209**	**18.420047209**	0.0	397
132	18.3665155	**18.384277320**	**18.384275633**	**18.384268888**	3.37E-06	12680
133	18.1164476	**18.273124165**	**18.273124165**	**18.273124165**	0.0	371
134	18.0352521	**18.200068456**	**18.200068456**	**18.200068456**	0.0	1590
135	17.8995373	**18.136512220**	**18.136512220**	**18.136512220**	0.0	464
136	17.8995373	**18.078032313**	**18.077511271**	**18.075228366**	9.95E-04	29274
137	17.8597825	**18.005882786**	**18.005882786**	**18.005882786**	0.0	389
138	17.6717639	**17.943942754**	**17.943942739**	**17.943942678**	3.04E-08	20303
139	17.6355493	**17.883394318**	**17.883394318**	**17.883394318**	0.0	3195
140	16.5945958	**17.828027560**	**17.828027560**	**17.828027560**	0.0	1365
141	17.5598327	**17.751660456**	**17.747822554**	**17.746935482**	1.78E-03	10294
142	17.4217485	**17.693922247**	**17.693922247**	**17.693922247**	0.0	682
143	17.4136350	**17.636328052**	**17.622533056**	**17.621000295**	4.60E-03	2635
144	17.4803100	**17.592390582**	**17.583784340**	**17.582828097**	2.87E-03	413
145	17.3587143	**17.520387535**	**17.508005494**	**17.496640380**	9.33E-03	28443
146	17.3312211	**17.466893736**	**17.454847184**	**17.440061598**	1.17E-02	24851
147	17.1801361	**17.401249997**	**17.395218183**	**17.383314817**	7.42E-03	22485
148	17.0450209	**17.343053927**	**17.342073075**	**17.340310702**	7.77E-04	23831
149	17.0020507	**17.281266525**	**17.281266525**	**17.281266525**	0.0	701
150	17.1075768	**17.249816554**	**17.249816554**	**17.249816554**	0.0	696
#Improved		26	24	22		
#Equal		21	21	23		
#Worse		3	5	5		

Table 5: Computational results and comparison on the instances in the range of $151 \leq N \leq 200$. The improved results are indicated in bold compared with the best-known results in the literature in terms of d_{best}, d_{avg} and d_{worst}.

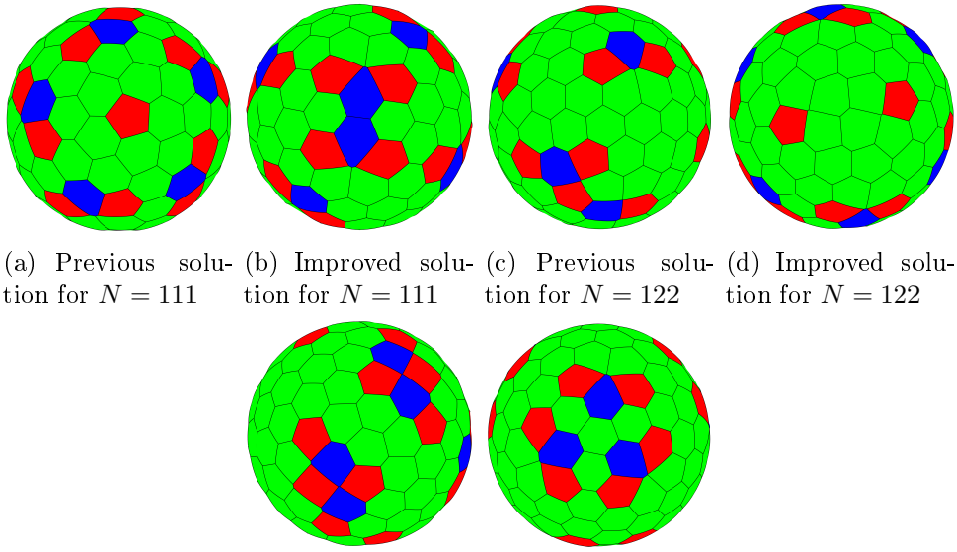
N	BKR (deg)	d_{best}(deg)	d_{avg}(deg)	d_{worst} (deg)	σ	time(s)
151	N/A	17.174585202	17.174335564	17.173790743	2.75E-04	28925
152	16.22487558	**17.127321801**	**17.125969412**	**17.114890190**	3.70E-03	28376
153	N/A	17.072410149	17.070127387	17.050645569	6.50E-03	31578
154	N/A	17.020367720	17.020306724	17.020198307	5.93E-05	31959
155	N/A	16.957032553	16.957011737	16.956958629	2.82E-05	36105
156	N/A	16.921870518	16.916458988	16.896145681	1.01E-02	28653
157	N/A	16.844221939	16.843639388	16.841328510	1.04E-03	34454
158	N/A	16.795708629	16.795185682	16.794085637	4.86E-04	32221
159	N/A	16.746475459	16.742873946	16.736992394	4.27E-03	26687
160	N/A	16.693204542	16.688543911	16.683883308	4.66E-03	21204
161	N/A	16.638778470	16.636422193	16.632985911	2.00E-03	34002
162	16.132192103	**16.606770297**	**16.602243974**	**16.594900388**	4.93E-03	26900
163	N/A	16.541137966	16.540163192	16.534523100	1.92E-03	30097
164	N/A	16.485809203	16.484338634	16.483441050	9.05E-04	20602
165	N/A	16.434849648	16.433901863	16.433104855	6.11E-04	33673
166	N/A	16.397996521	16.397966946	16.397743224	7.56E-05	37243
167	N/A	16.336594952	16.331890626	16.327290716	3.75E-03	34163
168	N/A	16.293215766	16.289906102	16.285491526	2.59E-03	32061
169	N/A	16.255546787	16.250942915	16.234537037	5.86E-03	25506
170	14.845631806	**16.236217883**	**16.236081108**	**16.235783104**	1.54E-04	28226
171	N/A	16.166465249	16.163859419	16.159066566	2.47E-03	31094
172	N/A	16.130878909	16.123784273	16.107224050	1.08E-02	41078
173	N/A	16.056631001	16.056626463	16.056603037	8.24E-06	39171
174	N/A	16.022382474	16.021977313	16.020436609	7.12E-04	39804
175	N/A	15.985629022	15.984740458	15.983094748	9.48E-04	39044
176	N/A	15.930734011	15.928351550	15.925833252	1.81E-03	36209
177	N/A	15.881694787	15.881482809	15.880798140	2.74E-04	34020
178	N/A	15.853104024	15.852002724	15.848687784	1.67E-03	41147
179	N/A	15.825647953	15.825626868	15.825488740	4.64E-05	44799
180	15.818759283	**15.818759336**	**15.818759336**	**15.818759336**	0.0	34856
181	N/A	15.738653922	15.738644273	15.738627742	6.57E-06	28114
182	14.515037788	**15.676673420**	**15.676534933**	**15.675344730**	3.97E-04	38433
183	N/A	15.637263188	15.635386225	15.632446644	1.90E-03	29236
184	N/A	15.586160863	15.586153058	15.586089294	2.13E-05	46139
185	N/A	15.549278801	15.549094271	15.548645292	2.04E-04	48282
186	N/A	15.506034396	15.506029750	15.506008334	7.91E-06	30011
187	N/A	15.451118279	15.451106542	15.451043774	2.45E-05	32681
188	N/A	15.420083491	15.420083403	15.420083394	2.91E-08	27161
189	N/A	15.379486067	15.379418380	15.379108477	1.08E-04	42820
190	N/A	15.357312790	15.356122097	15.348075717	2.73E-03	44676
191	N/A	15.311391737	15.311391737	15.311391737	0.0	23064
192	15.17866313	**15.293846752**	**15.293846752**	**15.293846752**	0.0	17657
193	N/A	15.226804266	15.226804198	15.226803926	1.36E-07	25765
194	N/A	15.185373070	15.185019272	15.183262743	6.08E-04	40756
195	N/A	15.148661157	15.137439936	15.134135175	5.39E-03	40147
196	N/A	15.099589969	15.097575664	15.096137624	1.18E-03	37635
197	N/A	15.055765691	15.054106350	15.052257521	1.28E-03	30738
198	14.60186	**15.021343394**	**15.020387794**	**15.018682441**	9.38E-04	40019
199	N/A	15.001874047	14.994478168	14.989824292	4.28E-03	37995
200	14.995766166	**14.996344132**	*14.973669204*	*14.947701872*	1.62E-02	43699

To give an intuitive interpretation for the solutions of the ECPOS problem, Fig. 3 provides a graphical representation of the best solutions of four representative instances, i.e., $N = 149$, 150, 171 and 177, where each point on the sphere corresponds to a circle with angular diameter of d_{best}. Fig. 3 shows that the best packing configuration found for these four instances has a high symmetry.



(a) $N = 149$ (b) $N = 150$ (c) $N = 171$ (d) $N = 177$

Figure 3: Best packing configurations found in this work for four representative instances, where each circle is colored according to its number of nearest neighbors.



(a) Previous solution for $N = 111$ (b) Improved solution for $N = 111$ (c) Previous solution for $N = 122$ (d) Improved solution for $N = 122$

(e) Previous solution for $N = 127$ (f) Improved solution for $N = 127$

Figure 4: Comparisons of Voronoi representations between the previous best-known solution and the improved best solution for three representative instances in the range of $N \leq 130$.

As demonstrated above, our IDNS algorithm improves the best-known results for 7 out of 125 instances with $N \leq 130$ which are the most studied in the literature. To disclose the differences between the improved solutions and the previous best-known solutions, we give the Voronoi representations for three representative instances in Fig. 4, since be the Voronoi representation

of solutions is able to more clearly exhibit the features of the configuration compared to the direct representation in Fig. 3.

The Voronoi representation is a configuration consisting of N points on the sphere, constituting a partition of the spherical surface S^2. Specifically, the spherical surface S^2 is partitioned to N disjoint Voronoi cells C_1, C_2, ..., C_N, where each cell C_i is a polygon on the sphere and corresponds to a given point r_i that is the spherical cap center. Each polygon is colored according to the number of its edges to portray its configuration characteristics. Formally, a cell C_i on S^2 can be written as follows [33]:

$$C_i = \{x \in S^2 : |x - r_i| = min_{1 \leq k \leq N} |x - r_k|\} \tag{14}$$

The configuration of solutions in the Voronoi representation is characterized by its topological defects, where a topological defect can be defined as a building block of one or several adjacent non-hexagonal cells. Non-hexagonal cells are inevitable for the configurations on a sphere according to Euler's formula $F - E + V = 2$, where F is the number of faces, E is the number of edges, and V is the number of vertices [33]. In the previous studies of the well-known Thomson problem [46, 47], the topological defects are widely used to characterize spherical configurations. In this study, the Voronoi representation is able to portray a fuller range of the structural differences between solutions. The configurations of Fig. 4 show that the improved solutions differ significantly from the previous best-known solutions for these three representative instances.

3.2.2. Comparison with the popular basin-hopping algorithm

To further assess the proposed IDNS algorithm, we compared it with the basin-hopping (BH) algorithm [45], which is a very popular Monte Carlo search based global optimization algorithm in the literature. At each iteration, the algorithm performs a perturbation move followed by a local optimization step to generate the new solution and then uses the Metropolis rule to accept the new solution. Specifically, the Metropolis acceptance rule is based on the objective change (i.e., $\Delta_f = f(X_{new}) - f(X_{cur})$) between the current solution X_{cur} and the new solution X_{new} and a temperature parameter T, and the new solution is accepted to become the current solution with a probability of $min\{1, e^{-\frac{\Delta_f}{T}}\}$.

To conduct our comparative study, we first created a variant BH* of the IDNS algorithm, where the DNS procedure is replaced by the basin-hopping

algorithm while keeping other algorithmic components unchanged. In this experiment, the BH* algorithm and the IDNS algorithm were respectively performed 10 times for each of 20 selected hard instances. The computational results are reported in Table 6, including the best objective values over 10 runs (d_{best}), the average objective values (d_{avg}) and the worst objective values (d_{worst}). The last rows '#Better', '#Equal' and '#Worse' of table respectively give the numbers of instances for which an algorithm obtained a better, equal and worse result compared to its competitor.

Table 6 shows that the IDNS algorithm outperforms significantly the BH* algorithm on the tested instances. In terms of d_{best}, the IDNS algorithm obtained a better and worst result than BH* respectively for 18 and 1 out of the 20 instances. In terms of d_{avg} and d_{worst}, the IDNS algorithm obtained a better result respectively for 19 and 18 instances. To assess the statistical significant differences between the results of the compared algorithms, the Wilcoxon signed-rank test was applied to the values of d_{best}, d_{avg} and d_{worst}, leading to p-values smaller than 0.05. This experiment indicates that the IDNS algorithm is a very competitive global optimization algorithm on the studied ECPOS problem.

Table 6: Comparison between the IDNS algorithm with the popular basin-hopping (BH) algorithm [45] on 20 hard instances in the range of $N \leq 200$. The dominating results are indicated in bold in terms of d_{best}, d_{avg} and d_{worst}.

N	d_{best}		d_{avg}		d_{worst}	
	BH*	IDNS	BH*	IDNS	BH*	IDNS
127	18.690305208	**18.690313825**	**18.690147913**	18.690133280	18.690050921	**18.690056810**
146	17.465930886	**17.466893736**	17.451844964	**17.454847184**	**17.441095528**	17.440061598
148	17.341860444	**17.343053927**	17.338237872	**17.342073075**	17.327139650	**17.340310702**
151	17.174567161	**17.174585202**	17.158951212	**17.174335564**	17.151670952	**17.173790743**
153	17.071090417	**17.072410149**	17.054333877	**17.070127387**	17.049804683	**17.050645569**
156	16.921411941	**16.921870518**	16.900683613	**16.916458988**	16.893047572	**16.896145681**
157	16.841950790	**16.844221939**	16.838713157	**16.843639388**	16.835548149	**16.841328510**
159	16.738050945	**16.746475459**	16.736353245	**16.742873946**	16.732368911	**16.736992394**
160	**16.693204542**	**16.693204542**	16.684477146	**16.688543911**	16.681472974	**16.683883308**
166	16.397924415	**16.397996521**	16.396660396	**16.397966946**	16.391921250	**16.397743224**
168	16.290928822	**16.293215766**	16.288824441	**16.289906102**	**16.285974926**	16.285491526
170	16.197343742	**16.236217883**	16.194450377	**16.236081108**	16.191882989	**16.235783104**
174	16.019628627	**16.022382474**	16.016837296	**16.021977313**	16.013224168	**16.020436609**
176	15.925604169	**15.930734011**	15.922735014	**15.928351550**	15.920273618	**15.925833252**
183	15.637185316	**15.637263188**	15.624755792	**15.635386225**	15.617711750	**15.632446644**
185	15.547603918	**15.549278801**	15.541858678	**15.549094271**	15.538561262	**15.548645292**
189	15.378978288	**15.379486067**	15.375799277	**15.379418380**	15.367998919	**15.379108477**
195	15.133901510	**15.148661157**	15.132159787	**15.137439936**	15.131164954	**15.134135175**
196	**15.100659852**	15.099589969	15.096130061	**15.097575664**	15.093690564	**15.096137624**
200	14.955636253	**14.996344132**	14.951768478	**14.973669204**	14.945801468	**14.947701872**
#Better	1	18	1	19	2	18
#Equal	1	1	0	0	0	0
#Worse	18	1	19	1	18	2
p-value		4.63E-4		1.03E-4		2.19E-4

3.2.3. Packing density and representative packing patterns

This subsection investigates the packing density and the representative configurations of putatively optimal solutions. The packing density p of a feasible packing configuration of N equal circles with an angular diameter of d ($\in [0, \pi]$) on the unit sphere is defined as the ratio of the area of N spherical caps to the whole area of the spherical surface and is calculated as follows [43]:

$$p(N, d) = \frac{N}{2}(1 - cos(\frac{d}{2})) \tag{15}$$

The packing densities of the best configurations found in this study are plotted in Fig. 5 as a function of N for the instances in the range of $N \leq 200$. We observe from Fig. 5 that the packing densities of these best configurations vary markedly at the beginning and then trend gradually toward a stable status as the number (N) of circles increases. On the other hand, we also observe that there are some special sizes for which the packing density is much higher or lower than its neighboring sizes, such as $N = 7$, 12, 13, 24, 48, 72, 98 and 180.

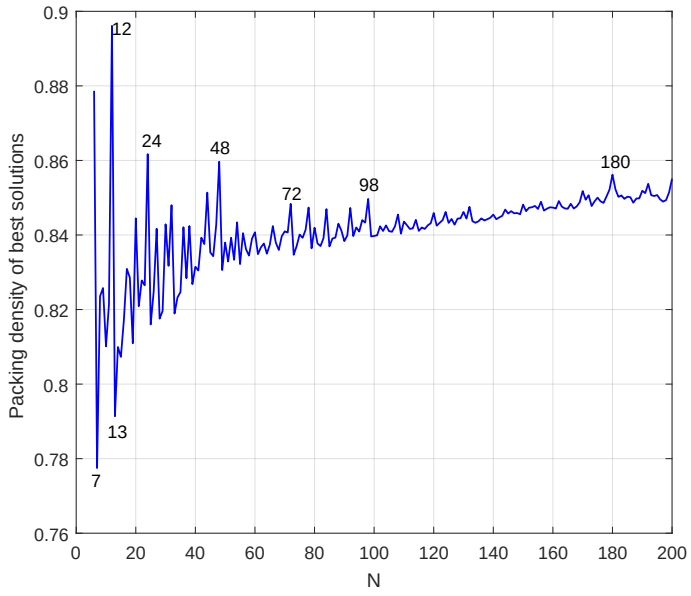
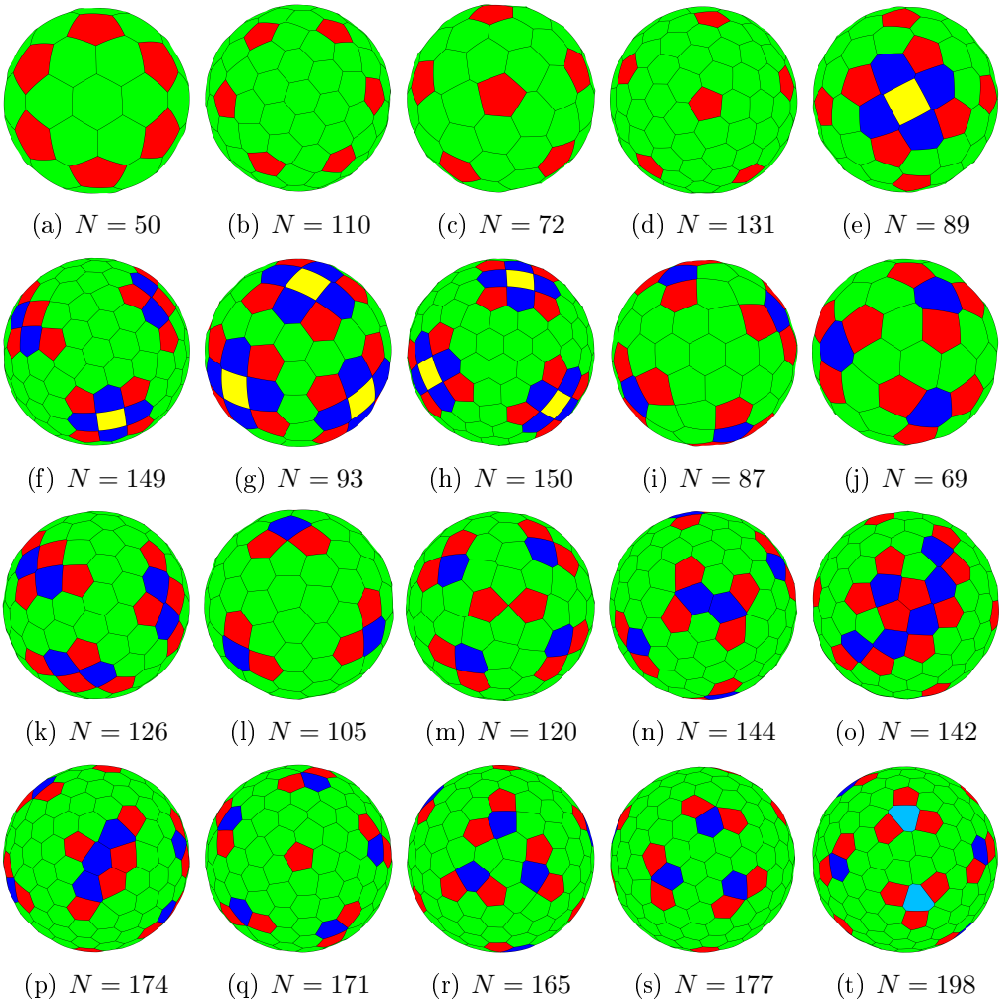


Figure 5: The packing densities of putatively optimal solutions as a function of the size N of instances.



(a) $N = 50$ (b) $N = 110$ (c) $N = 72$ (d) $N = 131$ (e) $N = 89$

(f) $N = 149$ (g) $N = 93$ (h) $N = 150$ (i) $N = 87$ (j) $N = 69$

(k) $N = 126$ (l) $N = 105$ (m) $N = 120$ (n) $N = 144$ (o) $N = 142$

(p) $N = 174$ (q) $N = 171$ (r) $N = 165$ (s) $N = 177$ (t) $N = 198$

Figure 6: Voronoi representations of the best solutions for some representative instances.

To show the structural characteristics of the best solutions found, we give in Fig. 6 the Voronoi representation for several representative instances, where the structural characteristics are well exhibited by the topological defects. According to the figure, the structural characteristics of these best configurations can be summarized as follows. First, these best solutions exhibit various categories of packing patterns which differ significantly for $N = 50$, 72, 87, 89, 120, 126, 142 and 171. Second, some best solutions share almost the same packing pattern and the only difference between their configurations lies in the distance between topological defects. This phenomenon

occurs noticeably for $N = 50$ and 110, 72 and 131, and 93 and 150. Third, some configurations share the same topological defects, as exemplified by the cases for $N = 89$, 93, 126 and 149. Fourth, some configurations exhibit very similar but different packing patterns, such as $N = 105$ and 126, and so on.

3.3. Computational results on large-scale instances

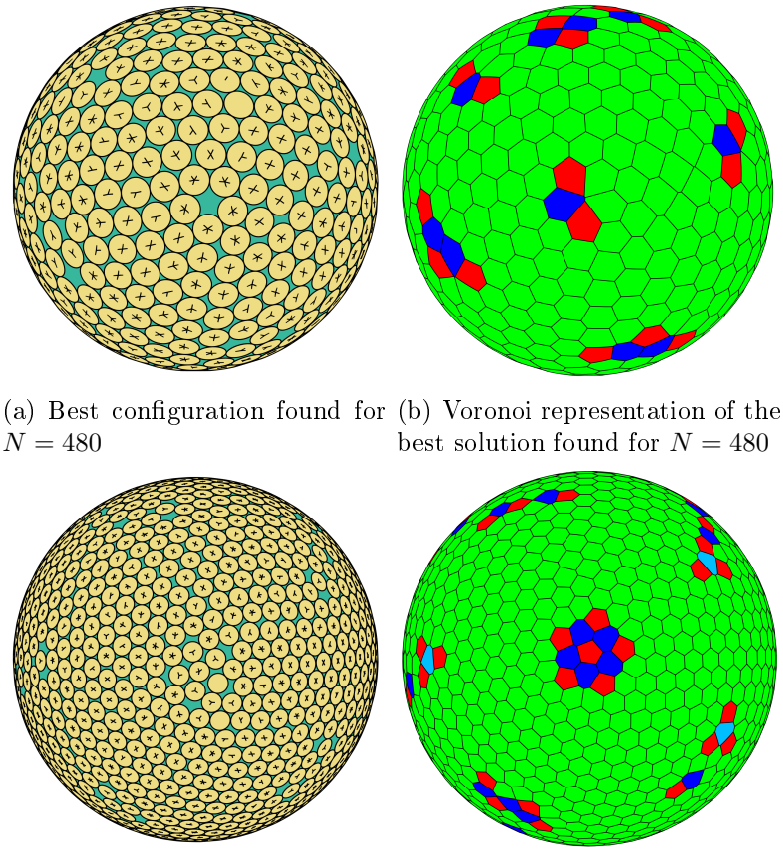
To assess the scalability of the algorithm, we tested the IDNS algorithm on 10 selected large-scale instances with N ranging from 216 to 1080 commonly studied in previous investigations [12, 14, 38, 42]. The computational results are summarized in Table 7, where most of the best-known results were obtained by the construction methods based on prior knowledge of the problem and the symbols have the same meaning as in Tables 2–4.

Table 7: Computational results and comparison on 10 large-scale instances with $N > 200$. The improved results are indicated in bold compared with the best-known results in the literature in terms of d_{best}, d_{avg} and d_{worst}, and the worse results are indicated by italic.

N	BKR (deg)	d_{best}(deg)	d_{avg}(deg)	d_{worst}(deg)	σ	time(s)
216	14.21184	**14.39684**	**14.38763**	**14.38318**	4.70E-03	46560
270	12.93699	12.93699	*12.91552*	*12.89962*	1.75E-02	47296
282	12.44139	**12.62615**	**12.62492**	**12.62147**	1.50E-03	61031
360	11.20247	*11.16916*	*11.16604*	*11.16304*	1.76E-03	86359
372	10.92372	**10.99022**	**10.98682**	**10.98404**	1.85E-03	66034
432	9.98344	**10.19665**	**10.19331**	**10.19099**	1.67E-03	88961
480	9.69375	*9.67197*	*9.67059*	*9.66887*	9.53E-04	107098
492	9.46111	**9.55591**	**9.55240**	**9.55028**	1.97E-03	93310
750	7.74674	**7.75782**	**7.75632**	**7.75531**	7.09E-04	165102
1080	6.44607	**6.47834**	**6.47737**	**6.47618**	6.48E-04	232943
#Improved		7	7	7		
#Equal		1	0	0		
#Worse		2	3	3		

Table 7 shows that the IDNS algorithm is very effective compared to other algorithms in the literature at obtaining the best-known results for these large-scale instances, improving the best-known result for 7 instances and matching the best-known result for one instance, while missing the best-known results only for two instances. The average result d_{avg} and the worst result d_{worst} over 10 independent runs of the IDNS algorithm are both superior to the best-known results for 7 instances. Such an outcome discloses that the multi-symmetric packing configurations obtained by the construction methods are not optimal even if they have a beautiful graphical representation and are widely conjectured to be the optimal solutions [14].

Nevertheless, the fact that the IDNS algorithm failed to find the best-known results for two instances indicates that the best-performing construction approaches still can be competitive for some special sizes compared to the IDNS algorithm. In terms of the computational time, the results of the IDNS algorithm are relatively long for these instances, which means that they are much harder to solve compared to the smaller instances with $N \leq 200$.



(a) Best configuration found for $N = 480$

(b) Voronoi representation of the best solution found for $N = 480$

(c) Best configuration found for $N = 1080$

(d) Voronoi representation of the best solution found for $N = 1080$

Figure 7: Best configurations found in this work and their Voronoi representations for two selected large-scale instances.

To provide an intuitive picture of the best configurations of the large-scale instances, we give in Fig. 7 the packing configurations and the corresponding Voronoi representations of the best solutions found for two representative

instances with $N = 480$ and 1080. The configurations in Fig. 7 show that the surface S^2 of the unit sphere contains a number of vacancies that were not covered by circles, leading to several regular topological defects in the Voronoi representations.

4. Analysis of key algorithmic components

We now turn our attention to analyzing two important components of the IDNS algorithm: the setting of key parameters and the multiple-stage local optimization method.

4.1. Sensitivity of key parameters

The DNS procedure of the proposed IDNS algorithm employs several parameters. This subsection analyzes their sensitivity and undertakes to find the appropriate settings for them. Due to a high correlation between parameters, we focus on the combination of three parameters M_{max}, η_{max}, η_s, where M_{max} represents the maximum neighborhood size , η_{max} represents the maximum perturbation strength, and η_s represents the incremental change of perturbation strength per iteration.

To check the sensitivity of these three parameters, we carried out an additional experiment based on 20 representative instances in the range of $N \leq 200$, where 9 parameter combinations shown in Table 8 were tested. For each parameter combination, the IDNS algorithm was run 5 times on each instance, and the computational results are summarized in Table 8, where the first column gives the sizes (N) of instances, the second row gives the parameter settings, columns 2-10 give the average objective values d_{avg} (in degree), and the last row shows the number of instances for which the associated parameter combination produced the best result among all the tested parameter combinations.

Table 8 shows that the performance of the IDNS algorithm is sensitive to the setting of these three parameters. The parameter combination of (M_{max},η_{max},η_s)= $(6, 0.6, 0.01)$ produced the best result in terms of the average objective value for 10 out of the 20 instances, and other 8 combinations respectively produced the best result for 0, 3, 1, 0, 2, 0 , 4, 0 instances, respectively. This outcome indicates $(6, 0.6, 0.01)$ is the best combination for the parameters (M_{max}, η_{max},η_s) among the combinations tested, and thus was chosen as the default setting for the IDNS algorithm.

Table 8: Influence of the settings of parameters on the performance of algorithm. Each instance was solved 5 times for each parameter combination of (M_{max},η_{max},η_s), and the average objective values (d_{avg}) over 5 runs are given in the table, where the best results among the tested parameter combinations are indicated in bold for each instance.

	d_{avg}								
N/(M_{max},η_{max},η_s)	(3, 0.7, 0.03)	(4, 0.7, 0.01)	(4, 0.7, 0.03)	(4, 0.7, 0.05)	(5, 0.7, 0.01)	(5, 0.7, 0.03)	(6, 0.6, 0.01)	(6, 0.7, 0.01)	(6, 1.0, 0.01)
127	18.6901498	18.6901082	18.6902397	18.6902494	18.6901082	18.6901958	18.6901583	**18.6902584**	18.6901063
146	17.4482676	**17.4582671**	17.4477357	17.4469208	17.4471473	17.4479971	17.4472764	17.4509179	17.4461207
148	17.3382886	17.3387882	**17.3417278**	17.3384857	17.3356355	17.3326310	17.3356970	17.3394506	17.3353972
151	17.1644585	17.1633563	17.1681313	17.1580338	17.1582059	17.1685724	**17.1738019**	17.1630695	17.1695592
153	17.0590791	17.0714462	17.0589176	17.0580142	17.0604717	17.0575317	**17.0683029**	17.0632901	17.0672641
156	16.9090326	16.9048533	16.9093797	16.9166010	**16.9216265**	16.9024000	16.9179374	16.9211816	16.9169318
157	16.8404528	16.8434530	16.8417584	16.8403518	16.8421071	16.8417126	**16.8439651**	16.8408719	16.8417145
159	16.7365568	16.7431259	16.7381247	16.7373414	16.7431552	16.7366532	16.7407873	**16.7444778**	16.7432388
160	16.6847527	16.6854405	16.6854889	16.6861354	16.6838720	16.6870552	**16.6881013**	16.6873830	16.6820394
166	16.3923725	16.3952353	16.3932665	16.3966948	16.3956327	16.3898364	**16.3979965**	16.3944327	16.3958718
168	16.2876051	**16.2936941**	16.2896172	16.2887163	16.2899262	16.2892704	16.2901879	16.2896299	16.2894372
170	16.2163142	**16.2348095**	16.2153551	16.2066063	16.2193611	16.2107841	16.2125983	16.2245123	16.2251980
174	16.0158297	16.0193235	16.0188455	16.0184606	16.0175039	16.0145093	**16.0214495**	16.0192027	16.0214050
176	15.9270161	15.9246417	15.9276029	15.9253874	15.9275527	15.9246041	**15.9296936**	15.9287096	15.9275110
183	15.6320341	15.6346540	15.6346371	15.6292564	15.6354346	15.6328937	15.6348465	**15.6355193**	15.6334947
185	15.5410014	15.5453560	15.5443835	15.5408988	15.5446377	15.5443810	**15.5480259**	15.5443088	15.5467411
189	15.3747624	15.3775044	15.3739427	15.3744127	15.3759129	15.3741625	**15.3785468**	15.3770740	15.3779191
195	15.1327735	15.1332095	15.1324183	15.1357594	**15.1399524**	15.1324085	15.1349263	15.1341429	15.1340331
196	15.0952901	15.0963988	15.0977746	15.0946044	15.0980392	15.0952901	**15.0985888**	15.0980695	15.0958733
200	14.9528029	14.9725562	14.9592674	14.9510179	14.9716083	14.9533088	14.9688485	**14.9770950**	14.9544023
	0	3	1	0	2	0	10	4	0

Finally, our experiments indicated that the parameters M_{min}, η_{min} of the DNS method also significantly impact the performance of the IDNS algorithm, while this is not the case for the parameter β_{max}. To sum, the settings of parameters have a great influence on the performance of the IDNS algorithm. Fine-tuning these parameters could lead to new results still better than those reported in this paper.

4.2. Importance of multiple-stage local optimization

The multiple-stage local optimization method is a main component of the proposed IDNS algorithm. To check its efficiency and effectiveness, we carried out a comparative experiment based on all the instances in the range of $6 \leq N \leq 200$. In this experiment, for each instance, 1000 initial solutions were first randomly generated, and then from each initial solution the multiple-stage local optimization method and the standard LBFGS method (i.e., one-stage local optimization method) were run to minimize the objective function $E_D(X)$ defined in Eq. (10), where the value of D was set to the current best-known result. The average running time in seconds is plotted in Fig. 8 as a function of N for the two local optimization methods. Thus this is a time-to-target analysis, which illustrates the time needed for both methods to reach the same local minimum solution from a given starting point.

Fig. 8 shows that the multiple-stage local optimization method significantly outperforms the popular one-stage LBFGS method in terms of computational speed. The running time increases almost linearly for the multiple-stage local optimization method as the size N of instance increases, which is a desired feature for the local optimization. However, the running time of the one-stage LBFGS method increases almost quadratically with respect to N. Moreover, the running time of the one-stage local optimization method is much longer than that of the multiple-stage local optimization method especially for the large instances. This experiment clearly shows that the use of the multiple-stage local optimization method is able to significantly speed up the search process of the IDNS algorithm.

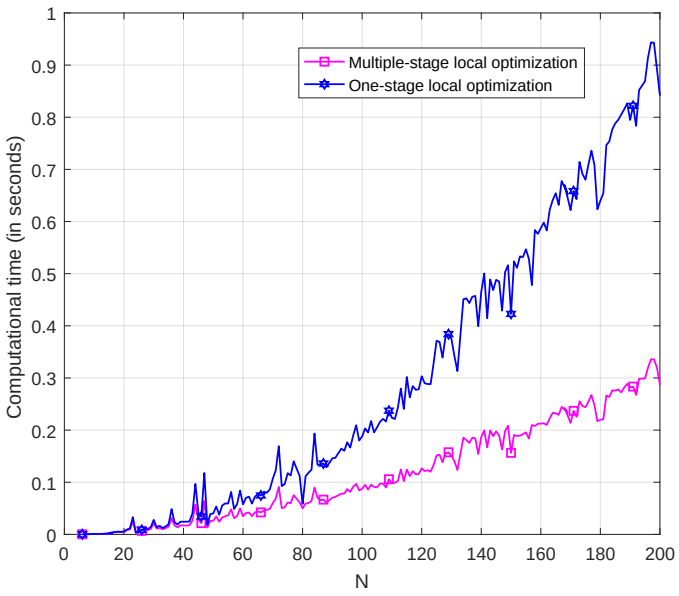


Figure 8: Comparison between the multiple-stage local optimization method and one-stage local optimization method.

To further check whether the multiple-stage local optimization strategy can be applied to other geometry optimization problems, we applied the method to the well-known Lennard-Jones cluster problem and conducted an experiment on 4 representative instances (LJ_{200}, LJ_{400}, LJ_{600} and LJ_{800}), where the multiple-stage local optimization method and the standard LBFGS method were respectively used in one run of the DNS algorithm for each tested cluster. The computational results show that the multiple-stage local optimization is faster than the standard LBFGS method on the Lennard-Jones clusters. Specifically, our multiple-stage local optimization method for 10^4 performed local optimizations requires on average 0.015, 0.066, 0.15 and 0.235 seconds to converge to the local minimum solutions respectively for LJ_{200}, LJ_{400}, LJ_{600} and LJ_{800} on our computer, while the standard LBFGS method requires 0.025, 0.115, 0.284, 0.480 seconds to reach the same results.

5. Generality of the DNS method

As one main component of the proposed IDNS algorithm, the dynamical neighborhood search (DNS) method described in Algorithm 2 is a general-

purpose optimization algorithm and can be applied to other optimization problems. To show its generality and effectiveness, we carried out an additional experiment by applying DNS to another well-known global optimization problem, i.e., the structural optimization of Lennard-Jones (LJ) clusters. In this experiment, the DNS method was performed in a multi-start fashion with the following parameter setting: $M_{min} = 5$, $M_{max} = 10$, $\eta_{min} = 0.8$, $\eta_{max} = 0.9$, $\eta_s = 0.01$ and $\beta_{max} = 300$.

To assess the performance of the DNS method, we used the very popular basin-hopping (BH) algorithm [45] as the reference algorithm. Both the temperature parameter T and the perturbation strength η of the BH algorithm were set to 0.8. To make a fair comparison, the two compared algorithms employed the same stopping condition, which is a maximum number $MaxStep$ of local optimizations fixed to 10^4.

The experimental results of these two algorithms are summarized in Table 9, where the first two columns give the sizes of the instances and the best-known results (BKR) reported in the literature. The results of the two algorithms are shown in the remaining columns. Columns 3–4 give the best objective values (f_{best}) obtained over 100 independent runs for the two algorithms, columns 5-6 present the average objective values (f_{avg}), and columns 7-8 show the success rates (SR) of hitting the best-known results for the two algorithm. The last row indicates the number of instances for which the corresponding algorithm yielded the best result between the compared algorithms in terms of f_{best}, f_{avg} and SR.

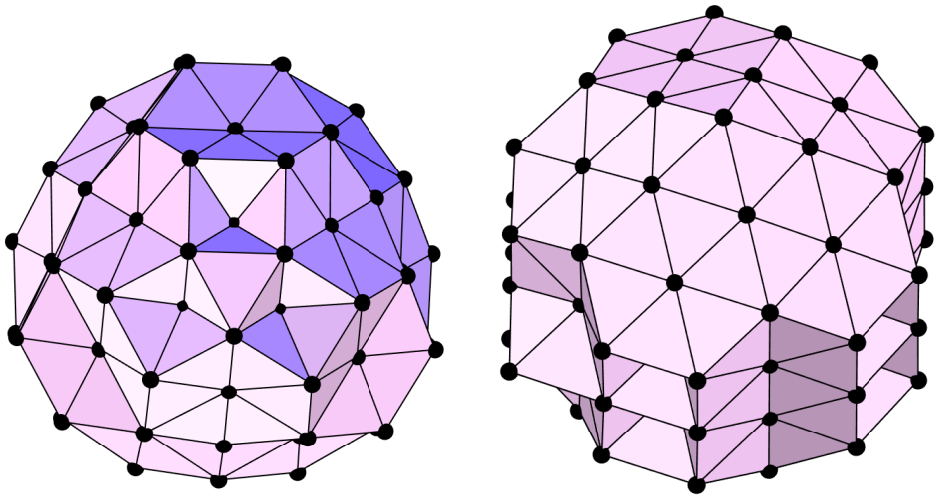
Table 9 discloses that the DNS algorithm performed better than the popular BH algorithm for the optimization of Lennard-Jones clusters. Specifically, in terms of f_{best}, both algorithms found the best-known solutions for the 10 tested instances, including the hardest instances LJ_{75}, LJ_{98} and LJ_{102} in the range of $N \leq 110$. For the average objective values, the DNS method obtained the best result for 9 instances, while the BH algorithm reached the best result only for 3 instances. In terms of the success rate of hitting the best-known solution, DNS obtained the best results for 9 instances against 5 instances for BH. For an intuitive interpretation of the solutions of Lennard-Jones clusters, we provide in Fig. 9 the best configurations found by our DNS method for two hard instances LJ_{98} and LJ_{102}. This experiment shows the competitiveness of the DNS method for finding Lennard-Jones clusters compared with the popular BH algorithm. This experiment provides an example of applying our DNS algorithm as a general-purpose method to solve other challenging global optimization problems with an differentiable objec-

tive function. It would be very interesting to check its effectiveness on other global optimization problems in the future.

In addition, it should be noted that there exist a number of successful specific optimization algorithms for the Lennard-Jones clusters in the literature, such as the monotonic global optimization based on a two-phase local search method [23] and the funnel hopping algorithm [6]. These algorithms outperform our DNS method especially on several hardest instances such as LJ_{75}, LJ_{98} and LJ_{102}. However, these algorithms employ more or less problem-specific knowledge of Lennard-Jones (LJ) clusters. Compared to those specialized methods, our DNS method has the advantage of wider applications due to its generality.

Table 9: Comparative results of the DNS method with the popular basin-hopping (BH) algorithm [45] on 10 representative Lennard-Jones clusters. The best results among the compared algorithms are indicated in bold in terms of f_{best}, f_{avg} and the success rate (SR).

N	BKR	f_{best}		f_{avg}		SR	
		BH	DNS	BH	DNS	BH	DNS
38	-173.9284	**-173.9284**	**-173.9284**	-173.8135	**-173.9014**	83/100	**96/100**
55	-279.2485	**-279.2485**	**-279.2485**	**-279.2485**	**-279.2485**	**100/100**	**100/100**
69	-359.8826	**-359.8826**	**-359.8826**	-359.8755	**-359.8799**	84/100	**96/100**
70	-366.8923	**-366.8923**	**-366.8923**	**-366.8923**	**-366.8923**	**100/100**	**100/100**
75	-397.4923	**-397.4923**	**-397.4923**	**-396.2922**	-396.2851	**1/100**	**1/100**
78	-414.7944	**-414.7944**	**-414.7944**	-414.7510	**-414.7837**	62/100	**90/100**
90	-492.4339	**-492.4339**	**-492.4339**	-491.6150	**-492.3017**	73/100	**93/100**
98	-543.6654	**-543.6654**	**-543.6654**	-542.4750	**-543.4987**	**1/100**	**1/100**
100	-557.0398	**-557.0398**	**-557.0398**	-555.3898	**-556.7677**	57/100	**85/100**
102	-569.3637	**-569.3637**	**-569.3637**	-568.3133	**-568.8893**	**16/100**	5/100
#Best		**10**	**10**	3	**9**	5	**9**



(a) LJ_{98} (b) LJ_{102}

Figure 9: Best solutions found by our DNS method for two hard instances.

6. Conclusions and future work

We propose a novel heuristic algorithm called the iterated dynamic neighborhood search (IDNS) algorithm for the well-known equal circle packing problem on a sphere (ECPOS), which has a number of significant applications in various domains. the ECPOS problem consists of packing N equal non-overlapping circles on a sphere such that the radius of circles is maximized and can be modeled as a non-convex constrained optimization problem. The proposed IDNS method relies on a spherical coordinate transformation of points in three-dimensional space that transforms the original constrained optimization problem into a series of unconstrained optimization subproblems, accompanied by a dynamic neighborhood search method to solve the unconstrained optimization subproblems, and a minimum distance adjustment method to adjust the minimum distance between N centers of spherical caps formed by circles.

The performance of the IDNS algorithm was assessed by conducting extensive experiments on 205 widely used instances with up to $N = 1080$. The experimental results showed that the IDNS algorithm was very effective and efficient compared with other methods for obtaining the best-known results in the literature. The algorithm improved the best-known results for 42 in-

stances and missed the best-known results for only 5 instances, while matching the best-known result for the remaining ones. On the other hand, the IDNS algorithm obtained a result inferior to the best-known result in the literature on several special instances for which construction methods based on prior knowledge of the problem hold the current best-known records. These outcomes indicate that our IDNS algorithm and the existing construction methods are complementary for solving the challenging Tammes problem.

The basic idea of the IDNS algorithm, i.e., transforming a constrained optimization problem defined on a curved surface into a series of unconstrained optimization subproblems and then solving them sequentially by the dynamic neighborhood search (DNS) method, is very general, and is applicable to a number of other constrained optimization problems such as the minimum energy configuration problems on the unit sphere. Moreover, the DNS method underlying the IDNS algorithm is also a general-purpose heuristic approach and can be applied to any unconstrained optimization problem with a first-order derivative such as sphere packing problems. In the future, we intend to further improve the DNS method by employing more effective perturbation strategies and local optimization methods. In addition, the multiple-stage local optimization method proposed in this study is very efficient to speed up the search process and can be applied to other related geometry optimization problems as well.

Acknowledgments

We are grateful to the reviewers for their valuable comments and suggestions which helped us to improve the paper. We also acknowledge the Beijing Beilong Super Cloud Computing Co., Ltd for providing HPC resources that have contributed to the computational results reported in this work (`http://www.blsc.cn/`). This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 61703213 and 61933005).

References

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

[2] Anjaly, P., Ratnoo, A., 2018. Tammes problem inspired multi-agent formation control over a manifold, in: 2018 AIAA Information Systems-AIAA Infotech@ Aerospace, p. 0898.

[3] Appelbaum, J., Weiss, Y., 1999. The packing of circles on a hemisphere. Measurement Science and Technology 10, 1015.

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

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

[6] Cheng, L., Feng, Y., Yang, J., Yang, J., 2009. Funnel hopping: Searching the cluster potential energy surface over the funnels. The Journal of Chemical Physics 130, 214112.

[7] Clare, B., Kepert, D., 1986. The closest packing of equal circles on a sphere. Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences 405, 329–344.

[8] Danzer, L., 1986. Finite point-sets on S^2 with minimum distance as large as possible. Discrete Mathematics 60, 3–66.

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

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

[11] Fowler, P.W., Tarnai, T., Kabai, S., 2005. Packing regular triplets of circles on a sphere. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 461, 2355–2367.

[12] Gáspár, Z., 1989. Some new multi-symmetric packings of equal circles on a 2-sphere. Acta Crystallographica Section B: Structural Science 45, 452–453.

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

[14] Hardin, R.H., Sloane, N.J.A., Smith, W.D., 2012. Tables of spherical codes with icosahedral symmetry. http://neilsloane.com/icosahedral.codes/ .

[15] Huang, H.X., Pardalos, P.M., Shen, Z.J., 2001. A point balance algorithm for the spherical code problem. Journal of Global Optimization 19, 329–344.

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

[17] Kottwitz, D.A., 1991. The densest packing of equal circles on a sphere. Acta Crystallographica Section A: Foundations of Crystallography 47, 158–165.

[18] L Fejes, T., 1943. U¨ ber die absca¨ tzung des ku¨ rzesten abstandes zweier punkte eines auf einer kugelfla¨ che liegenden punktsystemes. Jber. Deut. Math. Verein. 53, 66–68.

[19] Lai, X., Hao, J.K., Yue, D., Lü, Z., Fu, Z.H., 2022. Iterated dynamic thresholding search for packing equal circles into a circular container. European Journal of Operational Research 299, 137–153.

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

[21] Lipschütz, H., Skrodzki, M., Reitebuch, U., Polthier, K., 2021. Single-sized spheres on surfaces (S4). Computer Aided Geometric Design 85, 101971.

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

[23] Locatelli, M., Schoen, F., 2003. Efficient algorithms for large scale global optimization: Lennard-Jones clusters. Computational Optimization and Applications 26, 173–190.

[24] 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.

[25] 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.

[26] Mackay, A., Finney, J., Gotoh, K., 1977. The closest packing of equal spheres on a spherical surface. Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography 33, 98–100.

[27] Melnyk, T.W., Knop, O., Smith, W.R., 1977. Extremal arrangements of points and unit charges on a sphere: equilibrium configurations revisited. Canadian Journal of Chemistry 55, 1745–1761.

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

[29] Musin, O.R., Tarasov, A.S., 2012. The strong thirteen spheres problem. Discrete & Computational Geometry 48, 128–141.

[30] Musin, O.R., Tarasov, A.S., 2015. The Tammes problem for N = 14. Experimental Mathematics 24, 460–468.

[31] Newton, P.K., Sakajo, T., 2011. Point vortex equilibria and optimal packings of circles on a sphere. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 467, 1468–1490.

[32] Robinson, R.M., 1961. Arrangement of 24 points on a sphere. Mathematische Annalen 144, 17–48.

[33] Saff, E.B., Kuijlaars, A.B., 1997. Distributing many points on a sphere. The Mathematical Intelligencer 19, 5–11.

[34] Sloane, N.J.A., Hardin, R.H., Smith, W.D., 2022. Tables of spherical codes. http://neilsloane.com/packings/index.html .

[35] Specht, E., 2013. High density packings of equal circles in rectangles with variable aspect ratio. Computers & operations research 40, 58–69.

[36] Szabó, P.G., Markót, M.C., Csendes, T., Specht, E., Casado, L.G., García, I., 2007. New approaches to circle packing in a square: with program codes. volume 6 of *Optimization and Its Applications*. Springer Science & Business Media.

[37] Tammes, P.M.L., 1930. On the origin of number and arrangement of the places of exit on the surface of pollen-grains. Recueil des travaux botaniques néerlandais 27, 1–84.

[38] Tarnai, T., 2002. Polymorphism in multisymmetric close packings of equal spheres on a spherical surface. Structural Chemistry 13, 289–295.

[39] Tarnai, T., Fowler, P., 2007. Recent results in constrained packing of equal circles on a sphere. MATCH Communications in Mathematical and in Computer Chemistry 58, 461–479.

[40] Tarnai, T., Fowler, P., Kabai, S., 2003. Packing of regular tetrahedral quartets of circles on a sphere. Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences 459, 2847–2859.

[41] Tarnai, T., Gáspár, Z., 1983. Improved packing of equal circles on a sphere and rigidity of its graph, in: Mathematical Proceedings of the Cambridge Philosophical Society, Cambridge University Press. pp. 191–218.

[42] Tarnai, T., Gáspár, Z., 1987. Multi-symmetric close packings of equal spheres on the spherical surface. Acta Crystallographica Section A: Foundations of Crystallography 43, 612–616.

[43] Teshima, Y., Ogawa, T., 2000. Dense packing of equal circles on a sphere by the Minimum-Zenith method: symmetrical arrangement. Forma 15, 339–428.

[44] Tibor, T., 1984. Spherical circle-packing in nature, practice and theory. Structural Topology 9, 39–58.

[45] Wales, D.J., Doye, J.P., 1997. Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms. The Journal of Physical Chemistry A 101, 5111–5116.

[46] Wales, D.J., McKay, H., Altschuler, E.L., 2009. Defect motifs for spherical topologies. Physical Review B 79, 224115.

[47] Wales, D.J., Ulker, S., 2006. Structure and dynamics of spherical crystals characterized for the Thomson problem. Physical Review B 74, 212101.

[48] Wang, Z., Xiang, C., Zou, W., Xu, C., 2020. MMA regularization: Decorrelating weights of neural networks by maximizing the minimal angles. Advances in Neural Information Processing Systems 33, 19099–19110.

[49] Yuan, Y., Tole, K., Ni, F., He, K., Xiong, Z., Liu, J., 2022. Adaptive simulated annealing with greedy search for the circle bin packing problem. Computers & Operations Research , 105826.

[50] Zeng, Z.Z., Yu, X.G., Chen, M., Liu, Y.Y., 2018. A memetic algorithm to pack unequal circles into a square. Computers & Operations Research 92, 47–55.