Submitted to *INFORMS Journal on Computing*
manuscript JOC-2023-03-OA-0089

An efficient optimization model and tabu search-based global optimization approach for continuous p-dispersion problem

Xiangjing Lai
Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, P.R. China, laixiangjing@gmail.com

Zhenheng Lin
Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, P.R. China, linzhenheng@outlook.com

Jin-Kao Hao* (Corresponding author)
LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France, jin-kao.hao@univ-angers.fr

Qinghua Wu* (Corresponding author)
School of Management, Huazhong University of Science and Technology, 430074 Wuhan, P.R.China, qinghuawu@hust.edu.cn

https://doi.org/10.1287/ijoc.2023.0089

Continuous p-dispersion problems with and without boundary constraints are NP-hard optimization problems with numerous real-world applications, notably in facility location and circle packing, which are widely studied in mathematics and operations research. In this work, we concentrate on general cases with a non-convex multiply-connected region that are rarely studied in the literature due to their intractability and the absence of an efficient optimization model. Using the penalty function approach, we design a unified and almost everywhere differentiable optimization model for these complex problems and propose a tabu search-based global optimization (TSGO) algorithm for solving them. Computational results over a variety of benchmark instances show that the proposed model works very well, allowing popular local optimization methods (e.g., the quasi-Newton methods and the conjugate gradient methods) to reach high-precision solutions due to the differentiability of the model. These results further demonstrate that the proposed TSGO algorithm is very efficient and significantly outperforms several popular global optimization algorithms in the literature, improving the best-known solutions for several existing instances in a short computational time. Experimental analyses are conducted to show the influence of several key ingredients of the algorithm on computational performance.

Key words: Circle packing, continuous dispersion problem, global optimization, tabu search, nonlinear optimization.

1. Introduction

Equal circle packing and point arrangement are two important and closely related global optimization problems in mathematics and operations research. The equal circle packing problem consists of packing a fixed number p of non-overlapping congruent circles into a bounded region to maximize the common radius of the circles (Addis et al. 2008, López and Beasley 2011, Melissen 1993, Szabó et al. 2007, Weaire and Aste 2008). As a general model, the problem has a variety of practical applications ranging from circular cutting to container loading (Castillo et al. 2008). By contrast, the point arrangement problem aims to scatter p given points in a bounded region to maxmize the minimum distance between the points (Akiyama et al. 2002, Costa 2013, Goldberg 1970, Melissen 1993, Schwartz 1970). This problem finds application in areas such as facility location and design of computer experiments (Dimnaku et al. 2005, Drezner et al. 2019). For example, in the facility location problem (Drezner et al. 2019, Drezner and Erkut 1995, López-Sánchez et al. 2021), a facility is represented by a dispersion point and the goal is to place p obnoxious facilities (e.g., nuclear facilities) in locations to maximize the minimum distance between them. These two problems can be mutually converted into each other if the given bounded region is a regular convex set (e.g., a convex polygon, a circle or the surface S^2 of a sphere), and should be treated separately otherwise.

The equal circle packing and point arrangement problems are also known under the general name of the continuous p-dispersion problem (CpDP) (Drezner and Erkut 1995, Baur and Fekete 2001, Dimnaku et al. 2005, Chen et al. 2019, Dai et al. 2021) and can be described as follows. Given a positive integer p, a bounded region Ω composed of a container C containing K ($K \geq 0$) holes $\{U_1, U_2, \ldots, U_K\}$ (i.e., $\Omega = C \setminus \cup_{k=1}^{K} U_k$), the CpDP is to place p given dispersion points $\{c_1, c_2, \ldots, c_p\}$ in Ω so that the minimum distance from a dispersion point to the boundaries of the container and holes is no less than D_b, and the minimum distance D between the dispersion points is maximized, where D_b can be a non-negative constant or an increasing function with respect to D. Formally, the CpDP can be expressed as:

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

$$\text{Subject to} \quad d(c_i, c_j) \geq D, 1 \leq i \neq j \leq p \tag{2}$$

$$d(c_i, \partial C) \geq D_b, i = 1, 2, \ldots, p \tag{3}$$

$$d(c_i, \partial U_k) \geq D_b, i = 1, 2, \ldots, p, k = 1, 2, \ldots, K \tag{4}$$

$$c_i \in C \setminus \cup_{k=1}^{K} U_k, i = 1, 2, \ldots, p \tag{5}$$

where D is the objective function value to be maximized and represents the minimum distance between points, $d(c_i, c_j)$ is the Euclidean distance between points c_i and c_j, ∂C and ∂U_k denote respectively the boundaries of C and U_k, $d(c_i, \partial C)$ is the Euclidean distance between c_i and ∂C defined as $min\{d(c_i, c) : c \in \partial C\}$. D_b denotes the allowed minimum distance between a dispersion point and the boundaries of the container and holes, and D_b can be a constant or an increasing function with respect to D, such as $D_b(D) = \lambda \times D$ with $\lambda \in [0, 0.5]$. The constraints (2) are called the separation constraints, the constraints (3) and (4) are called the boundary constraints, and the constraints (5) are called the containment constraints. Following Dai et al. (2021), the container and holes are represented by a polygon because a simply-connected region can be approximated by a convex or non-convex polygon with multiple edges. The CpDP is a difficult nonlinear non-convex optimization problem like those in (Bierlaire et al. 2010, Geißler et al. 2018, Ugray et al. 2007) and is computationally challenging especially for large-scale instances.

As mentioned above, depending on the setting of D_b the CpDP problem formulated by Eqs. (1–5) has two well-known variants (i.e., the CpDP problems with and without a boundary constraint), corresponding to the equal circle packing problem and the point arrangement problem, respectively. First, if D_b is $\frac{1}{2} \times D$, the corresponding problem is called the CpDP problem with a boundary constraint, where the dispersion points must be at least D_b $(= \frac{D}{2})$ away from the boundaries of the container and holes. Second, if D_b is set to 0, the corresponding problem is called the CpDP problem without a boundary constraint, where the dispersion points are allowed to approach the boundaries of the container and holes. One observes that the only difference between the equal circle packing and point arrangement problems lies in whether the dispersion points (or the centers of circles) are allowed to approach the boundaries of the container and holes.

Due to their NP-hard feature (Demaine et al. 2010) and potential applications, the equal circle packing and point assignment problems have been widely studied in the literature by researchers from mathematics, facility location theory, and the design of computer experiments. From the computational and theoretical perspectives, there are a host of

studies in the literature (e.g., see (Lai et al. 2022, Szabó et al. 2007)) for equal circle packing). We summarize the most representative studies as follows.

Melissen (1993) studied the equal circle packing problem and the point assignment problem in an equilateral triangle, while Graham et al. (1998) tackled the former problem in a circular region using a billiard simulation method. Birgin and Sobral (2008) proposed a twice-differentiable nonlinear optimization model for the circle and sphere packing problems in a simple convex container. From contrasting perspectives, Grosso et al. (2010) and Addis et al. (2008) respectively used a monotonic basin hopping (MBH) algorithm and a population-based basin-hopping (PBH) algorithm for the problem of packing equal circles into a square and circular container. Mladenović et al. (2005) and López and Beasley (2011) designed heuristic algorithms based on the formulation space search method for the equal circle packing problem in a simple convex container. Huang and Ye (2010) presented a greedy vacancy search algorithm for packing equal circles into a square container, while more recently, Stoyan et al. (2020) designed several heuristic strategies for the circle and sphere packing problems in a simple convex container. Very recently, Amore (2023) studied the equal circle packing problem in regular polygons via optimizing a related energy function with a heuristic algorithm. Complementing these studies, the popular Packomania website (Specht 2023) provides the best-known solutions collected from different researchers for the equal circle packing problems in a variety of simple convex containers. In the design of computer experiments, the point assignment problem was studied as the max-min distance design problem (Mu and Xiong 2017, Van Dam et al. 2007).

Apart from the studies involving simple convex containers, several studies tackle the equal circle packing problem in a non-convex simply-connected region. For example, by the movement of circles in a rotated container under shaking and gravity, Machchhar and Elber (2017) proposed a two-phase heuristic algorithm for finding the densest packing of congruent circles in an arbitrary non-convex container with B-spline boundaries. Via the discretization of solution space and the popular bottom-left placement strategy, Yuan et al. (2018) proposed an angle bisector heuristic algorithm for packing equal circles into a convex or concave polygonal container, reporting computational experiments that show their proposed algorithm can produce high-quality solutions for some small-scale instances. Recently, Dai et al. (2021) introduced a heuristic algorithm for the continuous p-dispersion problem in a simply-connected non-convex polygonal container. The idea of their algorithm

is to model the circles (or points) as physical bodies and induce the circles to move over time under repulsive forces between circles and between the circles and container boundaries. Computational results show that their algorithm significantly outperforms several previous algorithms including the angle bisector heuristic algorithm (Yuan et al. 2018), establishing this algorithm as the current state-of-the-art algorithm for the continuous p-dispersion problem in a non-convex polygonal container.

Our literature review reveals that a large number of previous studies for the continuous p-dispersion problem focus on the case where the region is simply connected (in particular, involving a regular convex region such as a circular or square container). However, many practical situations involve a non-convex multiply-connected region, and the literature does not provide suitable optimization models nor efficient global optimization algorithms for dealing with the corresponding continuous p-dispersion problems, a limitation that provides the main motivation of this study. It should be clear that the non-convexity and multiply-connected feature of the given region makes the problem much more difficult to handle compared to the popular models involving a regular convex region.

The main contributions of our work can be summarized as follows:

1. We extend the research of the continuous p-dispersion problem from a simply-connected region to non-convex multiply-connected regions in contrast to the focus of all previous studies on the simply-connected region case.

2. We introduce a unified optimization model for the continuous p-dispersion problems with and without a boundary constraint, providing a model that can take advantage of popular continuous solvers such as the limit memory BFGS method to perform local optimization, thus making it possible for the global algorithm to reach high-precision solutions.

3. We propose an effective global optimization algorithm called TSGO for the continuous p-dispersion problems both with and without a boundary constraint. The proposed algorithm is capable of addressing a variety of cases, including those with a convex, non-convex, simply-connected or a multiply-connected region. The source codes of the proposed algorithm are made publicly available for potential real-world applications (Lai et al. 2024).

4. We provide a set of benchmark instances with a non-convex multiply-connected region (Lai et al. 2024), which fill the gap in the literature devoted to such instances. We expect that these instances will be useful for the evaluation and comparison of algorithms in the future.

The rest of the paper is organized as follows. Section 2 presents a unified optimization model for the continuous p-dispersion problems in a non-convex multiply-connected region. Section 3 describes the proposed TSGO algorithm. Section 4 evaluates the performance of the TSGO algorithm. Section 5 presents an analysis to get insights into the performance of the algorithm. The last section summarizes the contributions of the present work and provides some perspectives for future studies.

2. A unified optimization model for the continuous p-dispersion problems

The goal of this section is to present a unified optimization model for the continuous p-dispersion problems with and without boundary constraints.

2.1. Basic idea underlying the optimization model

The continuous p-dispersion problems studied are constrained max-min optimization problems, which are usually difficult to handle using popular local optimization methods (e.g., the quasi-Newton method). We adopt a popular approach to solving the equal circle packing problems (Huang and Ye 2010, Lai et al. 2022, 2023a,b), whose central idea is to transform the max-min optimization problem into a series of unconstrained optimization subproblems in which the allowed minimum distance between points is fixed to a constant D, and then solve them in sequence by an unconstrained global optimization algorithm.

The differentiability of the objective function is a very important feature, which allows a global optimization algorithm to reach high-precision solutions by using popular gradient-based local optimization methods, such as the quasi-Newton and conjugate gradient methods. However, for the studied continuous p-dispersion problem, it is very challenging to create a differentiable optimization model due to the non-convexity and multiply-connected feature of the region Ω.

In view of the foregoing considerations, we use the idea of repulsions and attractions between the dispersion points and the boundaries of the container and holes to devise a unified unconstrained optimization model with an almost everywhere differentiable objective function for the continuous p-dispersion problems with and without boundary constraints, where the allowed minimum distance between dispersion points is fixed to a constant D. The following subsections describe this optimization model in detail.

2.2. Continuous p-dispersion problem with an allowed minimum distance D between points

The continuous p-dispersion problem with an allowed minimum distance D between points (CpDP-D for short) is a subproblem of the continuous p-dispersion problem studied in this work, whose goal is to find a feasible solution satisfying the constraints described by Eqs. (2)-(5). We convert this constraint satisfaction problem into an unconstrained optimization problem by introducing a penalty function approach as follows.

Given an allowed minimum distance D between dispersion points and a multiply-connected region Ω consisting of a polygonal container C, K $(K \geq 0)$ polygonal holes $\{U_1, U_2, \ldots, U_K\}$, and a candidate solution $X = (x_1, y_1, \ldots, x_p, y_p)$ (i.e., Cartesian coordinates of p points in Ω), the quality of X is measured with the following objective function:

$$E_D(X) = \sum_{i=1}^{p-1} \sum_{j=i+1}^{p} l_{ij}^2 + \alpha \times (\sum_{i=1}^{p} O_{i0} + \sum_{i=1}^{p} \sum_{k=1}^{K} O_{ik}) \tag{6}$$

with

$$l_{ij} = max\{0, D - \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\} \tag{7}$$

where α is a penalty factor and l_{ij}^2 represents the degree of constraint violation of the distance between two dispersion points c_i and c_j, and the goal of including l_{ij}^2 is to force the dispersion points c_i and c_j to be separated by a distance of at least D. Equivalently, l_{ij} is the overlap depth between two equal circles with a radius of $\frac{D}{2}$, located respectively at the points c_i and c_j (see Fig. 1 for an illustration). Additionally, O_{i0} is the degree of violation of the containment constraint of point c_i and the distance constraint between c_i and the boundary of container C, and O_{ik} is the degree of constraint violation of the distance between c_i and the k-th hole U_k. Thus, $E_D(X)$ measures the total degree of constraint violation of the given solution X, and $E_D(X) = 0$ means that X is a feasible solution, i.e., all p points are contained in the container C, the minimum distance between the dispersion points is no less than the given D value, and the minimum distance from a dispersion point c_i $(1 \leq i \leq N)$ to all boundaries of the container and holes is no less than D_b.

O_{i0} and O_{ik} are two key components of our optimization model that are more complicated to understand and formulate than l_{ij} due to the non-convexity of the container C and the hole U_k. Therefore, we focus on these components in the following two subsections.

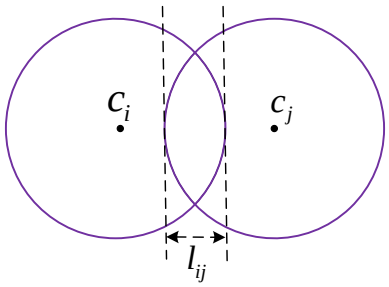


Figure 1 **Overlap depth (l_{ij}) between two equal circles c_i and c_j, where the radius of circles is $\frac{D}{2}$.**

2.2.1. Penalty term O_{i0} between a dispersion point and the polygonal container. The penalty term O_{i0} aims to force the dispersion point c_i into the container and keep it at least a distance of D_b from the container boundary via repulsion and attraction between the dispersion point and this boundary. To formulate the penalty O_{i0}, we first give the following two definitions.

- **Definition 1: Active edges of polygonal container**. Given a polygonal container $C = (V_0, E_0)$ and a dispersion point c_i with coordinates (x_i, y_i), where V_0 is the set of vertices of C and E_0 is the set of edges on the boundary of C, an edge e_l ($\in E_0$) is called an *active edge* with respect to the given point c_i if it satisfies the following condition.

Moving along the container boundary in a counter-clockwise direction, the current edge e_l ($1 \leq l \leq |E_0|$) is identified as an active edge in the following two cases: (1) the given point c_i is contained in C and lies on its left-hand side (i.e., using the left-hand rule), (2) the given point c_i is not contained in C and lies on its right-hand side (i.e., using the right-hand rule).

To illustrate, all active container edges are indicated by a solid line segment in Figs. 2 and 3. Clearly, the answer to the question of whether an edge of the polygonal container is active or not depends on the current position of the dispersion point c_i.

- **Definition 2: Active foot point on the container boundary**. Given a dispersion point c_i with coordinates (x_i, y_i) and active edge e on the container boundary, if the foot h of the perpendicular from c_i to the line containing edge e lies on this edge, then the point h is called an *active foot point* with respect to the given dispersion point c_i, and a non-active foot point otherwise. The set of all active foot points with respect to the given dispersion point c_i and the container C is denoted by H_0^i.

Figs. 2 and 3 provide some representative foot points, including active foot points and non-active foot points. Taking the subfigure (a) of Fig. 2 as an example, the points h_1 and

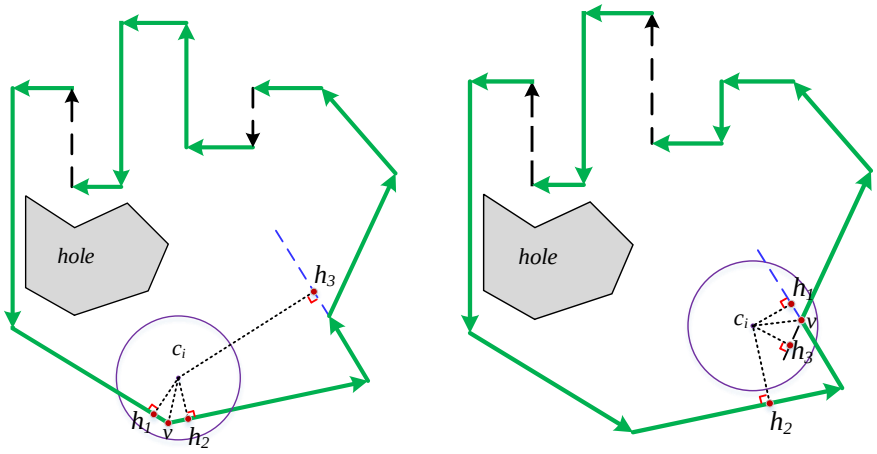
h_2 are active foot points, and h_3 is a non-active foot point since it lies on the extending line of an edge.

Using the set H_0^i of active foot points, we formulate the penalty term O_{i0} as follows.

$$O_{i0} = \begin{cases} \gamma \times (D_b + min\{d(c_i, v) : v \in V_0 \cup H_0^i\})^2, & c_i \notin C \\ \sum_{v \in V_0 \cup H_0^i} (max\{0, D_b - d(c_i, v)\})^2, & c_i \in C \end{cases} \tag{8}$$

where γ is a penalty factor, $d(c_i, v)$ denotes the Euclidean distance between the point c_i and point $v \in V_0 \cup H_0^i$, V_0 denotes the set of vertices of the polygonal container, and D_b is the allowed minimum distance from the dispersion point c_i to the container boundaries and can be written as $D_b = \lambda \times D$ with $\lambda \in [0, 0.5]$, where λ is a predetermined coefficient used to control the value of D_b.

One can observe from Eq. (8) that the value of O_{i0} is calculated according to whether or not the point c_i is contained in the container C. The fundamental principles behind the equation can be explained as follows:

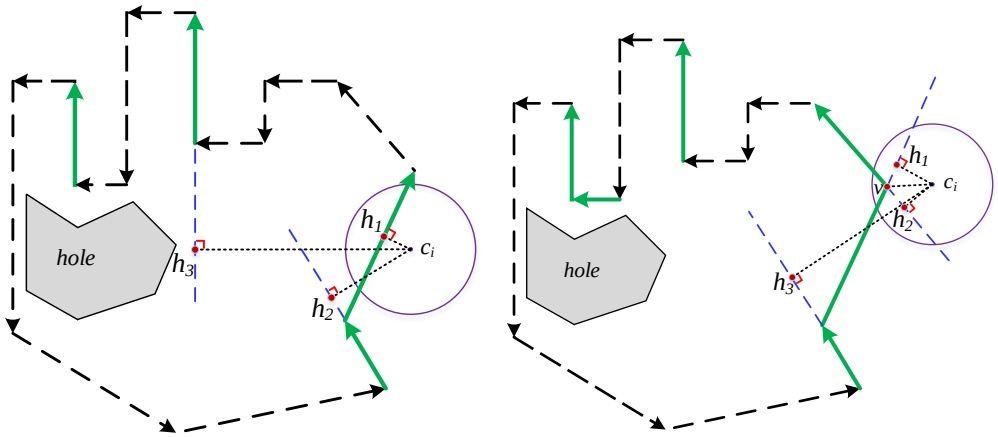


(a) $H_0^i \cap B(c_i, D_b) \neq \emptyset$ (b) $H_0^i \cap B(c_i, D_b) = \emptyset$

Figure 2 **The active edges and some representative foot points for the case that the dispersion point c_i lies in the container, where the active edges of container are indicated by a solid line segment and the circle is $B(c_i, D_b)$.**

First, when the dispersion point c_i with coordinates (x_i, y_i) is contained in C (i.e., $c_i \in C$), we need to identify the degree of constraint violation of the distance between the point c_i and the container boundary to keep the point c_i at least a distance of D_b from the container boundary, which requires consideration of two situations. In the first situation, there exist

one or several active foot points in the vicinity $B(c_i, D_b)$ of c_i (i.e., $H_0^i \cap B(c_i, D_b) \neq \emptyset$), where $B(c_i, D_b) = \{c : ||c_i - c|| \leq D_b\}$. In this case, all points in $(V_0 \cup H_0^i) \cap B(c_i, D_b)$ are used to generate the penalty term O_{i0}, thus producing a repulsive force $(-\frac{\partial O_{i0}}{\partial x_i}, -\frac{\partial O_{i0}}{\partial y_i})$ to the dispersion point c_i, driving it to move a distance of at least D_b away from the boundary. Fig. 2(a) gives a representative example for this situation. On the contrary, if there does not exist any active foot point in the vicinity of c_i (i.e., $H_0^i \cap B(c_i, D_b) = \emptyset$), all vertices of container C in the range of $B(c_i, D_b)$ are used to generate O_{i0} and thus produce a repulsive force to c_i. Otherwise, it is likely that the dispersion point c_i leaves the container by passing beyond one of its vertices during the optimization process due to the repulsive forces from other dispersion points, which is one of main reasons why the set V_0 of container vertices is involved in O_{i0}. Fig. 2(b) gives an example of this situation. Formally, O_{i0} can be written as $O_{i0} = \sum_{v \in V_0 \cup H_0^i} (max\{0, D_b - d(c_i, v)\})^2$ for these two situations.



(a) $min_{v \in H_0^i}\{d(c_i, v)\} < min_{v \in V_0}\{d(c_i, v)\}$ (b) $min_{v \in H_0^i}\{d(c_i, v)\} \geq min_{v \in V_0}\{d(c_i, v)\}$

Figure 3 **The active edges and some representative foot points for the case that the dispersion point c_i lies outside the container, where the active edges of container are indicated by a solid line segment and the circle is $B(c_i, D_b)$.**

Second, when the point c_i lies outside the container C (i.e., $c_i \notin C$), two situations need to be considered to handle the containment constraint of point c_i, to force the point c_i to enter the container C. In the first situation, the minimum distance between c_i and the set H_0^i is smaller than the minimum distance between c_i and the set V_0 (i.e., $min_{v \in H_0^i}\{d(c_i, v)\} < min_{v \in V_0}\{d(c_i, v)\}$). Then the foot point $v \in H_0^i$ closest to c_i is used to generate the penalty term O_{i0} and thus produce an attractive force to induce c_i to gradually approach the

boundary of container. Fig. 3(a) provides an example for this situation. In the second situation, $min_{v\in H_0^i}\{d(c_i,v)\} \geq min_{v\in V_0}\{d(c_i,v)\}$, and then the vertex v of V_0 closest to c_i is used to generate the penalty term O_{i0} and thus produces an attractive force $(-\frac{\partial O_{i0}}{\partial x_i}, -\frac{\partial O_{i0}}{\partial y_i})$ between the points v and c_i. Without these adjustments, the dispersion point c_i will always lie outside of container during the optimization due to the lack of an attractive force, which is another reason why the set V_0 of vertices is involved in O_{i0}. Fig. 3(b) provides an example for this situation. In summary, when $c_i \notin C$ holds, the point in $V_0 \cup H_0^i$ closest to c_i is used to generate O_{i0}. Formally, the penalty term O_{i0} can be uniformly written as $O_{i0} = \gamma \times (D_b + min\{d(c_i,v) : v \in V_0 \cup H_0^i\})^2$ for these two situations.

Thus, the inclusion of O_{i0} in $E_D(X)$ ensures that the dispersion point c_i is contained in the container and the minimum distance between c_i and boundary of container is no less than D_b for any candidate solution X with $E_D(X) = 0$.

2.2.2. Penalty term O_{ik} between a dispersion point and a polygonal hole. The penalty term O_{ik} aims to force the dispersion point c_i to leave from the hole U_k and keep at least a distance D_b from the hole boundary. To formulate O_{ik}, we first give two definitions.

• **Definition 3: Active edges of a polygonal hole**. Given a polygonal hole $U_k = (V_k, E_k)$ $(k \geq 1)$ and a dispersion point c_i with coordinates (x_i, y_i), an edge e $(\in E_k)$ is called an *active edge* of U_k with respect to c_i according to two situations. Specifically, when moving along the boundary of hole U_k in a counter-clockwise direction, the edge e is identified as an active edge of U_k if (1) the given point c_i is contained in U_k and lies on its right-hand side (i.e., using the right-hand rule), or (2) the given point c_i is not contained in U_k and lies on its left-hand side (i.e., using the left-hand rule).

For example, all active edges of a polygonal hole are indicated by a solid line segment in Figs. 4 and 5. Clearly, the answer to the question of whether an edge of a polygonal hole is active or not depends on the current position of the dispersion point c_i.

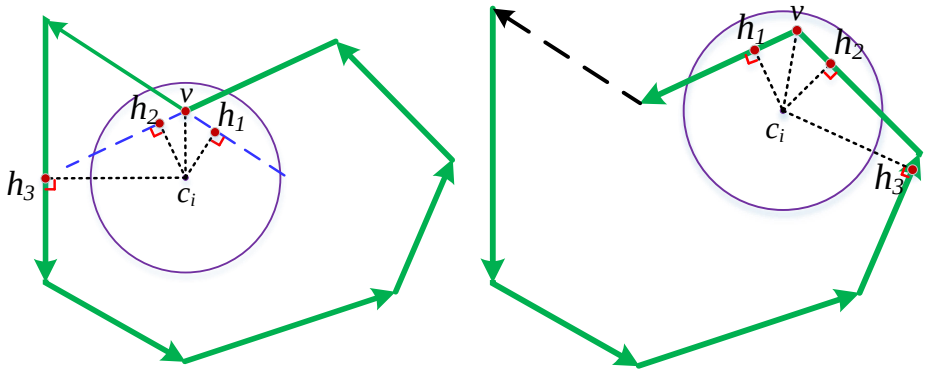
• **Definition 4: Active foot point on the boundary of a polygonal hole**. Given a dispersion point c_i with coordinates (x_i, y_i) and an active edge e of the hole U_k, if the foot h of perpendicular from c_i to the line containing the edge e lies on e, then h is called an *active foot point* of U_k with respect to the dispersion point c_i. The set of all active foot points of U_k with respect to the given dispersion point c_i is denoted by H_k^i.

Figs. 4 and 5 give some representative foot points, including active foot points and non-active foot points. In Fig. 4(a), the point h_3 is an active foot point, and the points h_1 and h_2 are non-active foot points since they lie on the extended line of the corresponding edge.

Using the set H_k^i of active foot points, the penalty term O_{ik} can be formulated as follows:

$$O_{ik} = \begin{cases} \gamma \times (D_b + min\{d(c_i, v) : v \in V_k \cup H_k^i\})^2, & c_i \in U_k \\ \sum_{v \in V_k \cup H_k^i} (max\{0, D_b - d(c_i, v)\})^2, & c_i \notin U_k \end{cases} \tag{9}$$

where γ is a penalty factor, $d(c_i, v)$ represents the distance between point c_i and v, V_k is the set of vertices of U_k, H_k^i is the set of active foot points of U_k with respect to the given dispersion point c_i, and D_b represents the allowed minimum distance from c_i to the boundaries of holes and container.

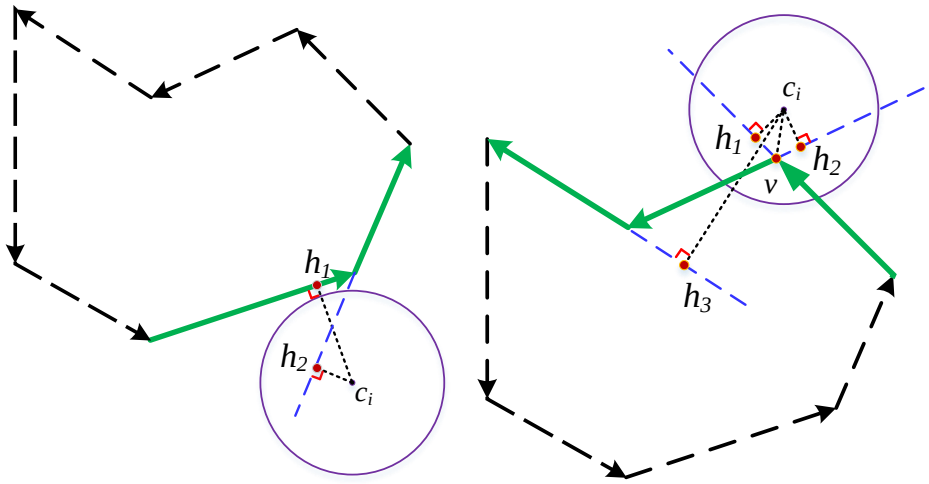


(a) $min_{v \in V_k}\{d(c_i, v)\} < min_{v \in H_k^i}\{d(c_i, v)\}$ (b) $min_{v \in V_k}\{d(c_i, v)\} \geq min_{v \in H_k^i}\{d(c_i, v)\}$

Figure 4 **The active edges and some representative foot points for the case that the dispersion point c_i lies in the hole U_k, where the active edges are indicated by a solid line segment and the circle is $B(c_i, D_b)$.**

Equation (9) indicates that the penalty term O_{ik} is calculated depending on whether the dispersion point c_i lies in the hole U_k or not. First, if $c_i \in U_k$, the point in $V_k \cup H_k^i$ closest to c_i is used to generate O_{ik} and the dispersion point c_i will receive an attractive force $(-\frac{\partial O_{ik}}{\partial x_i}, -\frac{\partial O_{ik}}{\partial y_i})$ from this point, thus guiding it to leave the hole U_k. Thus, O_{ik} can be written as $\gamma \times (D_b + min\{d(c_i, v) : v \in V_k \cup H_k^i\})^2$. Fig. 4 gives two examples for the following two situations: (1) the closest point belongs to V_k (subfigure (a)), (2) the closest point belongs to H_k^i (subfigure (b)).

Second, if the dispersion point c_i lies on the outside of hole U_k (i.e., $c_i \notin U_k$), all the points in $(V_k \cup H_k^i) \cap B(c_i, D_b)$ will provide a positive contribution to the penalty term O_{ik} and thus produce a repulsive resultant force $(-\frac{\partial O_{ik}}{\partial x_i}, -\frac{\partial O_{ik}}{\partial y_i})$ on the dispersion point c_i, making



(a) $H_k^i \cap B(c_i, D_b) \neq \emptyset$ (b) $H_k^i \cap B(c_i, D_b) = \emptyset$

Figure 5 **The active edges and some representative foot points for the case that the dispersion point c_i lies outside the hole $hole_k$, where the active edges are indicated by a solid line segment and the circle is $B(c_i, D_b)$.**

it away from the hole boundary by a distance of at least D_b. If there is no active foot point of U_k in the vicinity $B(c_i, D_b)$ of c_i, then the vertices of the hole U_k in $B(c_i, D_b)$ are used to generate the penalty term O_{ik}. An example of this situation can be found in subfigure (b) of Fig. 5. Thus, O_{ik} can be written as $O_{ik} = \sum_{v \in V_k \cup H_k^i} (max\{0, D_b - d(c_i, v)\})^2$.

It is worth noting that, according to Eqs. (8) and (9), the point-in-polygon test (Wang et al. 2005) is a basic subroutine that will be frequently called during the optimization process for any algorithm employing this optimization model.

2.3. The parameters and discussion of the proposed optimization model

The proposed optimization model involves three parameters α, γ, and λ, where α is used in Eq. (6), and γ is used in Eqs. (8) and (9) to quantify the penalty degree of constraint violation. Parameter λ is used to control the value of D_b as $D_b = \lambda \times D$, where D is the allowed minimum distance between dispersion points and D_b is the allowed minimum distance between a dispersion point and the boundaries of the container and holes, where a larger λ value means a larger distance constraint, and vice versa.

The optimization model in Eq. (6) is uniformly used to formulate the continuous p-dispersion problems with and without a boundary constraint. To distinguish these two problems, we use respectively different parameter settings as follows.

- For the continuous p-dispersion problem with a boundary constraint (i.e., the equal circle packing problem), the parameters α, γ, and λ are respectively set to 1.0, 2.0 and 0.5, and the resulting model is equivalent to the equal circle packing problem.
- For the continuous p-dispersion problem without a boundary constraint (i.e., the point arrangement problem), the parameters α, γ, and λ are respectively set to 3000, 1.0 and 10^{-3}. We set λ to a very small number (i.e., 10^{-3}), instead of 0, as our preliminary experiments disclosed that $\lambda = 0$ results in an ill-conditioned objective function for the problems with a non-convex region, significantly increasing the difficulty of local optimization.

One notices that the proposed optimization model is differentiable except for the points on the boundaries of the container and holes, making it possible to apply popular continuous solvers (e.g., the quasi-Newton method) as a local optimization method to reach high-precision solutions of the studied problem. Until now, such an optimization model has been missing for the continuous p-dispersion problems with a non-convex multiply-connected region, which explains why existing algorithms in the literature fail to obtain high-precision solutions for these cases. The proposed unified optimization model fills this gap and can be considered as one of the main innovations of the present work.

3. Tabu Search based Global Optimization for the continuous p-dispersion problems

Our TSGO algorithm is designed to solve the optimization model described in Section 2 for the continuous p-dispersion problems with and without boundary constraints. The basic idea of the algorithm is to dynamically convert a constrained optimization problem into a series of unconstrained optimization subproblems and then solve them by a tabu search method operating in the continuous solution space.

3.1. Main framework of the TSGO algorithm

The TSGO algorithm consists of three main components: a solution initialization procedure, a tabu search method to improve the solution, and a distance adjustment method to maximize the minimum distance between dispersion points while maintaining the feasibility of the solution. The pseudo-code of the algorithm is described in Algorithm 1, where X represents the current solution, X^* and D^* respectively denote the best solution found so far and the corresponding minimum distance between the dispersion points.

The algorithm works as follows. First, it calculates the area of the bounded region Ω (line 1), which is the difference between the area of the container and the total area of

Algorithm 1: Main framework of the proposed TSGO algorithm

Input: Number of points to be packed (p), container C and holes U_k ($k=1,2,\ldots,K$), maximum time limit (t_{max}), packing density of the initial solution (ρ)

Output: The best configuration found (X^*) and the minimum distance between points (D^*)

1 $Area \leftarrow CalculateArea(C,U_1,U_2,\ldots,U_K)$ /* Calculate the area of bounded region Ω as $Area = Area(C) - \sum_{k=1}^{K} Area(U_k)$ */

2 $D \leftarrow 2 \times \sqrt{\frac{\rho \times Area}{N \times \pi}}$ /* Calculate the initial value of D according to input density ρ */

3 $X \leftarrow RandomSolution(p)$ /* Generate randomly an initial solution X */

4 $X \leftarrow TabuSearch(E_D(X), X, D)$ /* Minimize the function $E_D(X)$ by Algorithm 2 */

5 $(X,D) \leftarrow AdjustDistance(X,D)$ /* Adjust configuration (X,D) by Algorithm 3 */

6 $D^* \leftarrow D$, $X^* \leftarrow X$

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

8 $D \leftarrow D^*$ /* Set D to the best value found so far */

9 $X \leftarrow RandomSolution(p)$

10 $(X,D) \leftarrow TabuSearch(E_D(X), X, D)$

11 /* $E_D(X) < 10^{-25}$ means that X is a feasible solution for the current D */

12 **if** $E_D(X) < 10^{-25}$ **then**

13 $(X,D) \leftarrow AdjustDistance(X,D)$

14 **if** $D > D^*$ **then**

15 $D^* \leftarrow D$

16 $X^* \leftarrow X$ /* Save the best solution found */

17 **end**

18 **end**

19 **end**

20 **return** (X^*, D^*)

the K holes. Then, assuming that each dispersion point is a circle with a radius of $\frac{D}{2}$ and that the preestimated packing density of these p circles in the region Ω is an input value ρ, the algorithm calculates $D = 2 \times \sqrt{\frac{\rho \times Area}{N \times \pi}}$, which is the initial estimation of the minimum distance between the p points (line 2).

After that, the algorithm generates an initial solution by distributing uniformly and randomly the p dispersion points in a minimum rectangle containing the region Ω and then employs the tabu search method to improve the quality of solution by minimizing the function $E_D(X)$ (lines 3-4). The improved solution from tabu search is slightly adjusted to maximize the minimum distance D between p points, maintaining the feasibility of the solution (line 5), and then the resulting solution is saved as X^*.

Then, the algorithm enters a 'while' loop to iterate until the time limit (t_{max}) is reached and finally returns the best solution X^* found (lines 7-19). At each iteration, based on the best value of D found so far, an initial solution is randomly generated and the tabu search method is used to improve its quality by minimizing $E_D(X)$ (lines 8-10). The distance adjustment procedure ($AdjustDistance(\cdot)$) is used to maximize the minimum distance D between points once a feasible solution is obtained by the tabu search method (line 13). Moreover, the value of D^* is updated each time an improving value is found (lines 14-17).

3.2. Tabu search method

Tabu search (TS) (Glover and Laguna 1998) is a popular metaheuristic for combinatorial optimization that has also been adapted to solve continuous global optimization problems (Chelouah and Siarry 2000). The present TS method exploits the solution space utilizing a neighborhood composed of nearby local minima. The general procedure, neighborhood structure and tabu list management strategy of our tabu search method are described in the following subsections.

3.2.1. Tabu search for the continuous p-dispersion problems. The TS method of our TSGO algorithm is depicted in Algorithm 2, where X and X^b respectively represent the current solution and the best solution found so far, and $X^{best}_{neighbor}$ represents the best individual in the current neighborhood $N_{ins}(X)$ in terms of the objective value. Starting from an input solution X_0 and the initialization of the tabu list T (lines 2-5), the TS method performs successive iterations until a feasible solution is found or the best solution X^b cannot be improved during the last β_{max} consecutive iterations (lines 6-31), where β_{max} is a parameter called the search depth. At each iteration, the candidate set P of high-energy dispersion points and the candidate set V of low-energy vacancy sites are first constructed for the current solution X (lines 7-8). Then, the vacancy-based insertion neighborhood $N_{ins}(X)$ of the current solution with respect to P and V is evaluated (lines 10-21) and a best non-tabu solution in $N_{ins}(X)$ is selected to replace the current solution X (line 22). After that, the tabu list T is updated and the current solution X is further improved by a very short run of the monotonic basin-hopping (MBH) method, which is proposed in (Leary 2000) for an intensified search (lines 23-24).

The MBH method is a simple iterated improvement procedure, which repeats a perturbation operator followed by a local optimization until the current solution cannot be

further improved for θ_{max} iterations, where θ_{max} is a parameter called its search depth. At each iteration of the MBH method, the current solution is first perturbed by shifting each solution coordinate of the solution in a given interval $[-\Delta, \Delta]$, where $\Delta = 0.4 \times D$, and then is locally improved by the L-BFGS method (Liu and Nocedal 1989) equipped with an efficient line search method (Hager and Zhang 2005). The new solution is accepted as the current solution if and only if it is superior to the current solution.

Algorithm 2: Tabu Search method for the continuous p-dispersion problems

Function *TabuSearch*
Input: Input solution X_0, depth of tabu search (β_{max}), parameter Q.
Output: The best solution found (X^b)
$X \leftarrow Local\_Optimization(X_0)$ `/* Minimize locally function` $E_D(\cdot)$ `starting from` X `*/`
$X^b \leftarrow X$ `/*` X^b `denotes the best solution found by the current tabu search */`
$NoImprove \leftarrow 0$
Initialize tabu list T
while $(NoImprove \leq \beta_{max}) \wedge (E_D(X^b) > 10^{-25})$ **do**
  Select Q dispersion points (i.e., $P[1:Q]$) with the highest energies from the current solution X
  Find Q vacancy sites (i.e., $V[1:Q]$)) with the lowest energies in the current solution X
  `/* Evaluate the insertion neighborhood` $N_{ins}(X)$ `*/`
  $i_1 \leftarrow rand(Q)$, $j_1 \leftarrow rand(Q)$ `/*` $rand(Q)$ `denotes a random integer in` $[1, Q]$ `*/`
  $X^{best}_{neighbor} \leftarrow X \oplus Move(P[i_1], V[j_1])$ `/*` $X^{best}_{neighbor}$ `denotes the best solution in` $N_{ins}(X)$ `*/`
  for $i \leftarrow 1$ ***to*** Q **do**
    for $j \leftarrow 1$ ***to*** Q **do**
      $X_{neighbor} \leftarrow X \oplus Move(P[i], V[j])$
      $X_{neighbor} \leftarrow Local\_Optimization(X_{neighbor})$ `/* Minimize` $E_D(X)$ `by L-BFGS */`
      if $Move(P[i], V[j])$ *is not forbidden* $\wedge\ E_D(X_{neighbor}) < E_D(X^{best}_{neighbor})$ **then**
        $X^{best}_{neighbor} \leftarrow X_{neighbor}$
        $I \leftarrow P[i]$
      end
    end
  end
  $X \leftarrow X^{best}_{neighbor}$ `/* Update the current solution */`
  Update tabu list T by I
  $X \leftarrow MBH(X)$ `/* Improve` X `by a short run of MBH */`
  if $E_D(X) < E_D(X^b)$ **then**
    $X^b \leftarrow X$ `/* Save the best solution found */`
    $NoImprove \leftarrow 0$
  else
    $NoImprove \leftarrow NoImprove + 1$
  end
end
return X^b

3.2.2. Neighborhood structure. The neighborhood structure is a highly important component of tabu search algorithms and is usually applied in the setting of discrete opti-

mization. However, for the continuous p-dispersion problems, an efficient neighborhood structure is not easy to design. In this study, we devise a vacancy-based insertion neighborhood $N_{ins}(X)$ based on the potential energies of the p dispersion points and the vacancy sites in the current configuration X, which is defined by the insertion operator that moves a high-energy dispersion point to a low-energy vacancy site.

Taking the continuous p-dispersion problem with a boundary constraint (i.e., the equal circle packing problem) as an example, we describe the vacancy-based insertion neighborhood $N_{ins}(X)$ as follows. First, the potential energy $E(c_i)$ of each circle c_i in X, which measures the degree of constraint violation of c_i in X, is calculated by using Eq. (10), and the first Q highest-energy circles constitute the candidate set P, where Q is a parameter.

$$E(c_i) = \sum_{j=1, j\neq i}^{p} l_{ij}^2 + \alpha \times (O_{i0} + \sum_{k=1}^{K} O_{ik}) \tag{10}$$

and l_{ij}, O_{i0} and O_{ik} are defined in Eqs. (7)–(9).

Then, to construct the candidate set V of low-energy vacancy sits, the algorithm performs $5p$ random detections. Specifically, for each random detection, an additional circle called the detecting circle is first placed randomly in the region Ω and then its position is locally optimized via minimizing the potential function Eq. (11), which measures the degree of constraint violation when a circle is placed at its current position, by means of the L-BFGS method (Liu and Nocedal 1989). After that, the local minimum solution returned by L-BFGS method is identified as the vacancy site detected. Finally, the first Q lowest-energy vacancy sites found are used to generate the candidate set V. It is worth noting that the minimization procedure for the potential energy function (11) is very fast and the average computational time is about 10^{-4} seconds on our computer when X contains 100 circles (or dispersion points).

$$E(v_i) = \sum_{j=1}^{p} l_{ij}^2 + \alpha \times (O_{i0} + \sum_{k=1}^{K} O_{ik}) \tag{11}$$

With the help of the candidate sets P and V, the vacancy-based insertion neighborhood $N_{ins}(X)$ of the current solution X can be easily defined as follows:

$$N_{ins}(X) = \{LS(X \oplus Move(P[i], V[j])) : 1 \leq i, j \leq Q\} \tag{12}$$

where $Move(P[i], V[j])$ denotes the insertion operation, which moves the circle (or dispersion point) $P[i]$ from its current position to the vacancy site $V[j]$, and $LS(\cdot)$ denotes a local

optimization procedure such as L-BFGS. Clearly, the size of the neighborhood $N_{ins}(X)$ equals Q^2. Figure 6 illustrates an insertion operation, where the first Q ($Q=5$) highest-energy circles are emphasized by the gridlines and the first Q lowest-energy vacancy sites obtained by the detection procedure are indicated with a dotted-line circle.

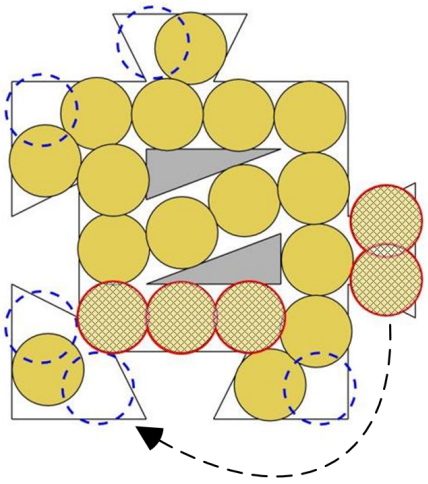


Figure 6 **The neighborhood move used in the tabu search method.**

3.2.3. Tabu list management. When a neighborhood move $Move(c_i, v_j)$ is performed, the dispersion point c_i is forbidden to move by the insertion operation for the next $tt(Iter)$ iterations, where $Iter$ is the iteration number. The tabu tenure $tt(Iter)$ is simply determined as $tt(Iter)=5+rand(0,5)$, where $rand(0,5)$ is a random number in [0,5].

3.3. Adjustment method for the minimum distance between dispersion points

Algorithm 3: Adjustment method for the minimum distance D between points

Function *AdjustDistance*
Input: Input solution (X_0, D_0), maximum number of iterations K $(=15)$
Output: Feasible local optimum solution (X, D)
$X \leftarrow X_0$, $D \leftarrow D_0$, $\mu \leftarrow 10$
for $i \leftarrow 1$ ***to*** K **do**
  $(X, D) \leftarrow Local\_Optimization(\Phi_\mu, X, D)$ `/* Minimize` $\Phi_\mu(X, D)$ `by L-BFGS method */`
  $\mu \leftarrow 5 \times \mu$
end
return (X, D)

Given an input solution X consisting of p dispersion points in region Ω and an estimated value D_0 for the minimum distance between the dispersion points, the distance adjustment procedure adjusts slightly the coordinates of the points of X, to maximize the minimum distance D between the dispersion points while preserving the feasibility of the resulting solution. Theoretically, this is equivalent to finding a locally optimal solution closest to the input solution X for the constrained optimization problem defined by Eqs. (1)–(5). To find a local optimal solution for a constrained optimization problem, we employ the well-known sequential unconstrained minimization technique (SUMT) (Fiacco and McCormick 1964). The basic idea of SUMT (Algorithm 3) is to progressively convert a constrained optimization problem into a series of unconstrained subproblems, and solves them sequentially until reaching a feasible local minimum solution for the original problem.

The constrained optimization problem corresponding to the continuous p-dispersion problem is defined by Eqs. (1)–(5) and can be progressively converted into the following unconstrained subproblems.

$$Minimize \quad \Phi_\mu(X, D) = -D^2 + \mu \times E(X, D) \tag{13}$$

where X denotes a candidate solution, μ is a penalty factor whose each value defines an unconstrained optimization function $\Phi_\mu(X, D)$, D is a variable representing the allowed minimum distance between p dispersion points, and the term $E(X, D)$ measures the degree of constraint violation in X and can be formulated as follows:

$$E(X, D) = \sum_{i=1}^{p-1} \sum_{j=i+1}^{p} l_{ij}^2 + \alpha \times (\sum_{i=1}^{p} O_{i0} + \sum_{i=1}^{p} \sum_{k=1}^{K} O_{ik}) \tag{14}$$

where l_{ij}, O_{i0} and O_{ik} are respectively defined in Eqs. (7)–(9). It is worth noting that $E(X, D)$ defined by Eq. (14) differs from $E_D(X)$ defined by Eq. (6) in Section 2.2, since $E(X, D)$ and $E_D(X)$ respectively contain $2N + 1$ and $2N$ variables, and D denotes a constant and a variable respectively in the functions $E_D(X)$ and $E(X, D)$.

4. Computational Experiments and Assessments

In this section, we evaluate the performance of the TSGO algorithm by performing extensive experiments on benchmark instances and comparing it with state-of-the-art algorithms from the literature. The main comparative results between TSGO and the reference algorithms are presented in the following subsections, and their detailed computational results are provided in the online supplement (Lai et al. 2024), where the source codes of the algorithm, the benchmark instances, and the best solutions found are also available.

4.1. Benchmark Instances, Parameter Settings and Experimental Protocol

Table 1 Settings of important parameters

Parameters	Section	Description	Values
ρ	3.1	packing density of initial solution	$\{0.85, 1.4\}$
Q	3.2.2	parameter for constructing neighborhood	3
β_{max}	3.2.1	search depth of tabu search	$\{5, 50\}$
θ_{max}	3.2.1	search depth of MBH	$\{10, 15\}$

For the equal circle packing problem, the test bed used includes 12 small-sized instances taken from (Dai et al. 2021) and Erich's packing center (`https://erich-friedman.github.io/packing/cirinl/`), together with 105 instances that we generated by constructing various regions for various problem sizes, where two regions are taken from the website `https://people.sc.fsu.edu/~jburkardt/datasets/polygon/polygon.html`. For the point arrangement problem, the test bed includes 75 instances that we generated by setting various problem sizes for these constructed regions.

The default settings and descriptions of TSGO's parameters are given in Table 1, where the values were obtained from preliminary experiments. The parameter ρ was set to 0.85 and 1.4, the parameter θ_{max} was set to 10 and 15, and the parameter β_{max} was set to 50 and 5 for the equal circle packing and point arrangement problems, respectively. The algorithm was implemented in C++ and was run on a computer with an Intel(R) Xeon (R) Platinum 9242 CPU (2.3 GHz). To evaluate the average performance, the TSGO algorithm as well as the main reference algorithms were run 10 times with different random seeds for each tested instance. The stopping criterion of the TSGO algorithm and its main reference algorithms is the maximum time limit t_{max} that varies according to the instance size p. For the CpDP problem with boundary constraints, t_{max} was set to 50 seconds for the small instances with $p \leq 50$, and set to 1, 2 and 4 hours respectively for the large instances with $p = 100$, 150 and 200. For the CpDP problem without boundary constraint, t_{max} was to 1, 4 and 8 hours respectively for the instances with $p = 50$, 100 and 150, since the local optimization method has a slow convergence for the potential functions employed.

4.2. Computational results and comparison on the equal circle packing problem

This section assesses the performance of the TSGO algorithm on the CpDP problem with boundary constraints, which corresponds to the equal circle packing problem. For the

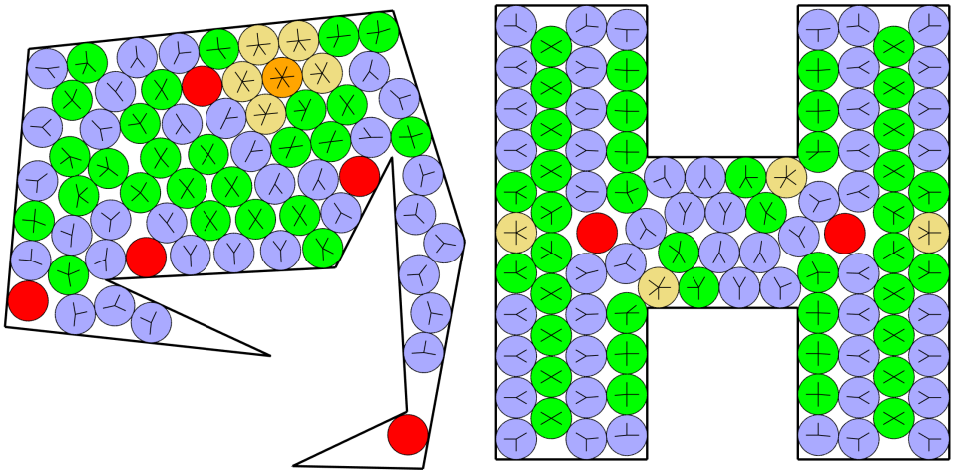
continuous p-dispersion problems in a simply-connected non-convex region, the repulsion-based dispersion algorithm (RBDA) recently proposed in (Dai et al. 2021) can be regarded as the state-of-the-art algorithm and consequently the objective of our first experiment is to compare the TSGO algorithm and the RBDA algorithm on popular benchmark instances in the literature, including 10 small-scale instances from Erich's packing center (`https://erich-friedman.github.io/packing/cirinl/`) and 2 larger instances from a previous study (Dai et al. 2021). The comparative results are summarized in Table 2. Columns 1-3 show instance features, including the container, the number ($|E|$) of edges of the container, and the number of dispersion points. Column 4 gives the best results of the RBDA algorithm, which were extracted from the literature. The remaining columns of the table give the detailed results of our TSGO algorithm, including the best objective value R_{best} (i.e., the circle radius, which equals $D_{best}/2$) over 10 independent runs, along with the average and the worst objective values R_{avg} and R_{worst}, the success rate of hitting the best solution (SR), the standard deviation of the objective values obtained (σ), and the average computational times in seconds to hit the best solution. The last row "#Best" of the table indicates the number of instances for which the corresponding result is the best in terms of the corresponding performance indicator.

Table 2 shows that our TSGO algorithm significantly outperforms the RBDA algorithm, obtaining a better R_{best} value for all instances tested. Moreover, the success rates of the TSGO algorithm is 100% for all instances and the computational time is short, especially for the small instances with $p \leq 16$. For very small instances with $p \leq 16$, the solutions obtained by both algorithms should have roughly the same geometry, and the differences in the results are caused by the precision of their results. Unlike the RBDA algorithm, TSGO attains a very high precision of 10^{-10} for the solutions. This indicates that an appropriate optimization model is very important for the effectiveness of global optimization algorithms for this continuous dispersion problem. For the two larger instances with $p = 70$ and 100, the TSGO results are appreciably superior to the RBDA results, and the improved configurations are given in Fig. 7 to show their differences relative to the previous best-known configurations published in (Dai et al. 2021).

To further evaluate TSGO's performance on more complicated instances with holes we carried out another experiment based on two sets of additional benchmark instances. The first set consists of 60 instances where the regions to be packed are relatively simple with a

Table 2 Comparison of the proposed TSGO algorithm with the best performing algorithm in the literature on 12 popular instances from the literature.

Instance			RBDA	TSGO (This work)					
Container	$\|E\|$	p	R_{best}	R_{best}	R_{avg}	R_{worst}	SR	σ	$time(s)$
E6H0	6	7	0.29446	**0.2946670216**	**0.2946670216**	**0.2946670216**	10/10	0.00	0.09
E6H0	6	8	0.28084	**0.2810468468**	**0.2810468468**	**0.2810468468**	10/10	0.00	0.09
E6H0	6	9	0.27273	**0.2729182718**	**0.2729182718**	**0.2729182718**	10/10	0.00	0.17
E6H0	6	10	0.26200	**0.2621819240**	**0.2621819240**	**0.2621819240**	10/10	0.00	0.17
E6H0	6	11	0.25017	**0.2543330951**	**0.2543330951**	**0.2543330951**	10/10	0.00	0.15
E6H0	6	12	0.24998	**0.2500000000**	**0.2500000000**	**0.2500000000**	10/10	0.00	0.14
E6H0	6	13	0.22667	**0.2269506117**	**0.2269506117**	**0.2269506117**	10/10	0.00	0.26
E6H0	6	14	0.22001	**0.2201214487**	**0.2201214487**	**0.2201214487**	10/10	0.00	1.25
E6H0	6	15	0.21153	**0.2124800251**	**0.2124800251**	**0.2124800251**	10/10	0.00	0.27
E6H0	6	16	0.20535	**0.2075604739**	**0.2075604739**	**0.2075604739**	10/10	0.00	0.36
E11H0	11	70	0.36610	**0.3729322173**	**0.3729322173**	**0.3729322173**	10/10	0.00	22.77
E12H0b	12	100	0.32867	**0.3361308135**	**0.3361308135**	**0.3361308135**	10/10	0.00	351.41
#Best			0	12	12	12			



(a) E11H0 for $p = 70$ (b) E12H0b for $p = 100$

Figure 7 Improved best solutions for two representative instances from the literature.

small number of edges and holes containing up to $p = 200$ dispersion points. In particular, the number $|H|$ of holes in the region is at most two for each instance. The second set consists of 45 instances containing up to 200 dispersion points, where the regions to be packed are much more complex with a large number of edges and holes. In this experiment, we created two TSGO variants BH* and MBH* by respectively replacing the tabu search method of TSGO with the popular basin-hoping (BH) algorithm (Wales and Doye 1997) and the monotonic basin-hopping (MBH) algorithm (Leary 2000), while keeping other TSGO components unchanged. Thus, BH* and MBH* employ the same optimization model as the TSGO algorithm, allowing us to make a fair comparison. For the subroutine BH of BH*, the maximum number of iterations was set to 1000 for a single run and the temperature parameter t was set to 0.1, and for the subroutine MBH of MBH* the search depth θ_{max} was set to 100 to ensure an intensified search.

Table 3 Comparisons of the proposed TSGO algorithm with two reference algorithms (i.e., BH* and MBH*) on the equal circle packing problem for problem instances with a region that consists of a small number of edges, where the best results among the compared algorithms are indicated in bold in terms of R_{best} and R_{avg}.

		R_{best}			R_{avg}		
Container	p	BH*	MBH*	TSGO	BH*	MBH*	TSGO
E4H0a	50	**0.0713771039**	**0.0713771039**	**0.0713771039**	**0.0713771039**	**0.0713771039**	**0.0713771039**
E4H0b	50	**0.2721671717**	**0.2721671717**	**0.2721671717**	**0.2721671717**	**0.2721671717**	**0.2721671717**
E6H0	50	**0.1226935182**	**0.1226935182**	**0.1226935182**	**0.1226935182**	**0.1226935182**	**0.1226935182**
E9H2	50	0.3686157489	0.3656662863	**0.3689738179**	0.3634125564	0.3616720252	**0.3689738179**
E11H0	50	**0.4368343737**	**0.4368343737**	**0.4368343737**	0.4357490250	0.4365780965	**0.4368343737**
E12H0a	50	**0.6225798476**	**0.6225798476**	**0.6225798476**	0.6225642581	0.6225763213	**0.6225798476**
E12H0b	50	**0.4620855791**	**0.4620855791**	**0.4620855791**	0.4619205142	0.4618933105	**0.4620855791**
E12H2	50	**0.4448612715**	**0.4448612715**	**0.4448612715**	0.4448604939	0.4448608827	**0.4448612715**
E18H0	50	**0.7877071509**	**0.7877071509**	**0.7877071509**	**0.7877071509**	**0.7877071509**	**0.7877071509**
E20H1	50	**0.4137431060**	**0.4137431060**	**0.4137431060**	0.4137379287	0.4137363551	**0.4137431060**
E20H2	50	**0.6402224988**	**0.6402224988**	**0.6402224988**	**0.6402224988**	0.6401546638	**0.6402224988**
E23H1	50	**0.4005485124**	**0.4005485124**	**0.4005485124**	**0.4005485124**	**0.4005485124**	**0.4005485124**
E27H1	50	**0.4844567507**	**0.4840050329**	**0.4844567507**	0.4843452715	0.4840050329	**0.4844567507**
E44H1	50	**0.5345742337**	0.5345665428	**0.5345742337**	0.5344076418	0.5342859578	**0.5345742337**
E59H1	50	0.0199320353	0.0198678555	**0.0199947494**	0.0198443767	0.0198127987	**0.0199947494**
E4H0a	100	**0.0514010718**	**0.0514010718**	**0.0514010718**	0.0514010668	**0.0514010718**	**0.0514010718**
E4H0b	100	**0.1978021937**	**0.1978021937**	**0.1978021937**	**0.1978021937**	**0.1978021937**	**0.1978021937**
E6H0	100	**0.0882825264**	**0.0882825264**	**0.0882825264**	**0.0882825264**	**0.0882825264**	**0.0882825264**
E9H2	100	**0.2801620820**	**0.2801620820**	**0.2801620820**	**0.2801620820**	0.2798302518	**0.2801620820**
E11H0	100	**0.3181468932**	**0.3181468932**	**0.3181468932**	0.3181468837	0.3181468853	**0.3181468932**
E12H0a	100	**0.4517241097**	**0.4517241097**	**0.4517241097**	**0.4517241097**	**0.4517241097**	**0.4517241097**
E12H0b	100	**0.3361308135**	**0.3361308135**	**0.3361308135**	**0.3361308135**	**0.3361308135**	**0.3361308135**
E12H2	100	**0.3172501921**	**0.3172501921**	**0.3172501921**	0.3172493170	**0.3172501921**	**0.3172501921**
E18H0	100	**0.5761713874**	**0.5761713874**	**0.5761713874**	**0.5761713874**	**0.5761713874**	**0.5761713874**
E20H1	100	**0.2989547054**	**0.2989547054**	**0.2989547054**	0.2989526815	**0.2989547054**	**0.2989547054**
E20H2	100	**0.4733830176**	0.4733830079	**0.4733830176**	0.4733823640	0.4733827279	**0.4733830176**
E23H1	100	**0.2896330498**	**0.2896330498**	**0.2896330498**	0.2896299959	**0.2896324594**	0.2896316545
E27H1	100	**0.3541756870**	**0.3541756870**	**0.3541756870**	**0.3541756870**	**0.3541756870**	**0.3541756870**
E44H1	100	**0.3907223295**	**0.3907223295**	**0.3907223295**	**0.3907223295**	**0.3907223295**	**0.3907223295**
E59H1	100	**0.0150349633**	**0.0150349633**	**0.0150349633**	0.0150334227	0.0150349542	**0.0150349588**
E4H0a	150	**0.0421454577**	**0.0421454577**	**0.0421454577**	0.0421453491	0.0421453491	**0.0421454454**
E4H0b	150	**0.1636890708**	0.1636890405	**0.1636890708**	0.1636890465	0.1636890405	**0.1636890708**
E6H0	150	**0.0725873025**	**0.0725873025**	**0.0725873025**	0.0725872801	**0.0725873025**	**0.0725873025**
E9H2	150	**0.2333481422**	**0.2333481422**	**0.2333481422**	0.2333137881	0.2332790355	**0.2333481422**
E11H0	150	**0.2628105627**	**0.2628105627**	**0.2628105627**	0.2628082575	0.2627997871	**0.2628104872**
E12H0a	150	**0.3730487798**	**0.3730487798**	**0.3730487798**	0.3730279634	0.3730429057	**0.3730484422**
E12H0b	150	**0.2783387066**	**0.2783387066**	**0.2783387066**	0.2783385307	0.2783384399	**0.2783387066**
E12H2	150	**0.2636360626**	**0.2636360626**	**0.2636360626**	0.2636357700	0.2636228471	**0.2636360626**
E18H0	150	0.4800548333	0.4800569423	**0.4800573753**	0.4800409940	**0.4800544890**	0.4800534988
E20H1	150	0.2508850879	0.2508904348	**0.2508905275**	0.2508718500	**0.2508899440**	0.2508892317
E20H2	150	**0.3929380796**	**0.3929380796**	**0.3929380796**	0.3929376328	0.3929379824	**0.3929380796**
E23H1	150	0.2425848102	0.2425853555	**0.2425858370**	0.2425745664	0.2425802538	**0.2425856154**
E27H1	150	0.2930400468	**0.2930778541**	0.2930756336	0.2929020446	0.2929866807	**0.2930447613**
E44H1	150	0.3253694621	**0.3253697234**	**0.3253697234**	0.3253445564	0.3253641926	**0.3253657613**
E59H1	150	0.0126020291	0.0126004282	**0.0126021266**	0.0125991512	0.0125977296	**0.0125992932**
E4H0a	200	0.0366127743	0.0366127127	**0.0366127989**	0.0366045964	0.0365977879	**0.0366083653**
E4H0b	200	**0.1429882126**	**0.1429882126**	**0.1429882126**	0.1429867411	0.1429880516	**0.1429882126**
E6H0	200	**0.0636129856**	**0.0636129856**	**0.0636129856**	0.0636129785	0.0636129789	**0.0636129856**
E9H2	200	0.2051749126	0.2051734607	**0.2051780567**	0.2051581717	0.2050920008	**0.2051759042**
E11H0	200	**0.2300479293**	0.2300479269	**0.2300479293**	0.2300475234	0.2300465204	**0.2300479293**
E12H0a	200	0.3303497674	0.3303513835	**0.3303513838**	0.3303312676	0.3303323760	**0.3303506670**
E12H0b	200	**0.2397150105**	0.2397149858	0.2397150046	0.2397144563	0.2397146114	**0.2397147555**
E12H2	200	**0.2334882838**	0.2334869495	0.2334870331	**0.2334860073**	0.2334026236	0.2334801734
E18H0	200	0.4190034532	0.4191417996	**0.4191458163**	0.4188462333	0.4190123862	**0.4190839637**
E20H1	200	0.2172626068	0.2172697806	**0.2172791061**	0.2172200262	0.2172338832	**0.2172676640**
E20H2	200	0.3446792546	0.3447052767	**0.3447056183**	0.3446675117	0.3446854835	**0.3446855508**
E23H1	200	0.2101428378	**0.2102077787**	0.2102037845	0.2101305311	0.2101637466	**0.2101661510**
E27H1	200	**0.2595897715**	**0.2595897715**	**0.2595897715**	0.2595276865	0.2595432008	**0.2595486576**
E44H1	200	0.2813674161	0.2814511662	**0.2816788437**	0.2812891330	0.2814024292	**0.2815233736**
E59H1	200	0.0111175198	0.0111161222	**0.0111178733**	0.0111155130	0.0111136582	**0.0111156356**
#Best		43	40	**56**	15	19	**56**
p-value		4.63E-04	9.16E-04		1.67E-08	1.25E-07	

The computational results of the BH*, MBH*, and TSGO algorithms are summarized in Tables 3 and 4 respectively for the first and second set of instances, including the best objective value (R_{best}) over 10 independent runs and the average objective value (R_{avg}). The row "#Best"of each table indicates the numbers of instances for which the corresponding algorithm yields the best result among the compared algorithms in terms of R_{best} and R_{avg}. The *p-values* from the Wilcoxon signed-rank test are provided in the last row to show the statistical difference between the TSGO algorithm and the BH* and MBH* algorithms in

Table 4 Comparisons of the proposed TSGO algorithm with two reference algorithms (i.e., BH* and MBH*) on the equal circle packing problem for problem instances with a complicated region consisting of a large number of edges, where the best results among the compared algorithms are indicated in bold in terms of R_{best} and R_{avg}.

		R_{best}			R_{avg}		
Container	p	BH*	MBH*	TSGO	BH*	MBH*	TSGO
E101H2	100	**0.5131508522**	**0.5131508522**	**0.5131508522**	0.5131462111	0.5131375011	**0.5131508522**
E101H3	100	0.7236466019	0.7238754116	**0.7238846144**	0.7233832936	0.7237614097	**0.7238775278**
E106H3	100	0.6039146219	**0.6039167191**	**0.6039167191**	0.6037780899	0.6038959925	**0.6039165094**
E106H5	100	0.5322579826	0.5328329107	**0.5332531100**	0.5308443285	0.5323526847	**0.5331975141**
E107H3	100	**1.1937852368**	**1.1937852368**	**1.1937852368**	**1.1937852368**	**1.1937852368**	**1.1937852368**
E120H3	100	0.8773504855	0.8773504855	**0.8773789961**	0.8772410287	0.8772947645	**0.8773533279**
E172H4	100	**0.4001953141**	0.3970925656	**0.4001953141**	0.4001810712	0.3954351442	**0.4001905629**
E193H1	100	0.8960463038	**0.8970410179**	0.8969499567	0.8953314693	0.8964794893	**0.8966769264**
E196H5	100	0.4727866982	0.4652219431	**0.4756223520**	0.4709323222	0.4595169179	**0.4756214961**
E203H3	100	0.2954793213	0.2892741421	**0.2964789245**	0.2932913997	0.2862528842	**0.2957012517**
E60H1	100	0.1389364842	**0.1389366502**	0.1389364842	0.1389127305	0.1389155735	**0.1389224366**
E81H3	100	0.4596181712	**0.4596304590**	**0.4596304590**	0.4595817333	**0.4596304590**	0.4596270833
E82H3	100	**0.4239063629**	**0.4239299929**	**0.4239299929**	0.4238394299	0.4239298328	**0.4239299929**
E84H3	100	**0.4320481250**	**0.4320481250**	**0.4320481250**	**0.4320481250**	**0.4320481250**	**0.4320481250**
E93H4	100	0.5936319873	0.5936386605	**0.5936444451**	0.5934827880	**0.5936305684**	0.5936093706
E101H2	150	0.4246039525	0.4246094078	**0.4246301877**	0.4245003932	0.4244980399	**0.4245517063**
E101H3	150	0.6022903843	0.6022967719	**0.6022981927**	0.6022903843	0.6022689505	**0.6022931886**
E106H3	150	0.4994823977	0.4996271982	**0.4996758005**	0.4994604604	0.4992973653	**0.4995933394**
E106H5	150	0.4443063213	**0.4444220001**	0.4444091109	0.4441837084	0.4442980251	**0.4443309623**
E107H3	150	0.9872584433	0.9872061372	**0.9872881106**	0.9871599470	0.9871103817	**0.9872064963**
E120H3	150	**0.7306445570**	0.7306445568	**0.7306445570**	0.7306208580	0.7303939124	**0.7306432422**
E172H4	150	0.3331491441	0.3331649258	**0.3332646035**	0.3330541489	0.3331410341	**0.3331663193**
E193H1	150	0.7408977499	**0.7409843995**	0.7409137632	**0.7406671323**	0.7405549328	0.7404794296
E196H5	150	0.3973150431	0.3971790864	**0.3997206613**	0.3955168408	0.3959672532	**0.3995365485**
E203H3	150	0.2501434557	0.2483444155	**0.2512178335**	0.2495857093	0.2473432994	**0.2500754648**
E60H1	150	0.1151603319	**0.1151655673**	0.1151505241	0.1151416702	**0.1151512904**	0.1151269139
E81H3	150	0.3801972735	**0.3802772848**	0.3802766660	0.3801341086	0.3802365212	**0.3802496299**
E82H3	150	0.3490790239	0.3490480371	**0.3490791105**	0.3490590187	0.3490358746	**0.3490722172**
E84H3	150	0.3579755298	0.3579715432	**0.3579871420**	0.3578167382	0.3578566666	**0.3579385074**
E93H4	150	0.4948046173	**0.4949205191**	0.4948698733	0.4947339911	**0.4948543190**	0.4948247350
E101H2	200	0.3707706478	0.3708004504	**0.3708266880**	0.3706979030	0.3706963168	**0.3708085633**
E101H3	200	0.5271335274	0.5268009361	**0.5271473970**	0.5268565459	0.5265542452	**0.5270959759**
E106H3	200	0.4384135384	0.4384076393	**0.4384167531**	0.4383338932	0.4382993999	**0.4383871870**
E106H5	200	0.3909836192	0.3909721826	**0.3910559351**	0.3909032381	0.3908447285	**0.3910204812**
E107H3	200	0.8600500229	**0.8601675879**	0.8601603145	0.8598515845	**0.8600649578**	0.8600083142
E120H3	200	0.6352015578	0.6352180420	**0.6352456666**	0.6349999088	0.6351366779	**0.6351595315**
E172H4	200	0.2937985244	0.2938115968	**0.2938264340**	**0.2936556747**	0.2935779286	0.2935365465
E193H1	200	0.6495304786	0.6496346640	**0.6497271655**	0.6493698892	0.6494441862	**0.6495499522**
E196H5	200	0.3528640237	0.3532782171	**0.3532967101**	0.3526929504	0.3529051796	**0.3532569447**
E203H3	200	0.2218022505	0.2230280767	**0.2233159947**	0.2214404497	0.2218444831	**0.2232194868**
E60H1	200	0.1002412092	0.1002605523	**0.1002674230**	0.1002173527	0.1002302451	**0.1002478734**
E81H3	200	0.3321541887	0.3322738793	**0.3322827528**	0.3320943976	0.3321386094	**0.3321969424**
E82H3	200	**0.3045767624**	0.3045651214	0.3045495631	**0.3044992578**	0.3044837034	0.3044844759
E84H3	200	0.3144020791	0.3143943149	**0.3144135487**	0.3143300047	0.3142707651	**0.3143854408**
E93H4	200	0.4321912739	0.4321330929	**0.4323742595**	0.4320001139	0.4318109785	**0.4322075414**
#Best		7	14	**36**	5	7	**37**
p-value		2.42E-07	2.42E-04		1.01E-06	5.30E-06	

terms of R_{best} and R_{avg}, where a *p-value* less than 0.05 means that there exists a significant difference between two groups of compared results. Detailed computational results of the three algorithms are provided in the online supplement (Lai et al. 2024).

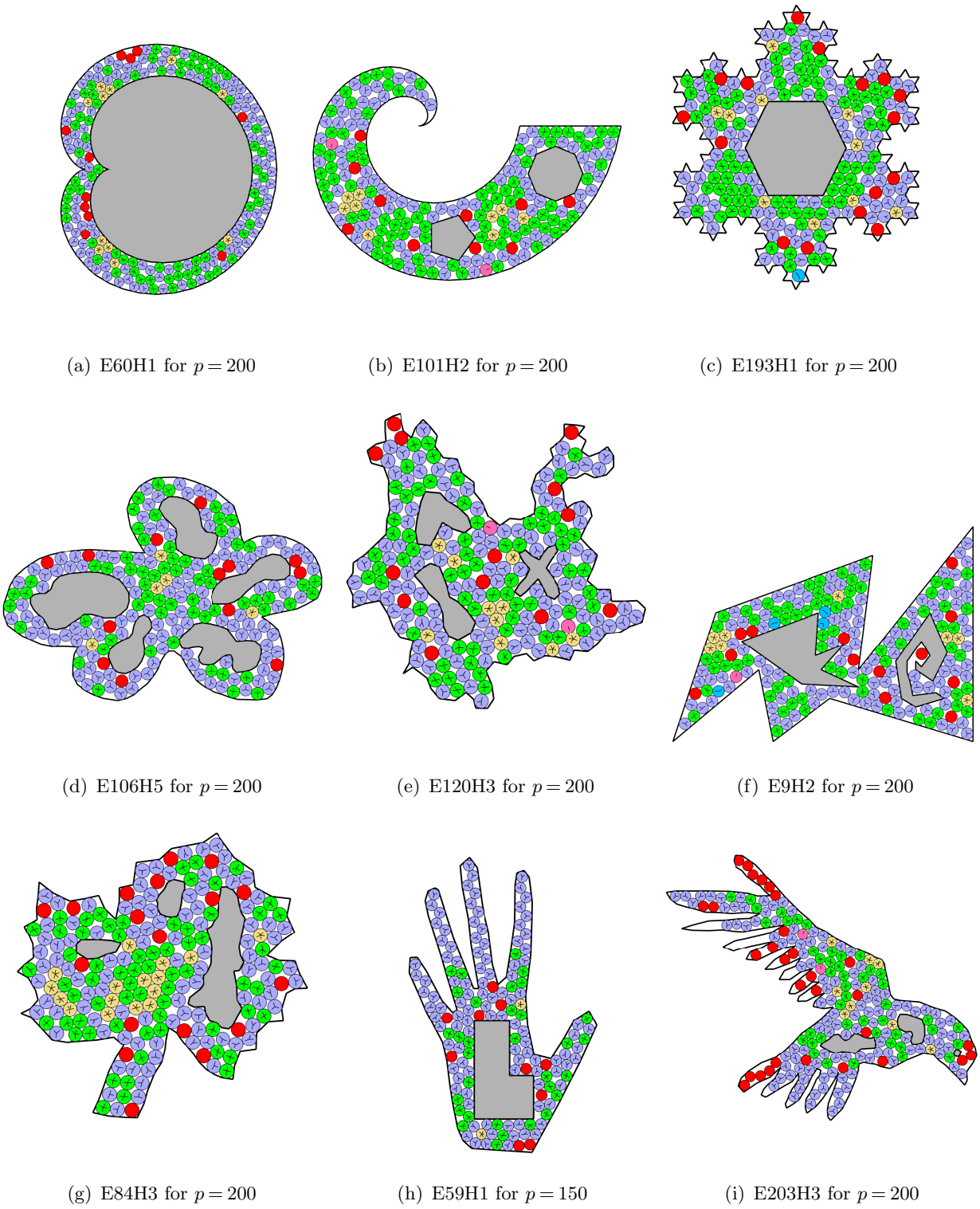
Table 3 shows that the TSGO algorithm significantly outperforms BH* and MBH* on both performance indicators. In terms of R_{best}, BH*, MBH* and TSGO respectively obtained the best result for 43, 40 and 56 out of the 60 instances. In terms of R_{avg}, the superiority of the TSGO algorithm over the BH* and MBH* algorithms is even more pronounced, where the number of instances for which the TSGO algorithm obtained the best result (56) is much larger than the numbers of instances (15 and 19) for which the BH* and MBH* algorithms obtained the best result. The small *p-values* (≤ 0.05) additionally confirm that the differences between the TSGO algorithm and the BH* and MBH* algorithms are statistically significant for both R_{best} and R_{avg}.

Table 4 further shows that for the more complicated instances the TSGO algorithm performs much better than BH* and MBH*. TSGO's superiority for these instances is more conspicuous compared to the previous instances in terms of both R_{best} and R_{avg}. In terms of R_{best}, BH*, MBH* and TSGO respectively obtained the best result for 7, 14 and 36 out of 45 instances. In terms of R_{avg}, the number of instances for which the TSGO algorithm obtained the best result is 37 which is much larger than that (5 and 7) of the BH* and MBH* algorithms. This experiment indicates that the proposed TSGO algorithm is particularly efficient for the equal circle packing problem in a complicated region composed of an irregular container and a number of irregular holes.

Fig. 8 gives graphical representations to provide an intuitive impression of the best configurations found for some representative instances. We see that the proposed model and algorithm are capable of effectively tackling highly complicated instances, including those containing a number of irregular holes and those composed of a large number of edges. The figures also show that the neighborhood operation of the tabu search, which moves high-energy dispersion points from their current positions to low-energy vacant sites, is very appropriate compared to the popular random displacement operation. Taking E9H2 with $p = 200$ as an example, it is very difficult for the sightless random displacement operation to reach the current best configuration (i.e., the subfigure (f)) when the dent of the right-hand polygonal hole does not surround a dispersion point in the initial solution.

4.3. Computational results and comparison on the point arrangement problem

The objective of this section is to assess the performance of the TSGO algorithm on the CpDP problem without a boundary constraint, which corresponds to the point arrangement problem. As in Section 4.2, two variants of the TSGO algorithm, called the BH* and MBH* algorithms, were created by replacing the tabu search method respectively with the popular basin-hopping (BH) and monotonic basin-hopping (MBH) algorithms, while keeping other components unchanged. Then, the experiments were conducted based on two sets of benchmark instances with p up to $p = 150$, where the first set contains 30 relatively simple instances for which the region to be packed consists of a small number of edges and holes and the second set contains 45 much complicated instances for which the region to be packed consists of a large number of edges and holes. The TSGO, BH* and MBH* algorithms were independently executed 10 times for each instance, and the experimental results are summarized in Tables 5 and 6 respectively for these two sets of



(a) E60H1 for $p = 200$ (b) E101H2 for $p = 200$ (c) E193H1 for $p = 200$

(d) E106H5 for $p = 200$ (e) E120H3 for $p = 200$ (f) E9H2 for $p = 200$

(g) E84H3 for $p = 200$ (h) E59H1 for $p = 150$ (i) E203H3 for $p = 200$

Figure 8 Best configurations found in this work for 9 representative instances.

instances, including the best objective value over 10 runs (D_{best}) and the average objective

Table 5 Comparisons of the proposed TSGO algorithm with two reference algorithms (i.e., BH* and MBH*) on the point arrangement problem for problem instances with a region that consists of a small number of edges, where the best results among three compared algorithms are indicated in bold both in terms of D_{best} and D_{avg}.

		D_{best}			D_{avg}		
Container	p	BH*	MBH*	TSGO	BH*	MBH*	TSGO
E4H0a	50	**0.1664988509**	**0.1664988509**	**0.1664988509**	**0.1664988509**	**0.1664988509**	**0.1664988509**
E4H0b	50	**0.7232436230**	**0.7232436230**	**0.7232436230**	**0.7232436230**	**0.7232436230**	**0.7232436230**
E6H0	50	0.2933779419	0.2933779419	**0.2933786207**	0.2933779419	0.2933779419	**0.2933780098**
E9H2	50	**1.1252199453**	**1.1252199453**	**1.1252199453**	1.1250991089	**1.1252199453**	**1.1252199453**
E11H0	50	**1.1330657353**	**1.1330657353**	**1.1330657353**	1.1330162295	**1.1330657353**	**1.1330657353**
E12H0a	50	**1.6364303754**	**1.6364303754**	**1.6364303754**	**1.6364303754**	**1.6364303754**	**1.6364303754**
E12H0b	50	**1.1512460802**	**1.1512460802**	**1.1512460802**	**1.1512460802**	**1.1512460802**	**1.1512460802**
E12H2	50	**1.2090410970**	**1.2090410970**	**1.2090410970**	1.2089809516	**1.2090410970**	**1.2090410970**
E18H0	50	**2.2664436736**	**2.2664436736**	**2.2664436736**	2.2664388856	**2.2664436736**	**2.2664436736**
E20H1	50	**1.1494074146**	**1.1494074146**	**1.1494074146**	1.1487617666	1.1483431535	**1.1492486621**
E20H2	50	1.7642062858	**1.7642062920**	1.7642062877	1.7640056517	**1.7642062866**	**1.7642062866**
E23H1	50	**1.0945028103**	**1.0945028103**	**1.0945028103**	1.0944768737	1.0943600422	**1.0944875530**
E27H1	50	**1.3363883975**	**1.3363883975**	**1.3363883975**	1.3353178683	**1.3363490513**	**1.3363490513**
E44H1	50	**1.4231184848**	**1.4231184848**	**1.4231184848**	1.4229226152	1.4230690380	**1.4231165211**
E59H1	50	0.0625113933	**0.0625365968**	**0.0625365968**	0.0624453409	0.0624835536	**0.0625045164**
E4H0a	100	0.1145681001	**0.1145682248**	**0.1145682248**	0.1145662184	**0.1145671956**	0.1145670435
E4H0b	100	**0.4826373801**	**0.4826373801**	**0.4826373801**	0.4826356082	0.4826179302	**0.4826359562**
E6H0	100	**0.1998574337**	0.1998574216	0.1998574281	0.1998560311	0.1997843134	**0.1998568131**
E9H2	100	0.7494661035	0.7511238475	**0.7511278062**	0.7478403599	0.7502858249	**0.7505626930**
E11H0	100	0.7655396868	0.7656897992	**0.7656953424**	0.7652491178	0.7655114064	**0.7656381539**
E12H0a	100	**1.1078034265**	**1.1078034265**	**1.1078034265**	**1.1077982675**	1.1077723402	1.1077925638
E12H0b	100	0.7755043144	0.7755043105	**0.7755043306**	0.7754464078	**0.7754653422**	0.7754471255
E12H2	100	**0.8146928310**	0.8146928235	**0.8146928310**	0.8143454056	**0.8146115427**	0.8145986367
E18H0	100	1.4913313807	**1.4935027529**	1.4928956213	1.4902461911	**1.4928152147**	1.4920205838
E20H1	100	0.7639769482	0.7638994850	**0.7640895485**	0.7636734678	0.7636264998	**0.7638340934**
E20H2	100	1.1842551290	1.1842672680	**1.1842695666**	1.1829491272	1.1842389685	**1.1842567749**
E23H1	100	0.7209507423	0.7211250210	**0.7211499167**	0.7196800678	0.7205693410	**0.7207608891**
E27H1	100	0.9087618011	**0.9088386862**	0.9088386700	0.9063990417	0.9077689522	**0.9087282509**
E44H1	100	0.9655365849	0.9655319844	**0.9656074713**	0.9633779411	0.9643480822	**0.9649732229**
E59H1	100	0.0424090996	**0.0424419006**	0.0424207994	0.0423971176	0.0423749032	**0.0423973095**
#Best		16	20	**25**	5	14	**25**
p-value		9.87E-04	1.40E-01		2.10E-05	4.55E-03	

value (D_{avg}). Other statistic data in the table are the same as those in Table 3. Detailed results of the compared algorithms are provided in the online supplement (Lai et al. 2024).

Table 5 shows that for the instances with a region consisting of a small number of edges the TSGO algorithm outperforms the BH* and MBH* algorithms. In terms of D_{best}, the BH*, MBH* and TSGO algorithms obtained the best result among the compared algorithms for 16, 20 and 25 out of 30 instances, respectively. Nevertheless, the large *p-values* mean there does not exist a significant difference between the MBH* and TSGO algorithms in terms of D_{best}. As for the average objective value D_{avg}, the BH*, MBH* and TSGO algorithms respectively obtained the best result for 5, 14 and 25 instances, indicating that the TSGO algorithm outperforms the BH* and MBH* algorithms on this measure. Moreover, the small *p-values* confirm that the differences between TSGO and the reference algorithms are statistically significant.

Table 6 shows that for the instances with a complicated region the TSGO algorithm also outperforms the BH* and MBH* algorithms for each performance indicator considered. Moreover, for these complicated instances the superiority of the TSGO algorithm over the BH* and MBH* algorithms is more pronounced compared to the previous instances in Table 5. Concretely, in terms of D_{best}, the BH*, MBH* and TSGO algorithms obtained

Table 6 Comparisons of the proposed TSGO algorithm with two reference algorithms (i.e., BH* and MBH*) on the point arrangement problem for problem instances with a complicated region that consists of a large number of edges, where the best results among three compared algorithms are indicated in bold both in terms of D_{best} and D_{avg}.

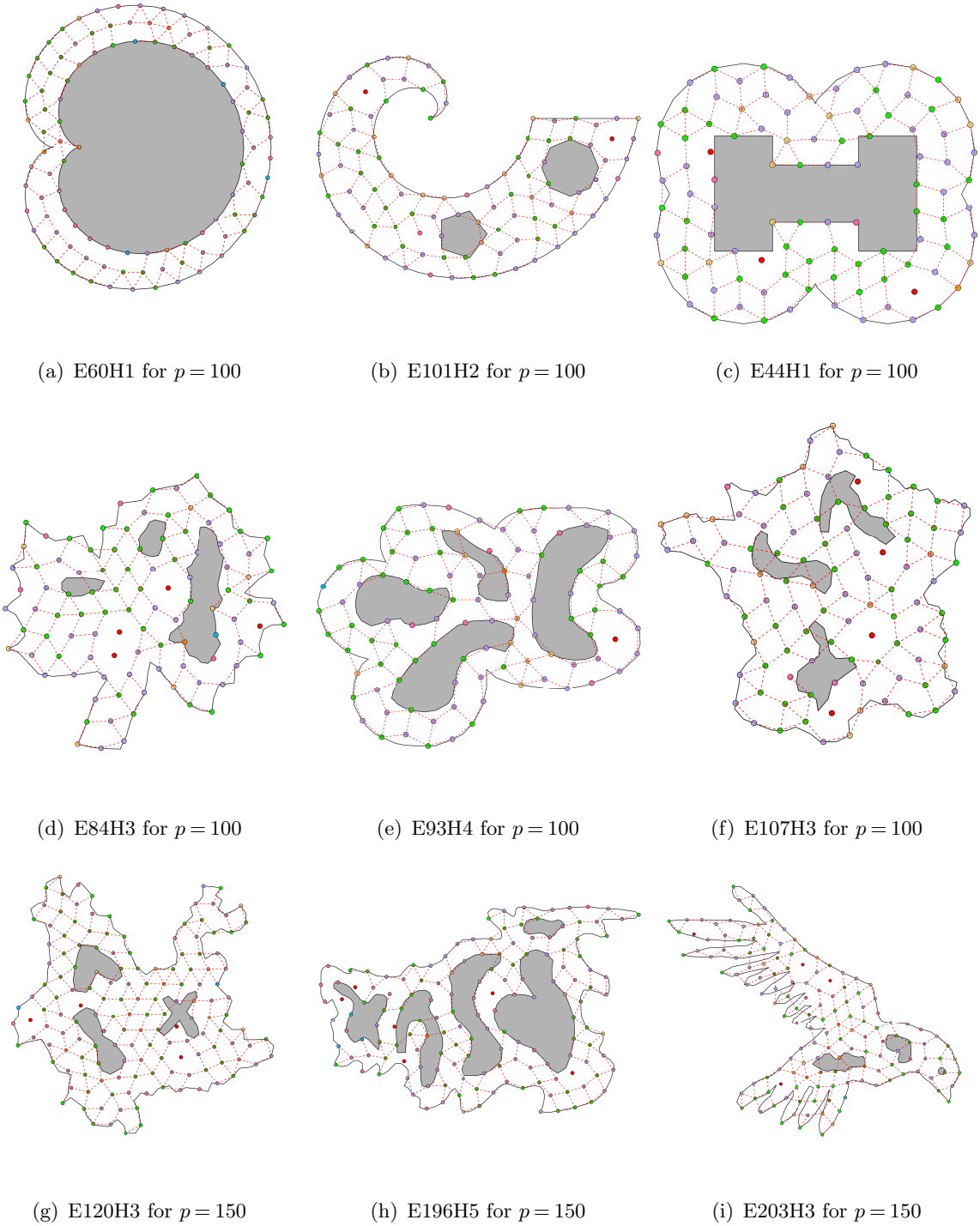
		D_{best}			D_{avg}		
Container	p	BH*	MBH*	TSGO	BH*	MBH*	TSGO
E101H2	50	1.9294792183	**1.9296611433**	**1.9296611433**	1.7373305150	**1.9293080575**	1.9290832147
E101H3	50	**2.7597012147**	**2.7597012147**	**2.7597012147**	2.7579227475	2.7528616370	**2.7583284264**
E106H3	50	2.2760303651	**2.2785596163**	**2.2785596163**	2.2721291472	2.2753498323	**2.2757095844**
E106H5	50	2.1492581495	**2.1503429086**	**2.1503429086**	2.1474806367	2.1495254107	**2.1495595896**
E107H3	50	**4.3776888646**	**4.3776888646**	**4.3776888646**	4.3744222239	4.3763245600	**4.3764749744**
E120H3	50	**3.3713160251**	**3.3713160251**	3.3711157887	3.3694128674	**3.3707797719**	3.3707419523
E172H4	50	**1.6086033153**	1.6083403888	1.6083403888	1.6073993775	**1.6080920152**	1.6079020027
E193H1	50	3.4885688530	3.4930481925	**3.4931672197**	3.4809669506	3.4828879612	**3.4849529684**
E196H5	50	2.1462437211	2.1458081483	**2.1525660024**	2.1393500463	2.1359675846	**2.1420151416**
E203H3	50	1.2947250241	**1.2963445060**	**1.2963445060**	1.2907711108	1.2944729939	**1.2950615485**
E60H1	50	**0.5509805177**	**0.5509805177**	**0.5509805177**	0.5509455932	0.5508626165	**0.5509709816**
E81H3	50	**1.6902021702**	**1.6902021702**	**1.6902021702**	1.6895559247	1.6900255476	**1.6900946298**
E82H3	50	1.5476779639	1.5477194735	**1.5477286761**	1.5460645514	1.5455161767	**1.5464949834**
E84H3	50	**1.6205855793**	**1.6205855793**	**1.6205855793**	1.6202744664	1.6202793679	**1.6205050592**
E93H4	50	2.4057626982	2.4087452204	**2.4096068833**	2.4038957388	2.4047430949	**2.4057504187**
E101H2	100	**1.2840210583**	1.2839958467	1.2833642210	1.2816485778	**1.2822682646**	1.2822230393
E101H3	100	1.8474917504	1.8517629108	**1.8527605603**	1.8454387326	1.8493839200	**1.8496615212**
E106H3	100	1.5368444673	1.5389609812	**1.5397160821**	1.5336969076	1.5373017679	**1.5382859065**
E106H5	100	1.4471249445	**1.4471430022**	**1.4471430022**	1.4448386089	1.4449916914	**1.4460133622**
E107H3	100	2.9391747678	2.9422207363	**2.9423829506**	2.9367746053	2.9394661810	**2.9397676542**
E120H3	100	2.2566551682	**2.2585254097**	2.2581001199	2.2527411251	**2.2548931409**	2.2535581665
E172H4	100	1.0758381251	**1.0767384436**	1.0759548576	1.0727718180	**1.0749063077**	1.0736695957
E193H1	100	2.3254402936	**2.3343111553**	**2.3343111553**	2.3205631956	**2.3295235036**	2.3287152398
E196H5	100	1.4098783673	1.4138861609	**1.4209052194**	1.4065769102	1.4103438073	**1.4128738037**
E203H3	100	0.8522060778	**0.8564731696**	0.8557972355	0.8507183234	0.8539301996	**0.8544934583**
E60H1	100	0.3582299427	0.3582466332	**0.3582993202**	0.3577676480	0.3578719449	**0.3580180655**
E81H3	100	1.1482805892	1.1485277749	**1.1485321215**	1.1472277187	1.1471834164	**1.1479214956**
E82H3	100	1.0473915944	**1.0474433895**	1.0474228732	1.0469480201	1.0470108082	**1.0471543246**
E84H3	100	1.0869889031	**1.0872842260**	1.0871845694	1.0853519460	1.0860224488	**1.0863459158**
E93H4	100	1.5934017314	**1.5939701632**	1.5937588256	1.5916298296	1.5925545584	**1.5930971364**
E101H2	150	1.0166314840	1.0155772115	**1.0175701224**	1.0127636545	1.0143834682	**1.0163457379**
E101H3	150	1.4749399669	**1.4759039711**	1.4758619447	1.4733512716	1.4737548666	**1.4748651219**
E106H3	150	1.2301928882	1.2307360530	**1.2316898249**	1.2272249098	1.2276956567	**1.2296099299**
E106H5	150	1.1379054440	1.1372298361	**1.1389649453**	1.1370513922	1.1364212055	**1.1373346074**
E107H3	150	2.3545859328	**2.3577371747**	2.3574562564	2.3507354789	2.3531119086	**2.3532942920**
E120H3	150	1.7957563567	1.7975153377	**1.8007023797**	1.7927535594	1.7955887570	**1.7959811597**
E172H4	150	0.8524978245	**0.8525962904**	0.8516701412	0.8508764111	0.8509029887	**0.8510752628**
E193H1	150	1.8619806126	1.8620687888	**1.8630694963**	1.8540394316	1.8561542253	**1.8562745973**
E196H5	150	1.1085450824	1.1096023844	**1.1096389040**	1.1015314163	1.1007478968	**1.1045592005**
E203H3	150	0.6798294129	0.6801848429	**0.6828531990**	0.6776296256	0.6786975696	**0.6802848226**
E60H1	150	0.2827053641	0.2828959343	**0.2829748122**	0.2824164724	0.2826403084	**0.2827246206**
E81H3	150	0.9151500861	0.9155216189	**0.9163973051**	0.9131879749	0.9141138269	**0.9149487359**
E82H3	150	0.8314206381	**0.8324378837**	0.8321274553	0.8310173646	0.8314429821	**0.8314888335**
E84H3	150	**0.8630536076**	0.8629312866	0.8629057017	0.8620981076	0.8618367930	**0.8620988622**
E93H4	150	**1.2604775170**	1.2601949020	1.2604568525	1.2587040149	1.2588156611	**1.2595900730**
#Best		10	22	30	0	7	38
p-value		3.10E-06	6.44E-02		5.18E-09	1.58E-05	

the best result for 10, 22 and 30 out of 45 instances, respectively. In terms of D_{avg}, the TSGO algorithm obtained the best result for 38 out of 45 instances, while the BH*, MBH* algorithms obtained the best result only for 0 and 7 instances, respectively.

To give an intuitive impression of the best configurations found, Fig. 9 provides the graphical representations of the best solutions for several representative instances. We observe that our TSGO algorithm is capable of effectively tackling these complicated instances for the point arrangement problem, including those with a complex container and holes as well as those consisting of a large number of edges.

5. Analysis of Algorithmic Components

We now analyze two important elements of our TSGO algorithm: the MBH procedure used to reinforce the intensified search of the tabu search method and the tabu search method



(a) E60H1 for $p = 100$ (b) E101H2 for $p = 100$ (c) E44H1 for $p = 100$

(d) E84H3 for $p = 100$ (e) E93H4 for $p = 100$ (f) E107H3 for $p = 100$

(g) E120H3 for $p = 150$ (h) E196H5 for $p = 150$ (i) E203H3 for $p = 150$

Figure 9 **Best configurations found in this work for 9 representative instances, where two dispersion points are connected by a dotted line if the distance between them equals the minimum distance found D_{best}.**

without MBH procedure (denoted by TS-MBH). A sensitivity analysis of TSGO's main parameters is shown in the online supplement (Lai et al. 2024).

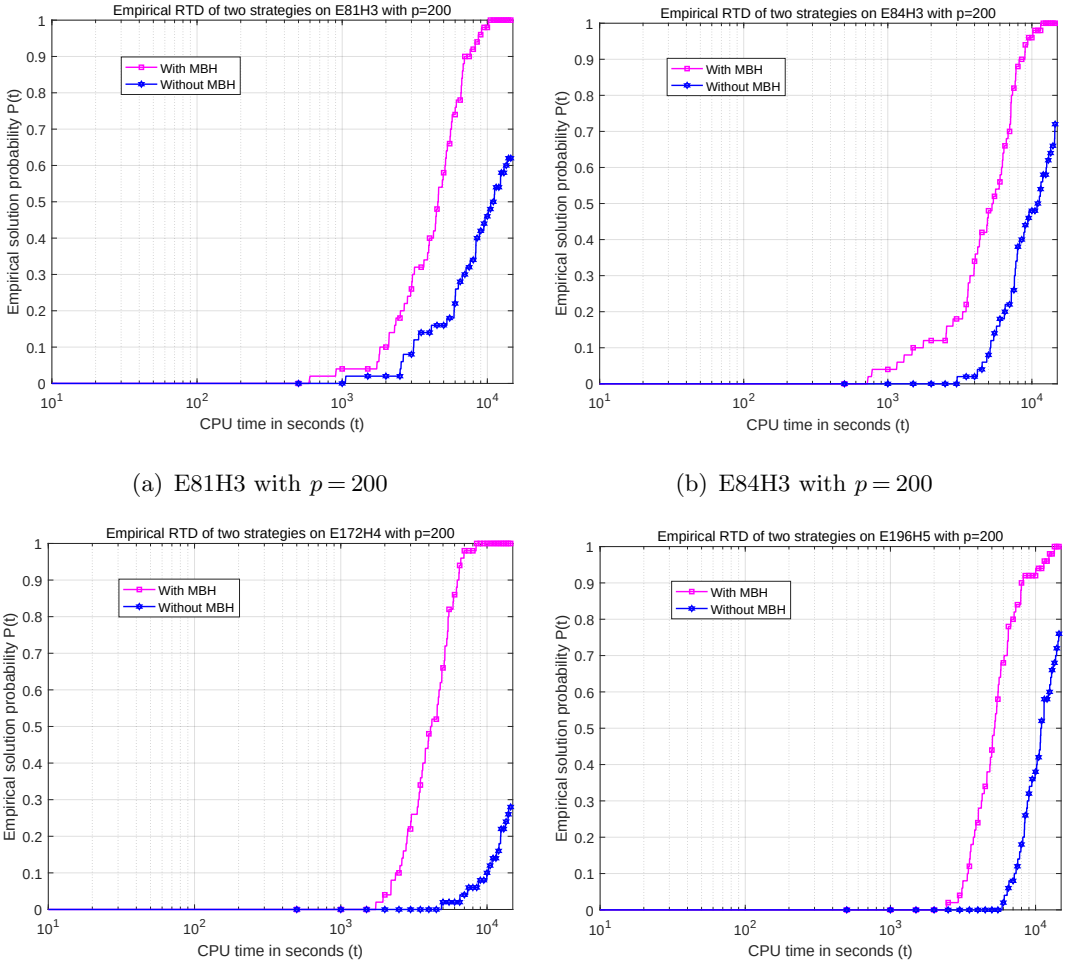
5.1. Effectiveness of the MBH procedure

Table 7 Comparison between the TSGO algorithms with and without the MBH procedure on 40 representative instances, where the best results between the compared algorithms in terms of R_{best}, R_{avg} and R_{worst} are indicated in bold.

		R_{best}		R_{avg}		R_{worst}	
Container	p	TSGO_1	TSGO	TSGO_1	TSGO	TSGO_1	TSGO
E6H0	200	0.0636129856	0.0636129856	0.0636129856	0.0636129856	0.0636129856	0.0636129856
E9H2	200	0.2051661681	**0.2051780567**	0.2051047688	**0.2051759042**	0.2050264023	**0.2051720545**
E11H0	200	0.2300454139	**0.2300479293**	0.2300337282	**0.2300479293**	0.2300317941	**0.2300479293**
E12H0a	200	0.3303438395	**0.3303513838**	0.3303170860	**0.3303506670**	0.3302830989	**0.3303497622**
E12H2	200	0.2334585328	**0.2334870331**	0.2334147356	**0.2334801734**	0.2333648795	**0.2334692402**
E18H0	200	0.4190207600	**0.4191458163**	0.4189440349	**0.4190839637**	0.4188694748	**0.4189449983**
E20H1	200	**0.2172791291**	0.2172791061	0.2171427598	**0.2172676640**	0.2170719917	**0.2172289079**
E20H2	200	0.3446755533	**0.3447056183**	0.3446691686	**0.3446855508**	0.3446593038	**0.3446820377**
E23H1	200	0.2101927860	**0.2102037845**	0.2101260082	**0.2101661510**	0.2100841375	**0.2101431359**
E27H1	200	0.2595269410	**0.2595897715**	0.2594167574	**0.2595486576**	0.2591826977	**0.2595101870**
E44H1	200	**0.2816799600**	0.2816788437	0.2814874593	**0.2815233736**	0.2813183912	**0.2814098733**
E59H1	200	0.0111103125	**0.0111178733**	0.0110977208	**0.0111156356**	0.0110894422	**0.0111114885**
E101H2	150	0.4244536975	**0.4246301877**	0.4243269607	**0.4245517063**	0.4241942738	**0.4244364627**
E101H3	150	0.6022548526	**0.6022981927**	0.6019088101	**0.6022931886**	0.6016646481	**0.6022903843**
E106H3	150	0.4994616507	**0.4996758005**	0.4994125371	**0.4995933394**	0.4993236995	**0.4994616507**
E106H5	150	0.4440727796	**0.4444091109**	0.4439297990	**0.4443309623**	0.4438270365	**0.4442520370**
E107H3	150	0.9872540160	**0.9872881106**	0.9870980239	**0.9872064963**	0.9869413117	**0.9870158687**
E120H3	150	0.7306111023	**0.7306445570**	0.7303310221	**0.7306432422**	0.7298998979	**0.7306324040**
E172H4	150	0.3327017919	**0.3332646035**	0.3322591879	**0.3331663193**	0.3319463556	**0.3330925105**
E193H1	150	0.7408295517	**0.7409137632**	0.7399829585	**0.7404794296**	0.7390118484	**0.7389815314**
E196H5	150	0.3991873083	**0.3997206613**	0.3984668794	**0.3995365485**	0.3981979101	**0.3988107216**
E203H3	150	0.2498770122	**0.2512178335**	0.2445707242	**0.2500754648**	0.2372750368	**0.2483444155**
E81H3	150	0.3802381769	**0.3802766660**	0.3800652384	**0.3802496299**	0.3799704229	**0.3801959565**
E82H3	150	0.3490326073	**0.3490791105**	0.3488907406	**0.3490722172**	0.3488160016	**0.3490514816**
E84H3	150	0.3579729123	**0.3579871420**	0.3576854002	**0.3579385074**	0.3575950893	**0.3577198430**
E93H4	150	0.4946286407	**0.4948698733**	0.4944905677	**0.4948247350**	0.4943910940	**0.4947154867**
E101H2	200	0.3707228206	**0.3708266880**	0.3707035113	**0.3708085633**	0.3706470633	**0.3708000494**
E101H3	200	0.5270364876	**0.5271473970**	0.5265310680	**0.5270959759**	0.5261496935	**0.5268497538**
E106H3	200	0.4382576604	**0.4384167531**	0.4381308018	**0.4383871870**	0.4379379803	**0.4383279802**
E106H5	200	0.3909102040	**0.3910559351**	0.3906747575	**0.3910204812**	0.3904369149	**0.3909799396**
E107H3	200	0.8601582219	**0.8601603145**	0.8596474782	**0.8600083142**	0.8592481578	**0.8597126848**
E120H3	200	0.6348111376	**0.6352456666**	0.6346367668	**0.6351595315**	0.6343821202	**0.6350429648**
E172H4	200	0.2933022486	**0.2938264340**	0.2931362207	**0.2935365465**	0.2928678971	**0.2933340360**
E193H1	200	0.6493809623	**0.6497271655**	0.6492284841	**0.6495499522**	0.6491236153	**0.6494284771**
E196H5	200	0.3528305296	**0.3532967101**	0.3523188998	**0.3532569447**	0.3509257201	**0.3531768106**
E203H3	200	0.2225266116	**0.2233159947**	0.2216821324	**0.2232194868**	0.2206605249	**0.2231246243**
E81H3	200	0.3322432698	**0.3322827528**	0.3320434488	**0.3321969424**	0.3318514747	**0.3321376262**
E82H3	200	0.3043041539	**0.3045495631**	0.3041968418	**0.3044844759**	0.3041170347	**0.3044037274**
E84H3	200	0.3143718343	**0.3144135487**	0.3143391699	**0.3143854408**	0.3142086130	**0.3143467468**
E93H4	200	0.4321793416	**0.4323742595**	0.4317616485	**0.4322075414**	0.4314124896	**0.4318942208**
#Better		2	37	0	39	0	40
#Equal		1	1	1	1	0	0
#Worse		37	2	39	0	40	0
p-value		6.64E-08		5.26E-08		7.18E-08	

As previously noted, due to the continuity of the solution space, it is likely that a number of local minima have a similar geometric configuration that will make it difficult to distinguish among there minima using tabu search. Thus, to reinforce the intensified search, the algorithm executes a very short MBH run to improve the current solution at each iteration of the tabu search method.

To check whether the MBH procedure plays an important role for the performance of our algorithm, we carried out a comparative experiment with 40 instances of the continuous



(a) E81H3 with $p = 200$

(b) E84H3 with $p = 200$

(c) E172H4 with $p = 200$

(d) E196H5 with $p = 200$

Figure 10 **Empirical run-time distribution of the TSGO algorithms with and without the MBH procedure on four representative instances.**

p-dispersion problem with boundary constraints (i.e., the equal circle packing problem). For this, we first created a variant TSGO_1 of our TSGO algorithm by disabling the MBH procedure while keeping other components of algorithm unchanged. Then, both of TSGO_1 and TSGO were run 10 times for each instance. The experimental results over 40 representative instances are summarized in Table 7, where columns 3-4 give the best results (R_{best}) over 10 independent runs respectively for TSGO_1 and TSGO, columns 5-6 give the average results (R_{avg}), and the last two columns give the worst results (R_{worst}). The rows "#Better", "#Equal", and "#Worse" respectively show the numbers of instances for which the corresponding algorithm obtains a better, equal and worse result compared with the reference algorithm. The p-values from the Wilcoxon signed-rank test are also provided in the last row of the table.

The experimental results in Table 7 show that the TSGO algorithm significantly outperforms its variant TSGO_1, which means that the MBH procedure plays an essential role in the high performance of the TSGO algorithm. Specifically, TSGO obtains a better, equal and worse result than TSGO_1 respectively for 37, 1 and 2 instances in terms of R_{best}. In terms of R_{avg}, TSGO performs better than TSGO_1 in 39 out of 40 instances and obtains equal results for the remaining instance. As for the R_{worst}, TSGO outperforms TSGO_1 on all the tested instances. Moreover, the small p-values (≤ 0.05) indicate that the differences between the results of the two algorithms are statistically significant.

In addition, we employed the empirical run-time distribution (RTD) of stochastic optimization approaches to further analyze and compare the two algorithms with and without the MBH procedure. Indeed, RTD is known to be an efficient graphic representation tool to investigate the behavior of stochastic optimization algorithms (Hoos and Stützle 2000). For a given instance, the cumulative empirical RTD of a stochastic algorithm is a function $P(t)$ mapping the run-time t to the probability of obtaining the target value within time t. Formally, the function $P(t)$ is defined as follows:

$$P(t) = \frac{|\{i : rt(i) \leq t\}|}{M} \tag{15}$$

where $rt(i)$ represents the running time of the i-th successful run to obtain the target value (the average objective value R_{avg} of TSGO_1 in Table 7 was used in this experiment) and M is the number of runs performed (where $M = 50$ in this experiment).

Fig. 10 shows TSGO's empirical RTD with and without the MBH procedure for four representative instances. One observes that for all tested instances the TSGO algorithm has a higher probability to reach the target value compared to the TSGO_1 algorithm without the MBH procedure under the same running time, confirming the critical role of the MBH procedure for the computational efficiency of the TSGO algorithm.

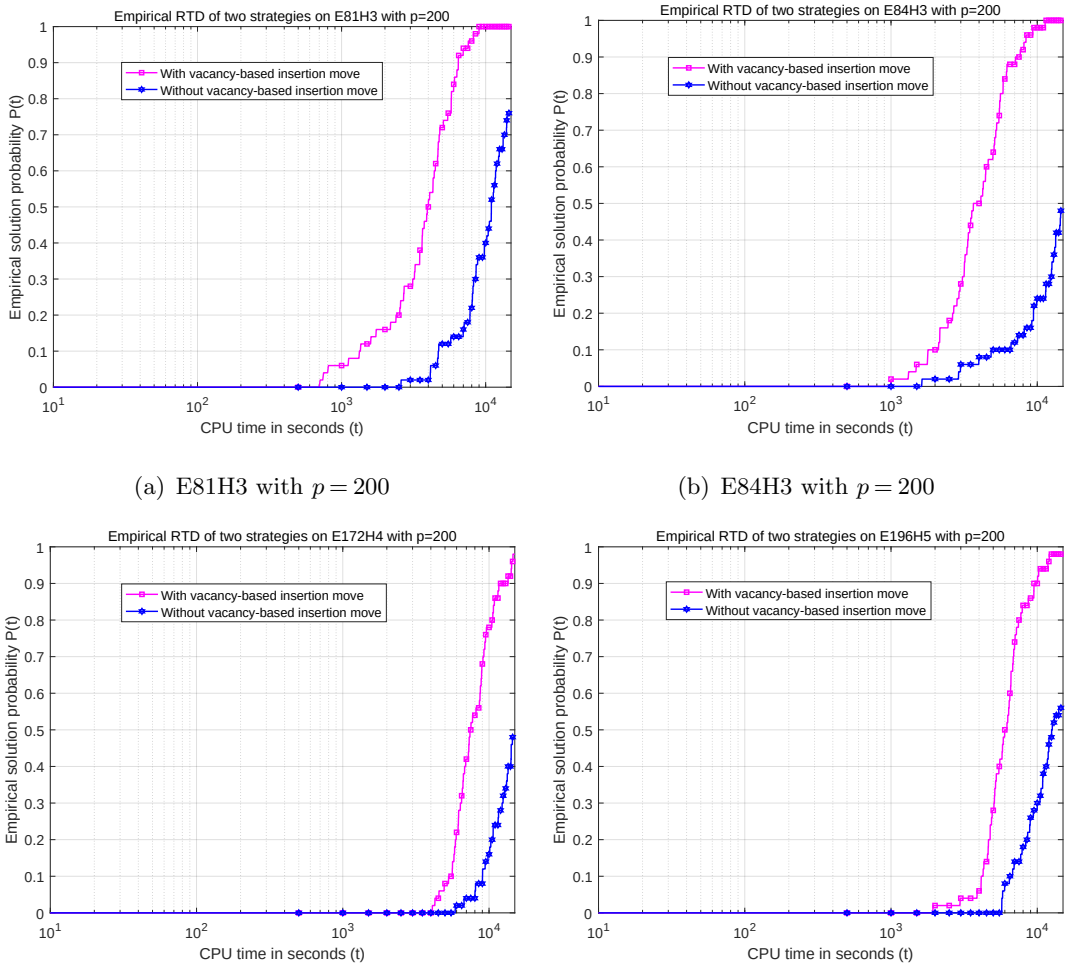
5.2. Effectiveness of the insertion neighborhood-based tabu search method

The TSGO algorithm uses the insertion neighborhood-based tabu search method without MBH (i.e., TS-MBH) as one of its main components. To check its merit, we conducted a comparative experiment based on 40 representative instances used in Section 5.1. In this experiment, we created a variant of TSGO (denote by MBH^*) by replacing the tabu search method with the MBH method, where the search depth θ_{max} of MBH was set to

Table 8 Comparison between the TSGO algorithm and MBH* on 40 representative instances, where the best results between the compared algorithms in terms of R_{best}, R_{avg} and R_{worst} are indicated in bold.

		R_{best}		R_{avg}		R_{worst}	
Container	p	MBH*	TSGO	MBH*	TSGO	MBH*	TSGO
E6H0	200	0.0636129856	0.0636129856	0.0636129697	**0.0636129856**	0.0636129182	**0.0636129856**
E9H2	200	0.2051540762	**0.2051780567**	0.2050474437	**0.2051759042**	0.2049556168	**0.2051720545**
E11H0	200	0.2300479269	**0.2300479293**	0.2300476538	**0.2300479293**	0.2300463511	**0.2300479293**
E12H0a	200	0.3303506858	**0.3303513838**	0.3303355932	**0.3303506670**	0.3302902073	**0.3303497622**
E12H2	200	0.2334693785	**0.2334870331**	0.2333686152	**0.2334801734**	0.2333409856	**0.2334692402**
E18H0	200	0.4191458163	0.4191458163	0.4189846377	**0.4190839637**	0.4188729560	**0.4189449983**
E20H1	200	0.2172791061	0.2172791061	0.2171934541	**0.2172676640**	0.2171462736	**0.2172289079**
E20H2	200	0.3446804055	**0.3447056183**	0.3446687159	**0.3446855508**	0.3446382280	**0.3446820377**
E23H1	200	0.2101523773	**0.2102037845**	0.2101274162	**0.2101661510**	0.2100656909	**0.2101431359**
E27H1	200	0.2595736347	**0.2595897715**	0.2594844815	**0.2595486576**	0.2592720014	**0.2595101870**
E44H1	200	0.2813550521	**0.2816788437**	0.2813160515	**0.2815233736**	0.2812910866	**0.2814098733**
E59H1	200	0.0111166985	**0.0111178733**	0.0111135982	**0.0111156356**	0.0111102478	**0.0111114885**
E101H2	150	0.4246265745	**0.4246301877**	0.4244914723	**0.4245517063**	0.4243906129	**0.4244364627**
E101H3	150	0.6022903843	**0.6022981927**	0.6022868453	**0.6022931886**	0.6022549939	**0.6022903843**
E106H3	150	0.4995261676	**0.4996758005**	0.4994056398	**0.4995933394**	0.4990840235	**0.4994616507**
E106H5	150	0.4443528894	**0.4444091109**	0.4442218487	**0.4443309623**	0.4440670713	**0.4442520370**
E107H3	150	0.9872493423	**0.9872881106**	0.9871345218	**0.9872064963**	0.9869322205	**0.9870158687**
E120H3	150	0.7306389612	**0.7306445570**	0.7304524692	**0.7306432422**	0.7300377900	**0.7306324040**
E172H4	150	0.3331491441	**0.3332646035**	0.3330060576	**0.3331663193**	0.3328827019	**0.3330925105**
E193H1	150	0.7407404525	**0.7409137632**	**0.7406106571**	0.7404794296	**0.7402252456**	0.7389815314
E196H5	150	0.3963913171	**0.3997206613**	0.3940397475	**0.3995365485**	0.3914854411	**0.3988107216**
E203H3	150	0.2474893922	**0.2512178335**	0.2423013081	**0.2500754648**	0.2402130965	**0.2483444155**
E81H3	150	**0.3802769198**	0.3802766660	0.3801916372	**0.3802496299**	0.3801313844	**0.3801959565**
E82H3	150	0.3490329680	**0.3490791105**	0.3489765862	**0.3490722172**	0.3488608556	**0.3490514816**
E84H3	150	0.3578615502	**0.3579871420**	0.3577509179	**0.3579385074**	0.3576006015	**0.3577198430**
E93H4	150	**0.4948711187**	0.4948698733	0.4948033224	**0.4948247350**	0.4946774844	**0.4947154867**
E101H2	200	0.3706939791	**0.3708266880**	0.3706180722	**0.3708085633**	0.3704792462	**0.3708000494**
E101H3	200	0.5270473482	**0.5271473970**	0.5265756595	**0.5270959759**	0.5263060636	**0.5268497538**
E106H3	200	0.4383759370	**0.4384167531**	0.4382643170	**0.4383871870**	0.4381635633	**0.4383279802**
E106H5	200	0.3909533120	**0.3910559351**	0.3908163599	**0.3910204812**	0.3907056241	**0.3909799396**
E107H3	200	0.8601374785	**0.8601603145**	0.8598492621	**0.8600083142**	0.8595900817	**0.8597126848**
E120H3	200	0.6351648252	**0.6352456666**	0.6350279145	**0.6351595315**	0.6349271692	**0.6350429648**
E172H4	200	0.2935898258	**0.2938264340**	0.2933538090	**0.2935365465**	0.2933022486	**0.2933340360**
E193H1	200	**0.6497860352**	0.6497271655	0.6494509651	**0.6495499522**	0.6493228504	**0.6494284771**
E196H5	200	0.3532410620	**0.3532967101**	0.3526497635	**0.3532569447**	0.3521495511	**0.3531768106**
E203H3	200	0.2217077066	**0.2233159947**	0.2213651594	**0.2232194868**	0.2209720181	**0.2231246243**
E81H3	200	0.3321164693	**0.3322827528**	0.3320201998	**0.3321969424**	0.3319349657	**0.3321376262**
E82H3	200	0.3044994679	**0.3045495631**	0.3044499252	**0.3044844759**	0.3043713087	**0.3044037274**
E84H3	200	0.3143622086	**0.3144135487**	0.3142096904	**0.3143854408**	0.3141190415	**0.3143467468**
E93H4	200	0.4319088649	**0.4323742595**	0.4317124215	**0.4322075414**	0.4315143568	**0.4318942208**
#Better		3	34	1	39	1	39
#Equal		3	3	0	0	0	0
#Worse		34	3	39	1	39	1
p-value		1.14E-06		1.84E-07		5.34E-07	

its default value. Consequently, in this variant the TS-MBH method is disabled while the other components of TSGO are kept unchanged. We ran TSGO and MBH* 10 times for each instance, and the computational results are summarized in Table 8. We observe that TSGO outperforms MBH* in all considered performance indicators. In terms of R_{best}, TSGO obtained a better, equal and worse result respectively for 34, 3 and 3 instances compared to MBH*. In terms of R_{avg} and R_{worst}, TSGO obtained a better result for 39 instances and a worse result for the remaining instance. This experiment indicates that the TS-MBH method also plays a key role for the high performance of the algorithm.



(a) E81H3 with $p = 200$ (b) E84H3 with $p = 200$

(c) E172H4 with $p = 200$ (d) E196H5 with $p = 200$

Figure 11 **Empirical run-time distribution of the TSGO algorithm and MBH* on four representative instances, where MBH* corresponds to the strategy without vacancy-based insertion moves.**

To complete this comparison, Fig. 11 provides their empirical RTD for four representative instances, where TSGO and MBH* were performed 50 times for each instance and the target value of the RTD was set to the average objective value of MBH* in Table 8. We observe that TSGO has a higher probability to reach the target value compared to MBH* under the same computational time for all tested instances.

In summary, the experiments in Sections 5.1 and 5.2 show that disabling the MBH method or the TS-MBH method deteriorates the performance of TSGO's performance. In fact, these two component are functionally complementary, where TS-MBH is capable of transforming the current solution into a significantly different new solution by applying the insertion moves, while MBH is capable of detecting the nearby local optimal solutions with

geometrical configurations very similar to that of the current solution. Thus, the combined use of these two methods results in a high performance of the TSGO algorithm.

6. Conclusions and Future Work

The continuous p-dispersion problems with and without boundary constraints, which are respectively equivalent to the classic equal circle packing problem and the point arrangement problem, have numerous important real-world applications, such as the facility location and the circle cutting problems. In this study, we investigated the general cases of the continuous p-dispersion problems, including those that involve a non-convex multiply-connected region. For this latter general case, there does not exist an effective optimization model in the literature and it is important to address the non-trivial challenge of identifying such a model as a foundation for developing appropriate solution algorithms.

By using the penalty function method, we designed an almost everywhere differentiable optimization model for the equal circle packing problem and the point arrangement problem that compose the continuous p-dispersion problems with and without boundary constraints. Based on this, we proposed a global optimization method called the TSGO utilizing the proposed model. The main component of the TSGO algorithm is a tabu search method to find a feasible solution for a given minimum distance D between dispersion points, coupled with a distance adjustment method to maximize the minimum distance between dispersion points while maintaining feasibility. The performance of the TSGO algorithm is assessed over a variety of existing and newly generated benchmark instances.

Thanks to the proposed optimization model, the popular continuous solvers (e.g., the quasi-Newton methods) can be applied to reach the high-precision solutions of the problems. Moreover, the proposed TSGO algorithm can reach a performance that no previous approach in the literature attains. The source code of the proposed algorithm, the set of benchmark instances used, and the best solutions found are available online for their potential real-world applications and future algorithmic comparisons.

There are several potential directions to extend the present study in future. First, for the continuous p-dispersion problems without a boundary constraint, the present distance adjustment method has a slow rate of convergence due to the nature of the objective function. To improve the performance of TSGO for these problems, the distance adjustment method can be replaced by the bisection method. Second, with an appropriate modification,

the present model and algorithm can be extended to handle three-dimensional problems, i.e., the continuous p-dispersion problems in a non-convex polyhedron containing multiple holes. Third, by changing the optimization model, the proposed TSGO algorithm can be extended to other packing problems, such as packing equal circles in a larger circle and packing equal spheres in a larger sphere.

Acknowledgments

We are grateful to the reviewers for their insightful comments and suggestions, which helped us to improve the paper significantly. We thank Prof. E. Pesch for his valuable helps. We also thank the Beijing Beilong Super Cloud Computing Co., Ltd (`http://www.blsc.cn/`) for providing HPC resources that have contributed to the computational experiments reported in this work. This work was partially supported by the National Natural Science Foundation of China under the grant numbers 72122006, 61703213 and 61933005.

References

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

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

Amore P (2023) Circle packing in regular polygons. *Physics of Fluids* 35(2):027130.

Baur C, Fekete SP (2001) Approximation of geometric dispersion problems. *Algorithmica* 30:451–470.

Bierlaire M, Thémans M, Zufferey N (2010) A heuristic for nonlinear global optimization. *INFORMS Journal on Computing* 22(1):59–70.

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

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

Chelouah R, Siarry P (2000) Tabu search applied to global optimization. *European Journal of Operational Rresearch* 123(2):256–270.

Chen J, Li B, Li Y (2019) Efficient approximations for the online dispersion problem. *SIAM Journal on Computing* 48(2):373–416.

Costa A (2013) Valid constraints for the point packing in a square problem. *Discrete Applied Mathematics* 161(18):2901–2909.

Dai Z, Xu K, Ornik M (2021) Repulsion-based p-dispersion with distance constraints in non-convex polygons. *Annals of Operations Research* 307:75–91.

Demaine ED, Fekete SP, Lang RJ (2010) Circle packing for origami design is hard. *arXiv preprint* arXiv:1008.1224v2.

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

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

Drezner Z, Kalczynski P, Salhi S (2019) The planar multiple obnoxious facilities location problem: A Voronoi based heuristic. *Omega* 87:105–116.

Fiacco AV, McCormick GP (1964) Computational algorithm for the sequential unconstrained minimization technique for nonlinear programming. *Management Science* 10(4):601–617.

Geißler B, Morsi A, Schewe L, Schmidt M (2018) Solving highly detailed gas transport MINLPs: block separability and penalty alternating direction methods. *INFORMS Journal on Computing* 30(2):309–323.

Glover F, Laguna M (1998) Tabu search. *Handbook of Combinatorial Optimization*, 2093–2229 (Springer).

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

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

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(1):63–81.

Hager WW, Zhang H (2005) A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM Journal on Optimization* 16(1):170–192.

Hoos HH, Stützle T (2000) Local search algorithms for SAT: An empirical evaluation. *Journal of Automated Reasoning* 24(4):421–481.

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

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

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

Lai X, Lin Z, Hao JK, Wu Q (2024) An efficient optimization model and tabu search-based global optimization approach for continuous p-dispersion problem. URL `http://dx.doi.org/10.1287/ijoc.2023.0089.cd`, https://github.com/INFORMSJoC/2023.0089.

Lai X, Yue D, Hao JK, Glover F, Lü Z (2023b) Iterated dynamic neighborhood search for packing equal circles on a sphere. *Computers & Operations Research* 151:106121.

Leary RH (2000) Global optimization on funneling landscapes. *Journal of Global Optimization* 18(4):367–383.

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

López CO, Beasley JE (2011) A heuristic for the circle packing problem with a variety of containers. *European Journal of Operational Research* 214(3):512–525.

López-Sánchez A, Sánchez-Oro J, Laguna M (2021) A new scatter search design for multiobjective combinatorial optimization with an application to facility location. *INFORMS Journal on Computing* 33(2):629–642.

Machchhar J, Elber G (2017) Dense packing of congruent circles in free-form non-convex containers. *Computer Aided Geometric Design* 52:13–27.

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

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

Mu W, Xiong S (2017) On algorithmic construction of maximin distance designs. *Communications in Statistics - Simulation and Computation* 46:7972–7985.

Schwartz B (1970) Separating points in a square. *Journal of Recreational Mathematics* 3:195–204.

Specht E (2023) Packomania website. *http://packomania.com* .

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

Szabó PG, Markót MC, Csendes T, Specht E, Casado LG, 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).

Ugray Z, Lasdon L, Plummer J, Glover F, Kelly J, Martí R (2007) Scatter search and local NLP solvers: A multistart framework for global optimization. *INFORMS Journal on Computing* 19(3):328–340.

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

Wales DJ, Doye JP (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(28):5111–5116.

Wang W, Li J, Wu E (2005) 2D point-in-polygon test by classifying edges into layers. *Computers & Graphics* 29(3):427–439.

Weaire D, Aste T (2008) *The pursuit of perfect packing* (CRC Press).

Yuan Z, Zhang Y, Dragoi M, Bai X (2018) Packing circle items in an arbitrary marble slab. *IOP Conference Series: Materials Science and Engineering*, volume 399, 012059 (IOP Publishing).

Submitted to *INFORMS Journal on Computing*
manuscript JOC-2023-03-OA-0089

Online Supplement for "An efficient optimization model and tabu search-based global optimization approach for continuous p-dispersion problem"

Xiangjing Lai
Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, P.R. China, laixiangjing@gmail.com

Zhenheng Lin
Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, P.R. China, linzhenheng@outlook.com

Jin-Kao Hao* (Corresponding author)
LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France, jin-kao.hao@univ-angers.fr

Qinghua Wu* (Corresponding author)
School of Management, Huazhong University of Science and Technology, 430074 Wuhan, P.R.China, qinghuawu@hust.edu.cn

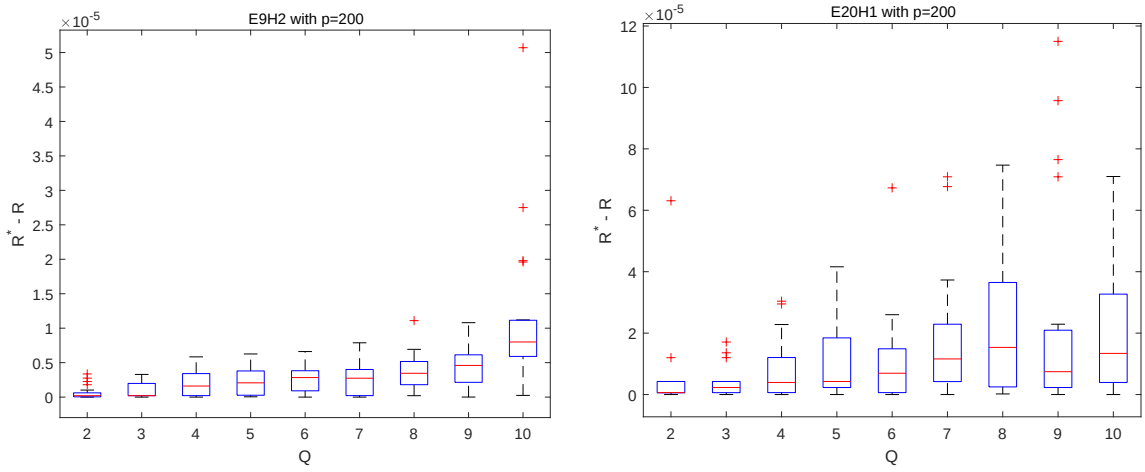
This online supplement provides a sensitivity analysis of several key parameters of the proposed TSGO algorithm and detailed computational results of the TSGO algorithm and its two main reference algorithms on benchmark instances.

Key words: Circle packing, continuous dispersion problem, global optimization, tabu search, nonlinear optimization.

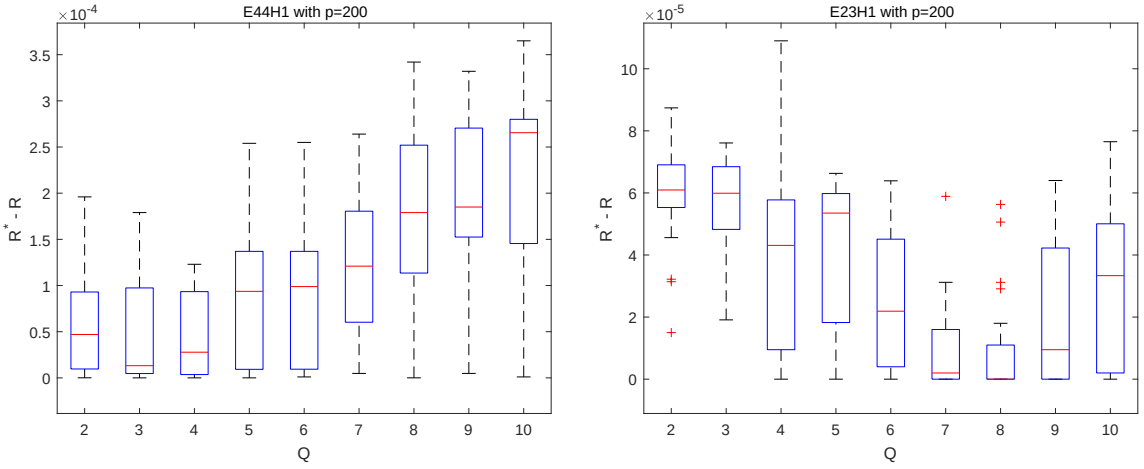
1. Sensitivity analysis of the key parameters

This section presents a sensitivity analysis of the tabu tenure of the tabu search method and three key parameters, i.e., Q, θ_{max} and β_{max}.

1.1. Sensitivity analysis of parameter Q.



(a) E9H2 with $p = 200$ (b) E20H1 with $p = 200$



(c) E44H1 with $p = 200$ (d) E23H1 with $p = 200$

Figure 1 **Influence of parameter Q on the performance of the TSGO algorithm for four representative instances.**

Recall that Q is a parameter of the tabu search component of our TSGO algorithm, which determines the size of the neighborhood $N_{ins}(X)$ by setting $|N_{ins}(X)| = Q^2$. To show the influence of the neighborhood size determined by Q on the performance of TSGO algorithm and determine a suitable Q value, we performed an experiment based on 4 representative instances of the continuous p-dispersion problem with boundary constraints.

In this experiment, the TSGO algorithm was run 20 times for each instance and for each Q value in the range of $\{2, 3, \ldots, 10\}$, recording the objective values obtained. The experimental results are given in Fig. 1 using the box plots, where the X-axis indicates the values of the parameter Q and the Y-axis indicates the gap between the objective values (R, i.e., the radius of circles) and the best result obtained (R^*).

Fig. 1 shows that the performance of the algorithm is sensitive to the setting of Q and it is difficult to find a setting that works well on all instances. For 3 out of 4 instances (i.e., E9H2, E20H1, E44H1) the algorithms with $Q \leq 3$ performs better than $Q \geq 4$, which implies that the setting of $Q \leq 3$ is suitable for most tested instances. Thus, the default value of parameter Q is set to 3. However, for E23H1 the setting of $Q = 7$ or 8 produced a significantly better result than other settings, which means that the effectiveness of parameter Q depends also on the instances to be solved and that the algorithmic performance can be further improved by fine-tuning the value of Q according to the given instance.

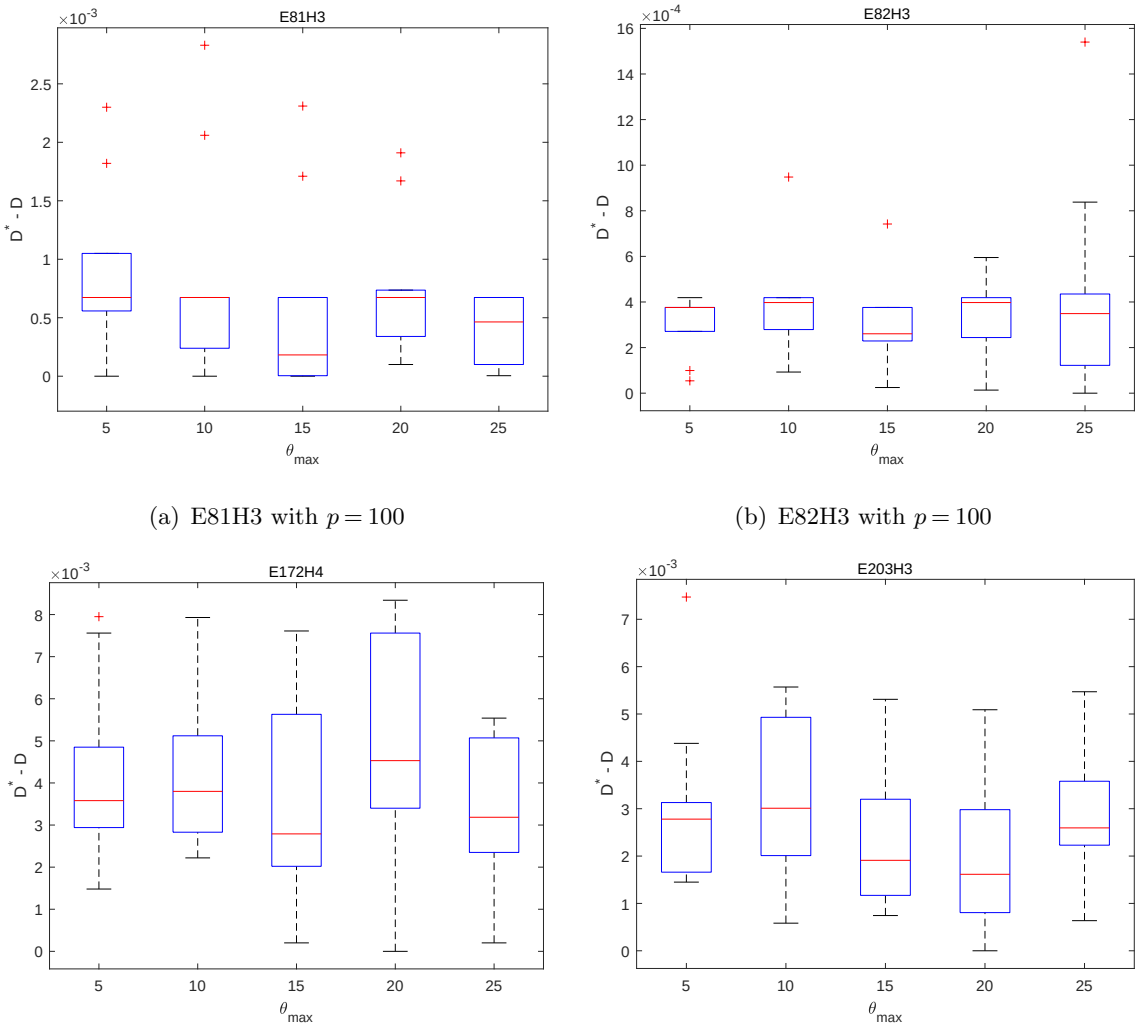
1.2. Sensitivity analysis of parameter θ_{max}.

Recall that θ_{max} is the parameter representing the search depth of the MBH method, which is one of the main components of TSGO, and a larger θ_{max} value means a more intensive search, and vice versa. To check the influence of θ_{max} on the performance of TSGO, we performed an experiment based on four representative instances of the point arrangement problem. In the experiment, the TSGO algorithm was performed 10 times for each instance and for each θ_{max} value in the range of $\{5, 10, 15, 20, 25\}$, and the results are given in Fig. 2 using the box plots, where the X-axis indicates the values of θ_{max} and the Y-axis indicates the gap between the objective values (D, i.e., the minimum distance between the dispersion points) and the best objective obtained (D^*).

Fig. 2 shows that TSGO's performance depends on the setting of parameter θ_{max} and that a too large or too small θ_{max} value deteriorates its performance. Specifically, the setting of $\theta_{max} = 15$ leads to the best performance for 3 out of 4 instances among the tested settings. Moreover, $\theta_{max} = 15$ and $\theta_{max} = 20$ lead to similar performance for the remaining instance (i.e., E203H3). Thus, the default value of θ_{max} was set to 15 for the point arrangement problem in this work.

1.3. Sensitivity analysis of parameter β_{max}.

The parameter β_{max} represents the search depth of the tabu search method, and a larger β_{max} value favors a more intensive search for the proposed algorithm, and vice versa.



(a) E81H3 with $p = 100$

(b) E82H3 with $p = 100$

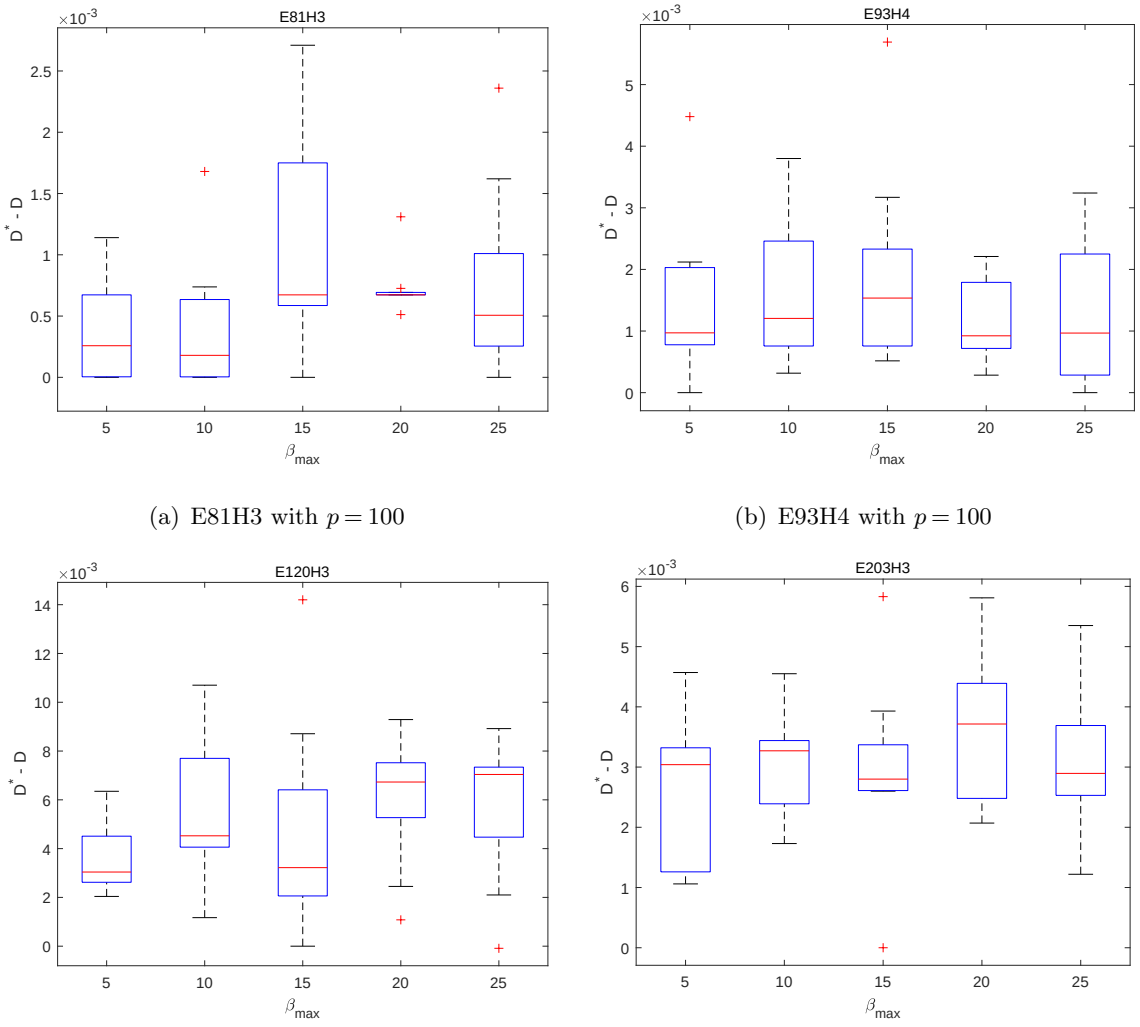
(c) E172H4 with $p = 100$

(d) E203H3 with $p = 100$

Figure 2 **Influence of parameter θ_{max} on the performance of the TSGO algorithm for four representative instances.**

To check its influence on the performance of TSGO, we conducted an experiment based on four representative instances of the point arrangement problem. In this experiment, TSGO was performed 10 times for each instance and for each β_{max} value in the range of $\{5, 10, 15, 20, 25\}$. The experiment results are summarized in Fig. 3 using the box plots, where the X-axis indicates the parameter values and the Y-axis indicates the gap between the objective values (D) and the best objective obtained (D^*).

One observes from Fig. 3 that TSGO is sensitive to the setting of parameter β_{max} and that the best setting of β_{max} depends on the instance to be solved. For example, for E81H3 with $p = 100$ $\beta_{max} = 10$ results in the best result among the five settings, but for E93H4



(a) E81H3 with $p = 100$

(b) E93H4 with $p = 100$

(c) E120H3 with $p = 100$

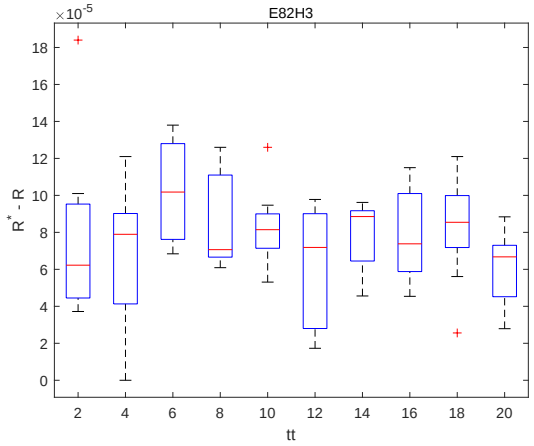
(d) E203H3 with $p = 100$

Figure 3 **Influence of parameter β_{max} on the performance of the TSGO algorithm for four representative instances.**

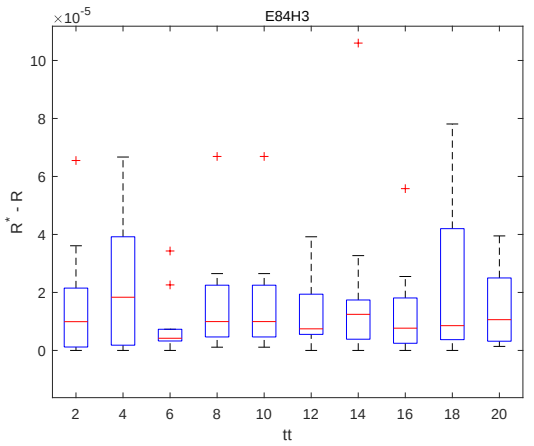
the setting of $\beta_{max} = 20$ results in the best result. On the other hand, a small β_{max} value like $\beta_{max} = 5$ is able to yield the desired result for most instances tested. Thus, the default value of β_{max} was set to 5 for the point arrangement problem in this work.

1.4. Sensitivity analysis of tabu tenure.

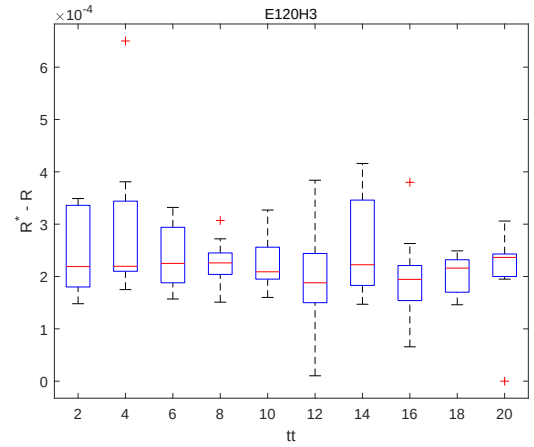
To evaluate the influence of the tabu tenure tt on the performance of the algorithm, we performed an experiment based on six representative instances of the equal circle packing problem, where the TSGO algorithm was performed 10 times for each instance and for each tabu tenure tt in the range of $\{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$. The results are summarized in Fig.4 using the box plots, where the X-axis indicates the value of tabu tenure tt and the



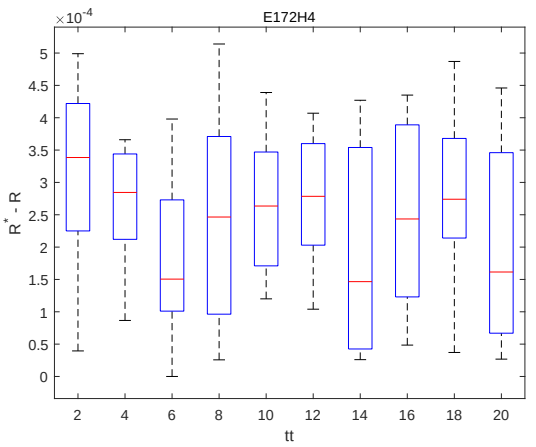
(a) E82H3 with $p = 200$



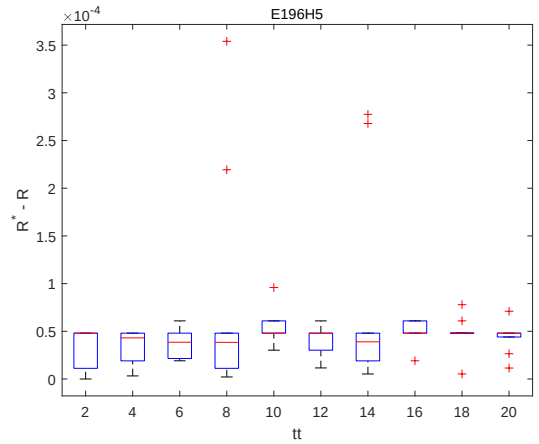
(b) E84H3 with $p = 200$



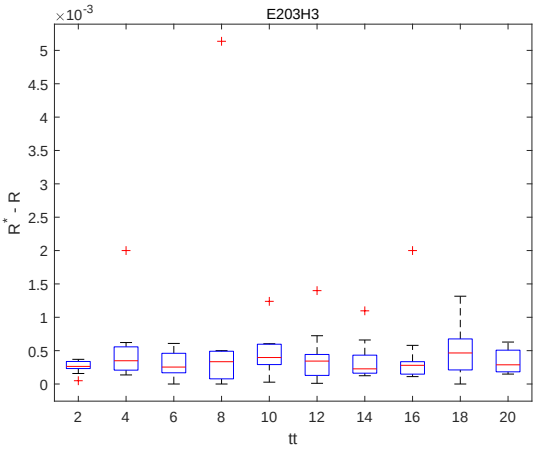
(c) E120H3 with $p = 200$



(d) E172H4 with $p = 200$



(e) E196H5 with $p = 200$



(f) E203H3 with $p = 200$

Figure 4 **Influence of tabu tenure tt on the performance of TSGO algorithm for six representative instances.**

Y-axis indicates the gap between the objective values (R, i.e., the radius of circles) and the best objective obtained (R^*).

Fig.4 shows that the best setting of the tabu tenure tt depends on the instance to be solved and it is difficult to find a general tt value for all instances. Based on the observation that the TSGO algorithm performs well for most tt values in the interval $[6, 10]$ and for most tested instances, we empirically set $tt = 5 + rand(5)$, where $rand(5)$ is a random number between 0 and 5.

2. Detailed computational results on the continuous p-dispersion problem with boundary constraints

Table 1 Computational results of the TSGO algorithm on 60 simple instances with boundary constraints.

Instance				TSGO (This work)					
Container	$\|E\|$	$\|H\|$	p	R_{best}	R_{avg}	R_{worst}	SR	σ	$time(s)$
E4H0a	4	0	50	0.0713771039	0.0713771039	0.0713771039	10/10	0.00	9.3
E4H0b	4	0	50	0.2721671717	0.2721671717	0.2721671717	10/10	0.00	5.5
E6H0	6	0	50	0.1226935182	0.1226935182	0.1226935182	10/10	0.00	2.4
E9H2	9	2	50	0.3689738179	0.3689738179	0.3689738179	10/10	0.00	19.5
E11H0	11	0	50	0.4368343737	0.4368343737	0.4368343737	10/10	0.00	11.7
E12H0a	12	0	50	0.6225798476	0.6225798476	0.6225798476	10/10	0.00	9.8
E12H0b	12	0	50	0.4620855791	0.4620855791	0.4620855791	10/10	0.00	42.6
E12H2	12	2	50	0.4448612715	0.4448612715	0.4448612715	10/10	0.00	15.7
E18H0	18	0	50	0.7877071509	0.7877071509	0.7877071509	10/10	0.00	13.3
E20H1	20	1	50	0.4137431060	0.4137431060	0.4137431060	10/10	0.00	29.4
E20H2	20	2	50	0.6402224988	0.6402224988	0.6402224988	10/10	0.00	19.4
E23H1	23	1	50	0.4005485124	0.4005485124	0.4005485124	10/10	0.00	10.6
E27H1	27	1	50	0.4844567507	0.4844567507	0.4844567507	10/10	0.00	8.8
E44H1	44	1	50	0.5345742337	0.5345742337	0.5345742337	10/10	0.00	122.4
E59H1	59	1	50	0.0199947494	0.0199947494	0.0199947494	10/10	0.00	111.8
E4H0a	4	0	100	0.0514010718	0.0514010718	0.0514010718	10/10	0.00	31.6
E4H0b	4	0	100	0.1978021937	0.1978021937	0.1978021937	10/10	0.00	110.5
E6H0	6	0	100	0.0882825264	0.0882825264	0.0882825264	10/10	0.00	77.1
E9H2	9	2	100	0.2801620820	0.2801620820	0.2801620820	10/10	0.00	107.8
E11H0	11	0	100	0.3181468932	0.3181468932	0.3181468932	10/10	0.00	142.4
E12H0a	12	0	100	0.4517241097	0.4517241097	0.4517241097	10/10	0.00	565.2
E12H0b	12	0	100	0.3361308135	0.3361308135	0.3361308135	10/10	0.00	351.4
E12H2	12	2	100	0.3172501921	0.3172501921	0.3172501921	10/10	0.00	252.1
E18H0	18	0	100	0.5761713874	0.5761713874	0.5761713874	10/10	0.00	400.1
E20H1	20	1	100	0.2989547054	0.2989547054	0.2989547054	10/10	0.00	1120.8
E20H2	20	2	100	0.4733830176	0.4733830176	0.4733830176	10/10	0.00	175.2
E23H1	23	1	100	0.2896330498	0.2896316545	0.2896216366	5/10	3.35E-06	2162.0
E27H1	27	1	100	0.3541756870	0.3541756870	0.3541756870	10/10	0.00	362.5
E44H1	44	1	100	0.3907223295	0.3907223295	0.3907223295	10/10	0.00	565.0
E59H1	59	1	100	0.0150349633	0.0150349588	0.0150349406	8/10	9.09E-09	1541.7
E4H0a	4	0	150	0.0421454577	0.0421454454	0.0421453344	9/10	3.70E-08	2062.1
E4H0b	4	0	150	0.1636890708	0.1636890708	0.1636890708	10/10	0.00	805.5
E6H0	6	0	150	0.0725873025	0.0725873025	0.0725873025	10/10	0.00	921.9
E9H2	9	2	150	0.2333481422	0.2333481422	0.2333481422	10/10	0.00	2878.0
E11H0	11	0	150	0.2628105627	0.2628104872	0.2628102971	7/10	1.15E-07	5247.1
E12H0a	12	0	150	0.3730487798	0.3730484422	0.3730454043	9/10	1.01E-06	4207.5
E12H0b	12	0	150	0.2783387066	0.2783387066	0.2783387066	10/10	0.00	1504.2
E12H2	12	2	150	0.2636360626	0.2636360626	0.2636360626	10/10	0.00	1512.0
E18H0	18	0	150	0.4800573753	0.4800534988	0.4800437063	1/10	5.02E-06	4397.4
E20H1	20	1	150	0.2508905275	0.2508892317	0.2508853392	2/10	1.68E-06	4907.4
E20H2	20	2	150	0.3929380796	0.3929380796	0.3929380796	10/10	0.00	1000.7
E23H1	23	1	150	0.2425858370	0.2425856154	0.2425853361	2/10	1.79E-07	5166.3
E27H1	27	1	150	0.2930756336	0.2930447613	0.2929186640	7/10	5.24E-05	4199.4
E44H1	44	1	150	0.3253697234	0.3253657613	0.3253339089	2/10	1.06E-05	4159.2
E59H1	59	1	150	0.0126021266	0.0125992932	0.0125955719	1/10	1.58E-06	6371.8
E4H0a	4	0	200	0.0366127989	0.0366083653	0.0365930121	3/10	7.69E-06	8538.4
E4H0b	4	0	200	0.1429882126	0.1429882126	0.1429882126	10/10	0.00	4186.0
E6H0	6	0	200	0.0636129856	0.0636129856	0.0636129856	10/10	0.00	7224.2
E9H2	9	2	200	0.2051780567	0.2051759042	0.2051720545	1/10	1.85E-06	8318.8
E11H0	11	0	200	0.2300479293	0.2300479293	0.2300479293	10/10	0.00	1821.3
E12H0a	12	0	200	0.3303513838	0.3303506670	0.3303497622	5/10	7.63E-07	4106.0
E12H0b	12	0	200	0.2397150046	0.2397147555	0.2397145332	3/10	1.81E-07	7345.3
E12H2	12	2	200	0.2334870331	0.2334801734	0.2334692402	1/10	7.44E-06	10673.5
E18H0	18	0	200	0.4191458163	0.4190839637	0.4189449983	2/10	6.27E-05	12150.4
E20H1	20	1	200	0.2172791061	0.2172676640	0.2172289079	1/10	1.42E-05	10461.0
E20H2	20	2	200	0.3447056183	0.3446855508	0.3446820377	1/10	6.79E-06	9201.7
E23H1	23	1	200	0.2102037845	0.2101661510	0.2101431359	1/10	1.99E-05	11331.5
E27H1	27	1	200	0.2595897715	0.2595486576	0.2595101870	1/10	3.16E-05	9821.7
E44H1	44	1	200	0.2816788437	0.2815233736	0.2814098733	1/10	7.34E-05	12057.6
E59H1	59	1	200	0.0111178733	0.0111156356	0.0111114885	1/10	1.72E-06	10219.1

Table 2 Computational results of the TSGO algorithm on 45 complicated instances with boundary constraints.

Instance				TSGO (This work)					
Container	$\|E\|$	$\|H\|$	p	R_{best}	R_{avg}	R_{worst}	SR	σ	$time(s)$
E101H2	101	2	100	0.5131508522	0.5131508522	0.5131508522	10/10	0.00	1397.0
E101H3	101	3	100	0.7238846144	0.7238775278	0.7238753902	2/10	3.65E-06	2268.4
E106H3	106	3	100	0.6039167191	0.6039165094	0.6039146219	9/10	6.29E-07	1982.0
E106H5	106	5	100	0.5332531100	0.5331975141	0.5331581027	1/10	2.55E-05	2813.1
E107H3	107	3	100	1.1937852368	1.1937852368	1.1937852368	10/10	0.00	690.6
E120H3	120	3	100	0.8773789961	0.8773533279	0.8773503983	1/10	8.56E-06	1495.2
E172H4	172	4	100	0.4001953141	0.4001905629	0.4001790263	5/10	6.44E-06	2856.6
E193H1	193	1	100	0.8969499567	0.8966769264	0.8963631269	1/10	1.95E-04	2435.9
E196H5	196	5	100	0.4756223520	0.4756214961	0.4756137936	9/10	2.57E-06	1817.9
E203H3	203	3	100	0.2964789245	0.2957012517	0.2950784493	1/10	3.95E-04	3460.3
E60H1	60	1	100	0.1389364842	0.1389224366	0.1389060989	3/10	1.20E-05	2364.9
E81H3	81	3	100	0.4596304590	0.4596270833	0.4596199620	5/10	3.63E-06	2565.6
E82H3	82	3	100	0.4239299929	0.4239299929	0.4239299929	10/10	0.00	1585.3
E84H3	84	3	100	0.4320481250	0.4320481250	0.4320481250	10/10	0.00	1163.4
E93H4	93	4	100	0.5936444451	0.5936093706	0.5935254279	1/10	4.51E-05	2503.9
E101H2	101	2	150	0.4246301877	0.4245517063	0.4244364627	1/10	6.42E-05	5233.0
E101H3	101	3	150	0.6022981927	0.6022931886	0.6022903843	1/10	2.51E-06	5311.9
E106H3	106	3	150	0.4996758005	0.4995933394	0.4994616507	1/10	7.46E-05	6205.4
E106H5	106	5	150	0.4444091109	0.4443309623	0.4442520370	1/10	4.18E-05	5681.8
E107H3	107	3	150	0.9872881106	0.9872064963	0.9870158687	1/10	8.05E-05	5813.1
E120H3	120	3	150	0.7306445570	0.7306432422	0.7306324040	8/10	3.62E-06	5624.6
E172H4	172	4	150	0.3332646035	0.3331663193	0.3330925105	1/10	4.18E-05	5907.3
E193H1	193	1	150	0.7409137632	0.7404794296	0.7389815314	1/10	5.26E-04	6926.1
E196H5	196	5	150	0.3997206613	0.3995365485	0.3988107216	1/10	2.65E-04	6546.5
E203H3	203	3	150	0.2512178335	0.2500754648	0.2483444155	1/10	9.41E-04	6987.4
E60H1	60	1	150	0.1151505241	0.1151269139	0.1151106888	1/10	1.08E-05	5677.9
E81H3	81	3	150	0.3802766660	0.3802496299	0.3801959565	1/10	2.89E-05	5473.0
E82H3	82	3	150	0.3490791105	0.3490722172	0.3490514816	2/10	9.80E-06	5087.5
E84H3	84	3	150	0.3579871420	0.3579385074	0.3577198430	3/10	8.18E-05	5645.2
E93H4	93	4	150	0.4948698733	0.4948247350	0.4947154867	1/10	4.19E-05	6173.2
E101H2	101	2	200	0.3708266880	0.3708085633	0.3708000494	2/10	1.01E-05	9235.6
E101H3	101	3	200	0.5271473970	0.5270959759	0.5268497538	1/10	9.64E-05	11894.7
E106H3	106	3	200	0.4384167531	0.4383871870	0.4383279802	1/10	2.87E-05	11821.2
E106H5	106	5	200	0.3910559351	0.3910204812	0.3909799396	1/10	2.56E-05	11785.9
E107H3	107	3	200	0.8601603145	0.8600083142	0.8597126848	1/10	1.69E-04	11456.7
E120H3	120	3	200	0.6352456666	0.6351595315	0.6350429648	1/10	5.28E-05	12337.9
E172H4	172	4	200	0.2938264340	0.2935365465	0.2933340360	1/10	1.54E-04	12277.3
E193H1	193	1	200	0.6497271655	0.6495499522	0.6494284771	1/10	1.19E-04	10798.2
E196H5	196	5	200	0.3532967101	0.3532569447	0.3531768106	1/10	3.09E-05	10726.4
E203H3	203	3	200	0.2233159947	0.2232194868	0.2231246243	1/10	5.38E-05	12856.2
E60H1	60	1	200	0.1002674230	0.1002478734	0.1002322688	1/10	1.02E-05	8220.7
E81H3	81	3	200	0.3322827528	0.3321969424	0.3321376262	1/10	4.86E-05	11181.5
E82H3	82	3	200	0.3045495631	0.3044844759	0.3044037274	1/10	4.26E-05	9352.3
E84H3	84	3	200	0.3144135487	0.3143854408	0.3143467468	1/10	2.24E-05	12855.1
E93H4	93	4	200	0.4323742595	0.4322075414	0.4318942208	1/10	1.55E-04	12831.8

This section reports the detailed computational results of the TSGO, BH* and MBH* algorithms for the continuous p-dispersion problem with the boundary constraints (i.e., the equal circle packing problem). Tables 1–2 give the detailed computational results of the TSGO respectively for two sets of instances, where the first set contains 60 relatively simple instances and the second set contains 45 complicated instances. Tables 3–4 give the results of the BH* algorithm, and Tables 5–6 give the results of the MBH* algorithm. The first three columns of the tables respectively indicate the names of container, the

Table 3 Computational results of the BH* algorithm on 60 simple instances with boundary constraints.

Instance				Basin-hopping					
Container	$\|E\|$	$\|H\|$	p	R_{best}	R_{avg}	R_{worst}	SR	σ	$time(s)$
E4H0a	4	0	50	0.0713771039	0.0713771039	0.0713771039	10/10	0.00	5.3
E4H0b	4	0	50	0.2721671717	0.2721671717	0.2721671717	10/10	0.00	9.4
E6H0	6	0	50	0.1226935182	0.1226935182	0.1226935182	10/10	0.00	2.9
E9H2	9	2	50	0.3686157489	0.3634125564	0.3572806998	1/10	2.87E-03	27.1
E11H0	11	0	50	0.4368343737	0.4357490250	0.4337620103	1/10	1.02E-03	40.7
E12H0a	12	0	50	0.6225798476	0.6225642581	0.6225027784	5/10	2.43E-05	29.4
E12H0b	12	0	50	0.4620855791	0.4619205142	0.4618509975	2/10	8.77E-05	29.4
E12H2	12	2	50	0.4448612715	0.4448604939	0.4448597163	1/10	3.48E-07	13.0
E18H0	18	0	50	0.7877071509	0.7877071509	0.7877071509	10/10	0.00	14.0
E20H1	20	1	50	0.4137431060	0.4137379287	0.4137333919	4/10	4.31E-06	18.3
E20H2	20	2	50	0.6402224988	0.6402224988	0.6402224988	10/10	0.00	20.2
E23H1	23	1	50	0.4005485124	0.4005485124	0.4005485124	10/10	0.00	9.0
E27H1	27	1	50	0.4844567507	0.4843452715	0.4840050329	7/10	1.81E-04	23.8
E44H1	44	1	50	0.5345742337	0.5344076418	0.5342642960	1/10	1.14E-04	36.2
E59H1	59	1	50	0.0199320353	0.0198443767	0.0197552772	1/10	3.95E-05	31.0
E4H0a	4	0	100	0.0514010718	0.0514010668	0.0514010216	9/10	1.51E-08	125.1
E4H0b	4	0	100	0.1978021937	0.1978021937	0.1978021937	10/10	0.00	72.1
E6H0	6	0	100	0.0882825264	0.0882825264	0.0882825264	10/10	0.00	161.3
E9H2	9	2	100	0.2801620820	0.2801620820	0.2801620820	10/10	0.00	788.2
E11H0	11	0	100	0.3181468932	0.3181468837	0.3181468775	4/10	7.70E-09	130.2
E12H0a	12	0	100	0.4517241097	0.4517241097	0.4517241097	10/10	0.00	934.5
E12H0b	12	0	100	0.3361308135	0.3361308135	0.3361308135	10/10	0.00	85.4
E12H2	12	2	100	0.3172501921	0.3172493170	0.3172472749	7/10	1.34E-06	1551.5
E18H0	18	0	100	0.5761713874	0.5761713874	0.5761713874	10/10	0.00	964.0
E20H1	20	1	100	0.2989547054	0.2989526815	0.2989506483	5/10	2.02E-06	1710.7
E20H2	20	2	100	0.4733830176	0.4733823640	0.4733801150	4/10	8.23E-07	708.5
E23H1	23	1	100	0.2896330498	0.2896299959	0.2896216366	4/10	4.48E-06	1791.0
E27H1	27	1	100	0.3541756870	0.3541756870	0.3541756870	10/10	0.00	216.2
E44H1	44	1	100	0.3907223295	0.3907223295	0.3907223295	10/10	0.00	1224.5
E59H1	59	1	100	0.0150349633	0.0150334227	0.0150292503	2/10	2.30E-06	2278.6
E4H0a	4	0	150	0.0421454577	0.0421453491	0.0421453344	1/10	3.69E-08	916.6
E4H0b	4	0	150	0.1636890708	0.1636890465	0.1636890405	2/10	1.21E-08	1177.0
E6H0	6	0	150	0.0725873025	0.0725872801	0.0725870787	9/10	6.72E-08	2162.5
E9H2	9	2	150	0.2333481422	0.2333137881	0.2332328075	2/10	3.81E-05	3948.0
E11H0	11	0	150	0.2628105627	0.2628082575	0.2627995664	3/10	3.67E-06	3421.3
E12H0a	12	0	150	0.3730487798	0.3730279634	0.3729689162	4/10	2.90E-05	3376.0
E12H0b	12	0	150	0.2783387066	0.2783385307	0.2783382260	4/10	2.02E-07	633.7
E12H2	12	2	150	0.2636360626	0.2636357700	0.2636338518	3/10	6.67E-07	2423.6
E18H0	18	0	150	0.4800548333	0.4800409940	0.4800160364	1/10	1.07E-05	4243.1
E20H1	20	1	150	0.2508850879	0.2508718500	0.2508473809	1/10	1.25E-05	3255.3
E20H2	20	2	150	0.3929380796	0.3929376328	0.3929355544	5/10	7.30E-07	2080.1
E23H1	23	1	150	0.2425848102	0.2425745664	0.2425624936	1/10	7.22E-06	3129.5
E27H1	27	1	150	0.2930400468	0.2929020446	0.2928217394	1/10	6.80E-05	2688.1
E44H1	44	1	150	0.3253694621	0.3253445564	0.3253234180	1/10	2.05E-05	3134.0
E59H1	59	1	150	0.0126020291	0.0125991512	0.0125946799	1/10	1.95E-06	4814.0
E4H0a	4	0	200	0.0366127743	0.0366045964	0.0365929910	1/10	9.47E-06	7416.6
E4H0b	4	0	200	0.1429882126	0.1429867411	0.1429851295	2/10	1.24E-06	7355.0
E6H0	6	0	200	0.0636129856	0.0636129785	0.0636129592	7/10	1.09E-08	4370.0
E9H2	9	2	200	0.2051749126	0.2051581717	0.2051371586	1/10	1.18E-05	7671.1
E11H0	11	0	200	0.2300479293	0.2300475234	0.2300462181	7/10	6.35E-07	6865.2
E12H0a	12	0	200	0.3303497674	0.3303312676	0.3303044960	1/10	1.62E-05	10572.9
E12H0b	12	0	200	0.2397150105	0.2397144563	0.2397141176	1/10	2.51E-07	1550.0
E12H2	12	2	200	0.2334882838	0.2334860073	0.2334799229	1/10	2.15E-06	6242.4
E18H0	18	0	200	0.4190034532	0.4188462333	0.4186487620	1/10	1.05E-04	9323.8
E20H1	20	1	200	0.2172626068	0.2172200262	0.2171481115	1/10	4.02E-05	5994.6
E20H2	20	2	200	0.3446792546	0.3446675117	0.3446425598	1/10	1.04E-05	7744.7
E23H1	23	1	200	0.2101428378	0.2101305311	0.2101131861	1/10	1.02E-05	5201.6
E27H1	27	1	200	0.2595897715	0.2595276865	0.2593957750	1/10	5.92E-05	6017.7
E44H1	44	1	200	0.2813674161	0.2812891330	0.2811962338	1/10	5.41E-05	8308.3
E59H1	59	1	200	0.0111175198	0.0111155130	0.0111133976	1/10	1.32E-06	10174.0

Table 4 Computational results of the BH* algorithm on 45 complicated instances with boundary constraints.

Instance				Basin-hopping					
Container	$\|E\|$	$\|H\|$	p	R_{best}	R_{avg}	R_{worst}	SR	σ	$time(s)$
E101H2	101	2	100	0.5131508522	0.5131462111	0.5131387318	1/10	3.47E-06	1608.6
E101H3	101	3	100	0.7236466019	0.7233832936	0.7231338721	1/10	1.52E-04	2331.5
E106H3	106	3	100	0.6039146219	0.6037780899	0.6033798239	2/10	1.44E-04	2398.2
E106H5	106	5	100	0.5322579826	0.5308443285	0.5296996479	1/10	7.64E-04	2148.6
E107H3	107	3	100	1.1937852368	1.1937852368	1.1937852368	10/10	0.00	1042.3
E120H3	120	3	100	0.8773504855	0.8772410287	0.8768077425	4/10	2.17E-04	1791.1
E172H4	172	4	100	0.4001953141	0.4001810712	0.4001767418	1/10	5.19E-06	1477.4
E193H1	193	1	100	0.8960463038	0.8953314693	0.8941600123	1/10	4.98E-04	1892.2
E196H5	196	5	100	0.4727866982	0.4709323222	0.4705322320	1/10	8.08E-04	1672.7
E203H3	203	3	100	0.2954793213	0.2932913997	0.2920748608	1/10	1.10E-03	1477.2
E60H1	60	1	100	0.1389364842	0.1389127305	0.1388932970	1/10	1.16E-05	1934.4
E81H3	81	3	100	0.4596181712	0.4595817333	0.4595293464	1/10	2.85E-05	2180.0
E82H3	82	3	100	0.4239063629	0.4238394299	0.4237861413	1/10	3.33E-05	2403.8
E84H3	84	3	100	0.4320481250	0.4320481250	0.4320481250	10/10	0.00	1026.5
E93H4	93	4	100	0.5936319873	0.5934827880	0.5931926460	1/10	1.33E-04	2473.0
E101H2	101	2	150	0.4246039525	0.4245003932	0.4244055027	1/10	7.00E-05	2861.5
E101H3	101	3	150	0.6022903843	0.6022903843	0.6022903843	10/10	0.00	3008.7
E106H3	106	3	150	0.4994823977	0.4994604604	0.4994208894	2/10	1.61E-05	5303.1
E106H5	106	5	150	0.4443063213	0.4441837084	0.4439883475	1/10	9.66E-05	4432.4
E107H3	107	3	150	0.9872584433	0.9871599470	0.9870806370	1/10	7.38E-05	4702.7
E120H3	120	3	150	0.7306445570	0.7306208580	0.7305410419	1/10	4.00E-05	4428.5
E172H4	172	4	150	0.3331491441	0.3330541489	0.3328827019	1/10	8.96E-05	4090.3
E193H1	193	1	150	0.7408977499	0.7406671323	0.7405158861	1/10	1.35E-04	5038.0
E196H5	196	5	150	0.3973150431	0.3955168408	0.3942261920	1/10	1.10E-03	4484.6
E203H3	203	3	150	0.2501434557	0.2495857093	0.2483444155	1/10	5.47E-04	3406.5
E60H1	60	1	150	0.1151603319	0.1151416702	0.1151321382	1/10	9.08E-06	3730.0
E81H3	81	3	150	0.3801972735	0.3801341086	0.3800693193	1/10	3.25E-05	3938.4
E82H3	82	3	150	0.3490790239	0.3490590187	0.3490261268	1/10	2.36E-05	2994.9
E84H3	84	3	150	0.3579755298	0.3578167382	0.3576707261	1/10	1.13E-04	5167.9
E93H4	93	4	150	0.4948046173	0.4947339911	0.4946839366	1/10	3.93E-05	4609.4
E101H2	101	2	200	0.3707706478	0.3706979030	0.3706596653	1/10	3.69E-05	10336.5
E101H3	101	3	200	0.5271335274	0.5268565459	0.5266142702	1/10	1.93E-04	9959.2
E106H3	106	3	200	0.4384135384	0.4383338932	0.4381814371	1/10	7.91E-05	11577.4
E106H5	106	5	200	0.3909836192	0.3909032381	0.3907981109	1/10	6.78E-05	10181.3
E107H3	107	3	200	0.8600500229	0.8598515845	0.8597461859	1/10	8.64E-05	10511.7
E120H3	120	3	200	0.6352015578	0.6349999088	0.6348730430	1/10	9.62E-05	9001.6
E172H4	172	4	200	0.2937985244	0.2936556747	0.2935399283	1/10	9.97E-05	10240.2
E193H1	193	1	200	0.6495304786	0.6493698892	0.6492518575	1/10	9.08E-05	9288.5
E196H5	196	5	200	0.3528640237	0.3526929504	0.3523534104	1/10	1.48E-04	9264.2
E203H3	203	3	200	0.2218022505	0.2214404497	0.2211647640	1/10	1.55E-04	9203.5
E60H1	60	1	200	0.1002412092	0.1002173527	0.1002054011	1/10	1.11E-05	9291.7
E81H3	81	3	200	0.3321541887	0.3320943976	0.3319899467	1/10	4.92E-05	6834.4
E82H3	82	3	200	0.3045767624	0.3044992578	0.3044429049	1/10	3.96E-05	9037.4
E84H3	84	3	200	0.3144020791	0.3143300047	0.3141287446	1/10	8.86E-05	11028.9
E93H4	93	4	200	0.4321912739	0.4320001139	0.4317981134	1/10	1.11E-04	9814.6

numbers of edges of polygonal container and the numbers of holes in the region, and the fourth column gives the size of problem (p). The results of the corresponding algorithm are given in columns 5–10, including the best objective value (R_{best}) over 10 independent runs, the average objective value (R_{avg}), the worst objective value (R_{worst}), the success rate (SR) of obtaining the best objective value R_{best}, the standard deviation of objective values obtained (σ), and the average running time in seconds to obtain its final result ($time(s)$) for each run of the algorithm. In addition, Fig. 5 gives the geometrical configurations of best solutions found in this work for 9 representative instances.

Table 5 Computational results of the MBH* algorithm on 60 simple instances with boundary constraints.

Instance				MBH					
Container	$\|E\|$	$\|H\|$	p	R_{best}	R_{avg}	R_{worst}	SR	σ	$time(s)$
E4H0a	4	0	50	0.0713771039	0.0713771039	0.0713771039	10/10	0.00	4.0
E4H0b	4	0	50	0.2721671717	0.2721671717	0.2721671717	10/10	0.00	5.4
E6H0	6	0	50	0.1226935182	0.1226935182	0.1226935182	10/10	0.00	4.0
E9H2	9	2	50	0.3656662863	0.3616720252	0.3566548400	1/10	2.61E-03	33.0
E11H0	11	0	50	0.4368343737	0.4365780965	0.4357715132	4/10	3.38E-04	30.4
E12H0a	12	0	50	0.6225798476	0.6225763213	0.6225445849	9/10	1.06E-05	31.7
E12H0b	12	0	50	0.4620855791	0.4618933105	0.4618376288	1/10	7.26E-05	24.3
E12H2	12	2	50	0.4448612715	0.4448608827	0.4448604939	5/10	3.89E-07	23.3
E18H0	18	0	50	0.7877071509	0.7877071509	0.7877071509	10/10	0.00	13.3
E20H1	20	1	50	0.4137431060	0.4137363551	0.4137333919	2/10	3.50E-06	16.9
E20H2	20	2	50	0.6402224988	0.6401546638	0.6395449671	8/10	2.03E-04	20.8
E23H1	23	1	50	0.4005485124	0.4005485124	0.4005485124	10/10	0.00	7.4
E27H1	27	1	50	0.4840050329	0.4840050329	0.4840050329	10/10	0.00	15.6
E44H1	44	1	50	0.5345665428	0.5342859578	0.5336099632	1/10	2.53E-04	36.3
E59H1	59	1	50	0.0198678555	0.0198127987	0.0197552311	1/10	4.44E-05	27.6
E4H0a	4	0	100	0.0514010718	0.0514010718	0.0514010718	10/10	0.00	231.4
E4H0b	4	0	100	0.1978021937	0.1978021937	0.1978021937	10/10	0.00	185.6
E6H0	6	0	100	0.0882825264	0.0882825264	0.0882825264	10/10	0.00	92.0
E9H2	9	2	100	0.2801620820	0.2798302518	0.2794630634	3/10	2.46E-04	2941.1
E11H0	11	0	100	0.3181468932	0.3181468853	0.3181468775	5/10	7.86E-09	276.7
E12H0a	12	0	100	0.4517241097	0.4517241097	0.4517241097	10/10	0.00	1200.6
E12H0b	12	0	100	0.3361308135	0.3361308135	0.3361308135	10/10	0.00	142.0
E12H2	12	2	100	0.3172501921	0.3172501921	0.3172501921	10/10	0.00	776.9
E18H0	18	0	100	0.5761713874	0.5761713874	0.5761713874	10/10	0.00	613.8
E20H1	20	1	100	0.2989547054	0.2989547054	0.2989547054	10/10	0.00	1544.6
E20H2	20	2	100	0.4733830079	0.4733827279	0.4733822983	6/10	3.51E-07	260.8
E23H1	23	1	100	0.2896330498	0.2896324594	0.2896271455	9/10	1.77E-06	1667.7
E27H1	27	1	100	0.3541756870	0.3541756870	0.3541756870	10/10	0.00	335.6
E44H1	44	1	100	0.3907223295	0.3907223295	0.3907223295	10/10	0.00	1066.9
E59H1	59	1	100	0.0150349633	0.0150349542	0.0150349406	6/10	1.11E-08	2077.0
E4H0a	4	0	150	0.0421454577	0.0421453491	0.0421453344	1/10	3.69E-08	1682.5
E4H0b	4	0	150	0.1636890405	0.1636890405	0.1636890405	10/10	0.00	1699.7
E6H0	6	0	150	0.0725873025	0.0725873025	0.0725873025	10/10	0.00	2177.9
E9H2	9	2	150	0.2333481422	0.2332790355	0.2331292927	3/10	7.44E-05	5324.0
E11H0	11	0	150	0.2628105627	0.2627997871	0.2627777167	1/10	1.32E-05	5268.9
E12H0a	12	0	150	0.3730487798	0.3730429057	0.3730115372	7/10	1.18E-05	4578.3
E12H0b	12	0	150	0.2783387066	0.2783384399	0.2783381591	2/10	1.90E-07	2081.5
E12H2	12	2	150	0.2636360626	0.2636228471	0.2635727378	3/10	2.14E-05	4563.1
E18H0	18	0	150	0.4800569423	0.4800544890	0.4800446523	1/10	3.51E-06	3928.7
E20H1	20	1	150	0.2508904348	0.2508899440	0.2508875776	2/10	8.04E-07	3335.0
E20H2	20	2	150	0.3929380796	0.3929379824	0.3929375939	8/10	1.94E-07	2313.4
E23H1	23	1	150	0.2425853555	0.2425802538	0.2425499672	1/10	1.05E-05	4884.8
E27H1	27	1	150	0.2930778541	0.2929866807	0.2928857460	1/10	7.59E-05	5448.3
E44H1	44	1	150	0.3253697234	0.3253641926	0.3253337514	1/10	1.04E-05	4020.1
E59H1	59	1	150	0.0126004282	0.0125977296	0.0125942582	1/10	2.10E-06	4983.6
E4H0a	4	0	200	0.0366127127	0.0365977879	0.0365837617	1/10	9.78E-06	9329.6
E4H0b	4	0	200	0.1429882126	0.1429880516	0.1429879687	3/10	1.06E-07	7036.3
E6H0	6	0	200	0.0636129856	0.0636129789	0.0636129632	7/10	1.03E-08	7340.8
E9H2	9	2	200	0.2051734607	0.2050920008	0.2050184327	1/10	5.92E-05	9089.7
E11H0	11	0	200	0.2300479269	0.2300465204	0.2300354350	8/10	3.73E-06	10117.1
E12H0a	12	0	200	0.3303513835	0.3303323760	0.3302731613	1/10	2.88E-05	13302.0
E12H0b	12	0	200	0.2397149858	0.2397146114	0.2397142110	1/10	2.28E-07	2692.9
E12H2	12	2	200	0.2334869495	0.2334026236	0.2333424972	1/10	5.24E-05	9666.0
E18H0	18	0	200	0.4191417996	0.4190123862	0.4188161775	1/10	1.02E-04	11804.2
E20H1	20	1	200	0.2172697806	0.2172338832	0.2171686562	1/10	2.89E-05	13046.0
E20H2	20	2	200	0.3447052767	0.3446854835	0.3446778800	1/10	7.39E-06	9545.9
E23H1	23	1	200	0.2102077787	0.2101637466	0.2101163560	1/10	2.99E-05	11021.2
E27H1	27	1	200	0.2595897715	0.2595432008	0.2594498405	5/10	4.95E-05	10223.3
E44H1	44	1	200	0.2814511662	0.2814024292	0.2813687533	1/10	2.66E-05	10452.3
E59H1	59	1	200	0.0111161222	0.0111136582	0.0111100607	1/10	1.76E-06	10668.5

Table 6 Computational results of the MBH* algorithm on 45 complicated instances with boundary constraints.

Instance				MBH					
Container	$\|E\|$	$\|H\|$	p	R_{best}	R_{avg}	R_{worst}	SR	σ	$time(s)$
E101H2	101	2	100	0.5131508522	0.5131375011	0.5130322996	3/10	3.51E-05	2676.4
E101H3	101	3	100	0.7238754116	0.7237614097	0.7235120559	1/10	1.54E-04	2611.4
E106H3	106	3	100	0.6039167191	0.6038959925	0.6037136472	7/10	6.08E-05	2397.7
E106H5	106	5	100	0.5328329107	0.5323526847	0.5312126077	1/10	5.06E-04	2869.1
E107H3	107	3	100	1.1937852368	1.1937852368	1.1937852368	10/10	0.00	604.6
E120H3	120	3	100	0.8773504855	0.8772947645	0.8768077425	6/10	1.62E-04	1857.0
E172H4	172	4	100	0.3970925656	0.3954351442	0.3916029705	1/10	1.78E-03	2578.8
E193H1	193	1	100	0.8970410179	0.8964794893	0.8961972146	1/10	2.77E-04	2572.5
E196H5	196	5	100	0.4652219431	0.4595169179	0.4543347868	1/10	3.38E-03	1804.4
E203H3	203	3	100	0.2892741421	0.2862528842	0.2818039493	1/10	2.37E-03	3476.9
E60H1	60	1	100	0.1389366502	0.1389155735	0.1389001085	1/10	1.30E-05	2501.5
E81H3	81	3	100	0.4596304590	0.4596304590	0.4596304590	10/10	0.00	2083.2
E82H3	82	3	100	0.4239299929	0.4239298328	0.4239291923	8/10	3.20E-07	1403.5
E84H3	84	3	100	0.4320481250	0.4320481250	0.4320481250	10/10	0.00	1145.2
E93H4	93	4	100	0.5936386605	0.5936305684	0.5936202818	1/10	4.78E-06	2722.0
E101H2	101	2	150	0.4246094078	0.4244980399	0.4244080511	1/10	6.59E-05	5945.0
E101H3	101	3	150	0.6022967719	0.6022689505	0.6020687695	2/10	6.68E-05	5440.3
E106H3	106	3	150	0.4996271982	0.4992973653	0.4983357367	1/10	3.45E-04	6155.2
E106H5	106	5	150	0.4444220001	0.4442980251	0.4442403628	1/10	6.28E-05	6090.1
E107H3	107	3	150	0.9872061372	0.9871103817	0.9869386607	1/10	7.77E-05	5692.9
E120H3	120	3	150	0.7306445568	0.7303939124	0.7299580764	2/10	2.72E-04	5945.4
E172H4	172	4	150	0.3331649258	0.3331410341	0.3330710559	2/10	2.55E-05	5954.0
E193H1	193	1	150	0.7409843995	0.7405549328	0.7400004318	1/10	2.59E-04	6112.8
E196H5	196	5	150	0.3971790864	0.3959672532	0.3929897609	1/10	1.29E-03	6055.5
E203H3	203	3	150	0.2483444155	0.2473432994	0.2449480525	3/10	1.08E-03	5544.1
E60H1	60	1	150	0.1151655673	0.1151512904	0.1151400557	1/10	6.82E-06	4639.9
E81H3	81	3	150	0.3802772848	0.3802365212	0.3801608921	1/10	3.94E-05	4912.3
E82H3	82	3	150	0.3490480371	0.3490358746	0.3490324492	2/10	6.08E-06	4886.6
E84H3	84	3	150	0.3579715432	0.3578566666	0.3577256969	1/10	9.31E-05	5453.8
E93H4	93	4	150	0.4949205191	0.4948543190	0.4947966163	1/10	4.22E-05	6143.9
E101H2	101	2	200	0.3708004504	0.3706963168	0.3705426420	1/10	9.21E-05	12444.9
E101H3	101	3	200	0.5268009361	0.5265542452	0.5262630474	1/10	1.91E-04	11027.1
E106H3	106	3	200	0.4384076393	0.4382993999	0.4380720423	1/10	9.62E-05	12067.6
E106H5	106	5	200	0.3909721826	0.3908447285	0.3907593101	1/10	6.44E-05	13431.8
E107H3	107	3	200	0.8601675879	0.8600649578	0.8597594956	1/10	1.32E-04	11941.6
E120H3	120	3	200	0.6352180420	0.6351366779	0.6349540662	1/10	7.72E-05	11122.8
E172H4	172	4	200	0.2938115968	0.2935779286	0.2933022486	1/10	1.77E-04	12560.0
E193H1	193	1	200	0.6496346640	0.6494441862	0.6493282527	1/10	7.93E-05	11664.1
E196H5	196	5	200	0.3532782171	0.3529051796	0.3524081717	1/10	2.09E-04	12676.7
E203H3	203	3	200	0.2230280767	0.2218444831	0.2213785287	1/10	5.53E-04	11169.2
E60H1	60	1	200	0.1002605523	0.1002302451	0.1002059940	1/10	1.44E-05	11051.1
E81H3	81	3	200	0.3322738793	0.3321386094	0.3320064877	1/10	7.22E-05	12811.5
E82H3	82	3	200	0.3045651214	0.3044837034	0.3044114861	1/10	4.78E-05	11140.6
E84H3	84	3	200	0.3143943149	0.3142707651	0.3141422594	1/10	8.67E-05	11265.4
E93H4	93	4	200	0.4321330929	0.4318109785	0.4315437465	1/10	1.82E-04	12904.3

3. Detailed computational results on the continuous p-dispersion problem without boundary constraint

This section reports the detailed computational results of TSGO, BH* and MBH* for the continuous p-dispersion problem without boundary constraint (i.e., the point arrangement problem). The results of the TSGO algorithm are summarized in Tables 7–8 respectively for two sets of benchmark instances, where the symbols are the same as in the previous tables. The results of the BH* algorithm are summarized in Tables 9–10, and the results of the MBH* algorithm are summarized in Tables 11–12. In addition, Fig. 6 gives the geometrical configurations of best solutions found in this work for 9 representative instances.

Table 7 Computational results of the TSGO algorithm on 30 simple instances without boundary constraint.

Instance				TSGO (This work)					
Container	$\|E\|$	$\|H\|$	p	D_{best}	D_{avg}	D_{worst}	SR	σ	$time(s)$
E4H0a	4	0	50	0.1664988509	0.1664988509	0.1664988509	10/10	0.00	180.5
E4H0b	4	0	50	0.7232436230	0.7232436230	0.7232436230	10/10	0.00	739.5
E6H0	6	0	50	0.2933786207	0.2933780098	0.2933779419	1/10	1.02E-07	19.1
E9H2	9	2	50	1.1252199453	1.1252199453	1.1252199453	10/10	0.00	901.6
E11H0	11	0	50	1.1330657353	1.1330657353	1.1330657353	10/10	0.00	1013.1
E12H0a	12	0	50	1.6364303754	1.6364303754	1.6364303754	10/10	0.00	616.7
E12H0b	12	0	50	1.1512460802	1.1512460802	1.1512460802	10/10	0.00	1116.5
E12H2	12	2	50	1.2090410970	1.2090410970	1.2090410970	10/10	0.00	779.9
E18H0	18	0	50	2.2664436736	2.2664436736	2.2664436736	10/10	0.00	1068.9
E20H1	20	1	50	1.1494074146	1.1492486621	1.1478238181	7/10	2.37E-04	1488.1
E20H2	20	2	50	1.7642062877	1.7642062866	1.7642062831	10/10	0.00	460.2
E23H1	23	1	50	1.0945028103	1.0944875530	1.0943526665	7/10	2.25E-05	2122.1
E27H1	27	1	50	1.3363883975	1.3363490513	1.3359949361	9/10	5.90E-05	1020.2
E44H1	44	1	50	1.4231184848	1.4231165211	1.4231119392	7/10	1.50E-06	1865.9
E59H1	59	1	50	0.0625365968	0.0625045164	0.0624105491	1/10	1.78E-05	1706.7
E4H0a	4	0	100	0.1145682248	0.1145670435	0.1145631122	2/10	8.87E-07	5158.0
E4H0b	4	0	100	0.4826373801	0.4826359562	0.4826303355	5/10	1.03E-06	4267.0
E6H0	6	0	100	0.1998574281	0.1998568131	0.1998543437	8/10	6.17E-07	5872.6
E9H2	9	2	100	0.7511278062	0.7505626930	0.7496910108	1/10	2.22E-04	8439.7
E11H0	11	0	100	0.7656953424	0.7656381539	0.7655095383	1/10	2.59E-05	5744.0
E12H0a	12	0	100	1.1078034265	1.1077925638	1.1076947990	9/10	1.63E-05	8483.8
E12H0b	12	0	100	0.7755043306	0.7754471255	0.7754149644	2/10	1.88E-05	3320.2
E12H2	12	2	100	0.8146928310	0.8145986367	0.8142894851	3/10	7.75E-05	6330.5
E18H0	18	0	100	1.4928956213	1.4920205838	1.4906975757	1/10	3.49E-04	8936.5
E20H1	20	1	100	0.7640895485	0.7638340934	0.7634178672	1/10	9.61E-05	8420.7
E20H2	20	2	100	1.1842695666	1.1842567749	1.1842401334	2/10	5.13E-06	6673.3
E23H1	23	1	100	0.7211499167	0.7207608891	0.7202295349	2/10	1.74E-04	7346.7
E27H1	27	1	100	0.9088386700	0.9087282509	0.9079030060	4/10	1.38E-04	7385.1
E44H1	44	1	100	0.9656074713	0.9649732229	0.9635173694	1/10	3.46E-04	10796.5
E59H1	59	1	100	0.0424207994	0.0423973095	0.0423371213	1/10	1.33E-05	7539.1

Table 8 Computational results of the TSGO algorithm on 45 complicated instances without boundary constraint.

Instance				TSGO (This work)					
Container	$\|E\|$	$\|H\|$	p	D_{best}	D_{avg}	D_{worst}	SR	σ	$time(s)$
E101H2	101	2	50	1.9296611433	1.9290832147	1.9260122914	6/10	5.64E-04	2716.6
E101H3	101	3	50	2.7597012147	2.7583284264	2.7500455113	8/10	1.51E-03	1522.4
E106H3	106	3	50	2.2785596163	2.2757095844	2.2735971160	1/10	9.59E-04	2635.2
E106H5	106	5	50	2.1503429086	2.1495595896	2.1482859621	4/10	3.47E-04	2788.9
E107H3	107	3	50	4.3776888646	4.3764749744	4.3676150769	7/10	1.49E-03	2774.0
E120H3	120	3	50	3.3711157887	3.3707419523	3.3688833673	5/10	3.22E-04	2084.6
E172H4	172	4	50	1.6083403888	1.6079020027	1.6062504085	5/10	3.43E-04	2573.1
E193H1	193	1	50	3.4931672197	3.4849529684	3.4732117898	1/10	2.68E-03	1422.9
E196H5	196	5	50	2.1525660024	2.1420151416	2.1257351904	1/10	4.21E-03	3887.8
E203H3	203	3	50	1.2963445060	1.2950615485	1.2913780638	3/10	7.32E-04	2812.4
E60H1	60	1	50	0.5509805177	0.5509709816	0.5509397844	6/10	7.91E-06	1875.4
E81H3	81	3	50	1.6902021702	1.6900946298	1.6898437021	7/10	8.21E-05	2245.0
E82H3	82	3	50	1.5477286761	1.5464949834	1.5433672155	1/10	7.17E-04	2686.8
E84H3	84	3	50	1.6205855793	1.6205050592	1.6201584144	5/10	7.57E-05	2096.3
E93H4	93	4	50	2.4096068833	2.4057504187	2.3995648727	2/10	1.73E-03	2549.6
E101H2	101	2	100	1.2833642210	1.2822230393	1.2811424026	1/10	3.34E-04	10678.4
E101H3	101	3	100	1.8527605603	1.8496615212	1.8473991377	1/10	9.04E-04	10525.8
E106H3	106	3	100	1.5397160821	1.5382859065	1.5367187289	1/10	4.45E-04	10505.0
E106H5	106	5	100	1.4471430022	1.4460133622	1.4435874200	1/10	6.64E-04	9858.7
E107H3	107	3	100	2.9423829506	2.9397676542	2.9356166933	1/10	1.17E-03	9959.0
E120H3	120	3	100	2.2581001199	2.2535581665	2.2504824507	1/10	1.11E-03	11658.0
E172H4	172	4	100	1.0759548576	1.0736695957	1.0710510662	1/10	8.86E-04	12042.6
E193H1	193	1	100	2.3343111553	2.3287152398	2.3192866300	2/10	2.53E-03	7972.2
E196H5	196	5	100	1.4209052194	1.4128738037	1.4083978659	1/10	1.62E-03	12567.9
E203H3	203	3	100	0.8557972355	0.8544934583	0.8525183466	1/10	4.62E-04	7296.4
E60H1	60	1	100	0.3582993202	0.3580180655	0.3577563662	1/10	8.46E-05	7573.0
E81H3	81	3	100	1.1485321215	1.1479214956	1.1469500706	1/10	2.45E-04	4950.3
E82H3	82	3	100	1.0474228732	1.0471543246	1.0470037558	1/10	6.41E-05	8110.2
E84H3	84	3	100	1.0871845694	1.0863459158	1.0841220576	1/10	4.42E-04	10003.2
E93H4	93	4	100	1.5937588256	1.5930971364	1.5900537773	1/10	5.18E-04	12201.4
E101H2	101	2	150	1.0175701224	1.0163457379	1.0153537781	1/10	3.85E-04	24165.8
E101H3	101	3	150	1.4758619447	1.4748651219	1.4739416150	1/10	3.29E-04	19676.8
E106H3	106	3	150	1.2316898249	1.2296099299	1.2267879837	1/10	7.42E-04	20253.7
E106H5	106	5	150	1.1389649453	1.1373346074	1.1361706108	1/10	4.24E-04	14919.8
E107H3	107	3	150	2.3574562564	2.3532942920	2.3477191885	1/10	1.51E-03	23193.7
E120H3	120	3	150	1.8007023797	1.7959811597	1.7913526206	1/10	1.25E-03	24401.7
E172H4	172	4	150	0.8516701412	0.8510752628	0.8504897298	1/10	1.76E-04	22075.5
E193H1	193	1	150	1.8630694963	1.8562745973	1.8529781995	1/10	1.81E-03	25078.5
E196H5	196	5	150	1.1096389040	1.1045592005	1.0996482711	1/10	1.47E-03	26994.1
E203H3	203	3	150	0.6828531990	0.6802848226	0.6787715690	1/10	5.62E-04	12261.9
E60H1	60	1	150	0.2829748122	0.2827246206	0.2825256834	1/10	5.87E-05	23385.7
E81H3	81	3	150	0.9163973051	0.9149487359	0.9134319129	1/10	3.98E-04	22917.4
E82H3	82	3	150	0.8321274553	0.8314888335	0.8307940763	1/10	1.76E-04	22437.6
E84H3	84	3	150	0.8629057017	0.8620988622	0.8609983138	1/10	3.14E-04	21476.5
E93H4	93	4	150	1.2604568525	1.2595900730	1.2577228575	1/10	4.25E-04	19305.2

Table 9 Computational results of the BH* algorithm on 30 simple instances without boundary constraint.

Instance				Basin-hopping					
Container	$\|E\|$	$\|H\|$	p	D_{best}	D_{avg}	D_{worst}	SR	σ	$time(s)$
E4H0a	4	0	50	0.1664988509	0.1664988509	0.1664988509	10/10	0.00	235.0
E4H0b	4	0	50	0.7232436230	0.7232436230	0.7232436230	10/10	0.00	1409.8
E6H0	6	0	50	0.2933779419	0.2933779419	0.2933779419	10/10	0.00	36.4
E9H2	9	2	50	1.1252199453	1.1250991089	1.1241573407	8/10	1.58E-04	1762.4
E11H0	11	0	50	1.1330657353	1.1330162295	1.1328182064	8/10	4.95E-05	1060.2
E12H0a	12	0	50	1.6364303754	1.6364303754	1.6364303754	10/10	0.00	1467.9
E12H0b	12	0	50	1.1512460802	1.1512460802	1.1512460802	10/10	0.00	244.7
E12H2	12	2	50	1.2090410970	1.2089809516	1.2089357034	4/10	2.47E-05	1317.2
E18H0	18	0	50	2.2664436736	2.2664388856	2.2664277138	7/10	3.66E-06	880.0
E20H1	20	1	50	1.1494074146	1.1487617666	1.1474782420	6/10	4.03E-04	999.7
E20H2	20	2	50	1.7642062858	1.7640056517	1.7638036026	5/10	1.00E-04	1178.0
E23H1	23	1	50	1.0945028103	1.0944768737	1.0943314661	7/10	2.75E-05	771.0
E27H1	44	1	50	1.3363883975	1.3353178683	1.3341503446	2/10	3.76E-04	1600.3
E44H1	27	1	50	1.4231184848	1.4229226152	1.4226789226	2/10	7.20E-05	1711.4
E59H1	59	1	50	0.0625113933	0.0624453409	0.0623820199	1/10	2.30E-05	2104.1
E4H0a	4	0	100	0.1145681001	0.1145662184	0.1145628786	2/10	8.41E-07	5491.1
E4H0b	4	0	100	0.4826373801	0.4826356082	0.4826285040	7/10	1.58E-06	5426.4
E6H0	6	0	100	0.1998574337	0.1998560311	0.1998543373	4/10	7.06E-07	4811.3
E9H2	9	2	100	0.7494661035	0.7478403599	0.7460060656	1/10	5.88E-04	8130.4
E11H0	11	0	100	0.7655396868	0.7652491178	0.7648781012	1/10	1.19E-04	5013.9
E12H0a	12	0	100	1.1078034265	1.1077982675	1.1077640795	7/10	5.86E-06	5830.0
E12H0b	12	0	100	0.7755043144	0.7754464078	0.7751726491	2/10	4.76E-05	5436.0
E12H2	12	2	100	0.8146928310	0.8143454056	0.8138911318	2/10	1.61E-04	5445.7
E18H0	18	0	100	1.4913313807	1.4902461911	1.4881580646	1/10	5.01E-04	4922.2
E20H1	20	1	100	0.7639769482	0.7636734678	0.7630778453	1/10	1.33E-04	4749.2
E20H2	20	2	100	1.1842551290	1.1829491272	1.1806416366	1/10	6.61E-04	6246.5
E23H1	23	1	100	0.7209507423	0.7196800678	0.7186048229	1/10	3.35E-04	7233.2
E27H1	27	1	100	0.9087618011	0.9063990417	0.9049753033	1/10	5.31E-04	7206.3
E44H1	44	1	100	0.9655365849	0.9633779411	0.9626444341	1/10	4.36E-04	10230.6
E59H1	59	1	100	0.0424090996	0.0423971176	0.0423592941	1/10	8.59E-06	5309.8

Table 10 Computational results of the BH* algorithm on 45 complicated instances without boundary constraint.

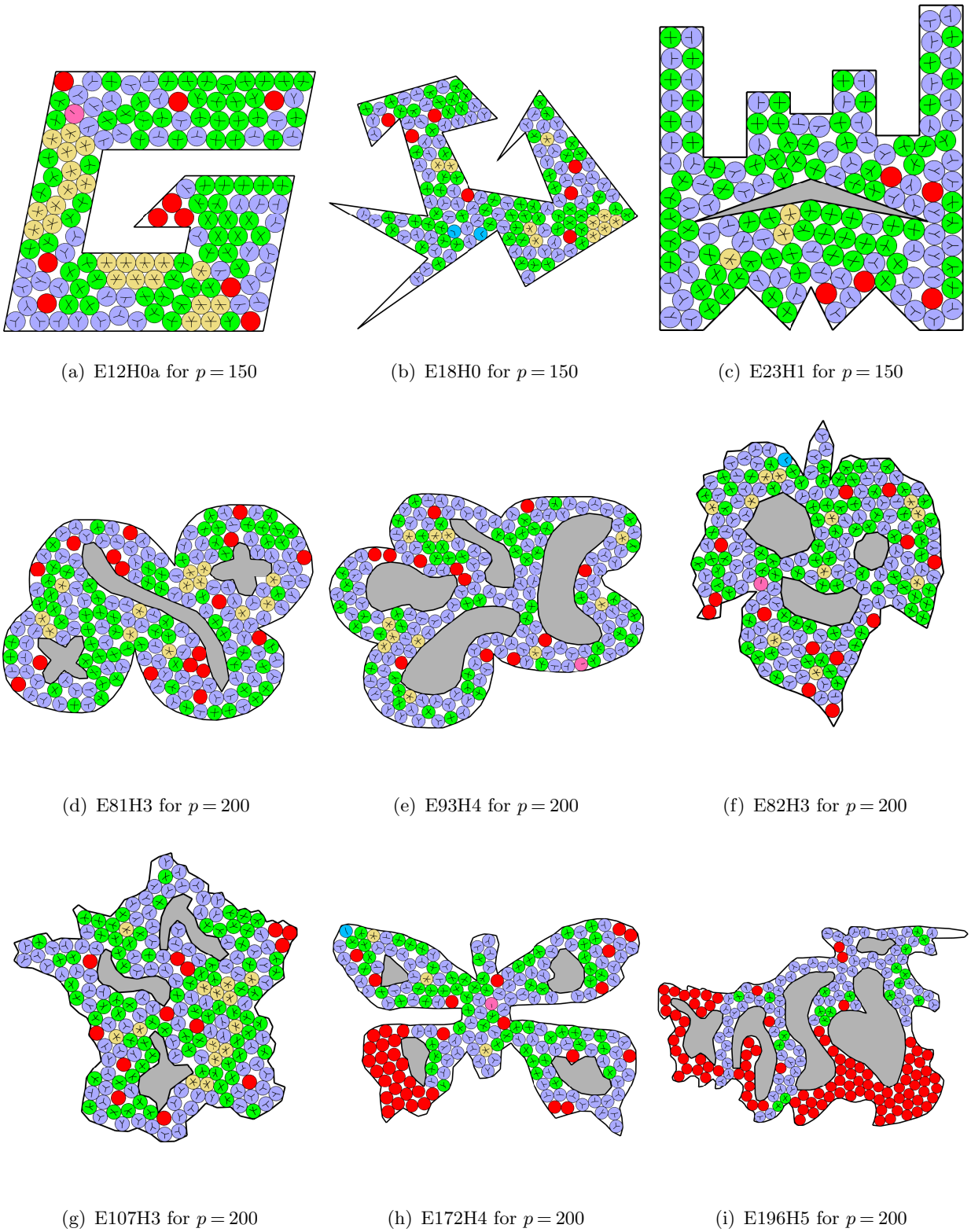
Instance				Basin-hopping					
Container	$\|E\|$	$\|H\|$	p	D_{best}	D_{avg}	D_{worst}	SR	σ	$time(s)$
E101H2	101	2	50	1.9294792183	1.7373305150	0.9582699256	1/10	1.66E-01	3576.2
E101H3	101	3	50	2.7597012147	2.7579227475	2.7542749209	6/10	1.10E-03	1911.0
E106H3	106	3	50	2.2760303651	2.2721291472	2.2656263048	1/10	1.77E-03	1953.8
E106H5	106	5	50	2.1492581495	2.1474806367	2.1443049302	2/10	9.70E-04	2444.3
E107H3	107	3	50	4.3776888646	4.3744222239	4.3665051231	1/10	1.99E-03	2741.0
E120H3	120	3	50	3.3713160251	3.3694128674	3.3688833673	1/10	4.61E-04	2240.0
E172H4	172	4	50	1.6086033153	1.6073993775	1.6058290113	1/10	4.27E-04	2078.2
E193H1	193	1	50	3.4885688530	3.4809669506	3.4737344936	1/10	1.92E-03	1918.0
E196H5	196	5	50	2.1462437211	2.1393500463	2.1258826241	1/10	3.13E-03	2780.3
E203H3	203	3	50	1.2947250241	1.2907711108	1.2829250186	2/10	1.69E-03	2705.9
E60H1	60	1	50	0.5509805177	0.5509455932	0.5508558364	1/10	1.89E-05	1672.9
E81H3	81	3	50	1.6902021702	1.6895559247	1.6879125148	3/10	3.77E-04	2550.2
E82H3	82	3	50	1.5476779639	1.5460645514	1.5432248699	1/10	8.69E-04	2304.9
E84H3	84	3	50	1.6205855793	1.6202744664	1.6199020060	1/10	9.71E-05	2591.9
E93H4	93	4	50	2.4057626982	2.4038957388	2.3985438152	4/10	1.45E-03	2567.8
E101H2	101	2	100	1.2840210583	1.2816485778	1.2806627457	1/10	4.69E-04	10487.4
E101H3	101	3	100	1.8474917504	1.8454387326	1.8427915595	1/10	6.82E-04	9669.7
E106H3	106	3	100	1.5368444673	1.5336969076	1.5314968823	1/10	7.95E-04	7813.6
E106H5	106	5	100	1.4471249445	1.4448386089	1.4406016680	1/10	1.07E-03	8291.4
E107H3	107	3	100	2.9391747678	2.9367746053	2.9334389332	1/10	1.02E-03	7076.3
E120H3	120	3	100	2.2566551682	2.2527411251	2.2471432621	1/10	1.62E-03	8783.9
E172H4	172	4	100	1.0758381251	1.0727718180	1.0704585715	1/10	7.98E-04	10689.5
E193H1	193	1	100	2.3254402936	2.3205631956	2.3149865899	1/10	1.47E-03	9249.4
E196H5	196	5	100	1.4098783673	1.4065769102	1.4037875223	1/10	8.40E-04	10447.4
E203H3	203	3	100	0.8522060778	0.8507183234	0.8489127720	1/10	5.47E-04	5676.0
E60H1	60	1	100	0.3582299427	0.3577676480	0.3571829765	1/10	1.42E-04	7899.8
E81H3	81	3	100	1.1482805892	1.1472277187	1.1441207090	1/10	6.45E-04	3039.9
E82H3	82	3	100	1.0473915944	1.0469480201	1.0458872171	1/10	1.85E-04	2793.5
E84H3	84	3	100	1.0869889031	1.0853519460	1.0841252819	1/10	4.57E-04	8166.1
E93H4	93	4	100	1.5934017314	1.5916298296	1.5899418976	1/10	5.19E-04	7782.2
E101H2	101	2	150	1.0166314840	1.0127636545	0.9996473509	1/10	2.28E-03	22759.0
E101H3	101	3	150	1.4749399669	1.4733512716	1.4693714428	1/10	1.06E-03	7479.9
E106H3	106	3	150	1.2301928882	1.2272249098	1.2247890921	1/10	9.08E-04	10787.9
E106H5	106	5	150	1.1379054440	1.1370513922	1.1363740161	1/10	2.74E-04	12795.7
E107H3	107	3	150	2.3545859328	2.3507354789	2.3482549453	1/10	1.14E-03	19836.0
E120H3	120	3	150	1.7957563567	1.7927535594	1.7881833987	1/10	1.22E-03	18717.6
E172H4	172	4	150	0.8524978245	0.8508764111	0.8487578335	1/10	6.11E-04	21729.9
E193H1	193	1	150	1.8619806126	1.8540394316	1.8453921476	1/10	2.69E-03	23883.0
E196H5	196	5	150	1.1085450824	1.1015314163	1.0944318454	1/10	2.24E-03	22917.5
E203H3	203	3	150	0.6798294129	0.6776296256	0.6765371099	1/10	4.46E-04	11821.8
E60H1	60	1	150	0.2827053641	0.2824164724	0.2822220188	1/10	7.09E-05	14629.7
E81H3	81	3	150	0.9151500861	0.9131879749	0.9121142852	1/10	5.03E-04	15451.9
E82H3	82	3	150	0.8314206381	0.8310173646	0.8303719556	1/10	1.64E-04	16126.1
E84H3	84	3	150	0.8630536076	0.8620981076	0.8612198615	1/10	3.29E-04	17433.8
E93H4	93	4	150	1.2604775170	1.2587040149	1.2561608210	1/10	5.68E-04	13075.9

Table 11 Computational results of the MBH* algorithm on 30 relatively simple instances without boundary constraint.

Instance				Basin-hopping					
Container	$\|E\|$	$\|H\|$	p	D_{best}	D_{avg}	D_{worst}	SR	σ	$time(s)$
E4H0a	4	0	50	0.1664988509	0.1664988509	0.1664988509	10/10	0.00	491.2
E4H0b	4	0	50	0.7232436230	0.7232436230	0.7232436230	10/10	0.00	662.8
E6H0	6	0	50	0.2933779419	0.2933779419	0.2933779419	10/10	0.00	49.5
E9H2	9	2	50	1.1252199453	1.1252199453	1.1252199453	10/10	0.00	841.0
E11H0	11	0	50	1.1330657353	1.1330657353	1.1330657353	10/10	0.00	967.4
E12H0a	12	0	50	1.6364303754	1.6364303754	1.6364303754	10/10	0.00	666.3
E12H0b	12	0	50	1.1512460802	1.1512460802	1.1512460802	10/10	0.00	488.7
E12H2	12	2	50	1.2090410970	1.2090410970	1.2090410970	10/10	0.00	1373.8
E18H0	18	0	50	2.2664436736	2.2664436736	2.2664436736	10/10	0.00	532.7
E20H1	20	1	50	1.1494074146	1.1483431535	1.1475611405	2/10	2.83E-04	1862.2
E20H2	20	2	50	1.7642062920	1.7642062866	1.7642062834	8/10	1.21E-08	381.2
E23H1	23	1	50	1.0945028103	1.0943600422	1.0932145191	8/10	1.92E-04	1985.0
E27H1	27	1	50	1.3363883975	1.3363490513	1.3359949361	9/10	5.90E-05	1938.3
E44H1	44	1	50	1.4231184848	1.4230690380	1.4229607184	3/10	3.49E-05	1990.1
E59H1	59	1	50	0.0625365968	0.0624835536	0.0624554522	1/10	1.61E-05	1866.9
E4H0a	4	0	100	0.1145682248	0.1145671956	0.1145640931	2/10	6.61E-07	6166.4
E4H0b	4	0	100	0.4826373801	0.4826179302	0.4825356355	4/10	1.75E-05	7812.4
E6H0	6	0	100	0.1998574216	0.1997843134	0.1991362773	5/10	1.08E-04	8935.3
E9H2	9	2	100	0.7511238475	0.7502858249	0.7492433888	1/10	2.57E-04	11835.5
E11H0	11	0	100	0.7656897992	0.7655114064	0.7649914495	1/10	9.52E-05	6899.5
E12H0a	12	0	100	1.1078034265	1.1077723402	1.1075502010	6/10	3.75E-05	9443.9
E12H0b	12	0	100	0.7755043105	0.7754653422	0.7754149644	3/10	1.78E-05	7972.2
E12H2	12	2	100	0.8146928235	0.8146115427	0.8142894851	5/10	7.22E-05	5162.5
E18H0	18	0	100	1.4935027529	1.4928152147	1.4904918591	1/10	4.46E-04	9056.9
E20H1	59	1	100	0.7638994850	0.7636264998	0.7630640975	1/10	1.18E-04	6597.8
E20H2	20	1	100	1.1842672680	1.1842389685	1.1841346694	1/10	1.80E-05	5169.2
E23H1	20	2	100	0.7211250210	0.7205693410	0.7200443848	1/10	1.66E-04	8966.1
E27H1	23	1	100	0.9088386862	0.9077689522	0.9067437064	1/10	4.04E-04	8322.5
E44H1	44	1	100	0.9655319844	0.9643480822	0.9628066562	1/10	4.84E-04	10325.3
E59H1	27	1	100	0.0424419006	0.0423749032	0.0421698750	1/10	3.72E-05	6288.4

Table 12 Computational results of the MBH* algorithm on 45 complicated instances without boundary constraint.

Instance				Basin-hopping					
Container	$\|E\|$	$\|H\|$	p	D_{best}	D_{avg}	D_{worst}	SR	σ	$time(s)$
E101H2	101	2	50	1.9296611433	1.9293080575	1.9279676174	5/10	2.57E-04	3019.1
E101H3	101	3	50	2.7597012147	2.7528616370	2.7451484943	3/10	2.86E-03	1982.8
E106H3	106	3	50	2.2785596163	2.2753498323	2.2712160749	1/10	1.16E-03	2144.3
E106H5	106	5	50	2.1503429086	2.1495254107	2.1490434210	3/10	2.72E-04	2619.3
E107H3	107	3	50	4.3776888646	4.3763245600	4.3650783757	8/10	1.88E-03	1951.4
E120H3	120	3	50	3.3713160251	3.3707797719	3.3690613268	1/10	3.04E-04	2321.8
E172H4	172	4	50	1.6083403888	1.6080920152	1.6078112251	4/10	1.19E-04	2604.7
E193H1	193	1	50	3.4930481925	3.4828879612	3.4690733136	1/10	3.40E-03	2139.2
E196H5	196	5	50	2.1458081483	2.1359675846	2.1126245719	1/10	4.32E-03	3044.0
E203H3	203	3	50	1.2963445060	1.2944729939	1.2881415582	4/10	1.26E-03	2854.5
E60H1	60	1	50	0.5509805177	0.5508626165	0.5506094089	2/10	6.40E-05	2174.6
E81H3	81	3	50	1.6902021702	1.6900255476	1.6895113479	6/10	1.17E-04	1541.5
E82H3	82	3	50	1.5477194735	1.5455161767	1.5429605308	1/10	9.53E-04	2494.9
E84H3	84	3	50	1.6205855793	1.6202793679	1.6200263765	2/10	1.01E-04	2228.8
E93H4	93	4	50	2.4087452204	2.4047430949	2.3975061928	1/10	1.49E-03	2178.3
E101H2	101	2	100	1.2839958467	1.2822682646	1.2803151948	1/10	4.82E-04	9696.0
E101H3	101	3	100	1.8517629108	1.8493839200	1.8468545093	1/10	8.89E-04	9336.4
E106H3	106	3	100	1.5389609812	1.5373017679	1.5355478145	1/10	5.62E-04	8820.2
E106H5	106	5	100	1.4471430022	1.4449916914	1.4422214241	1/10	9.16E-04	9055.9
E107H3	107	3	100	2.9422207363	2.9394661810	2.9367264776	1/10	8.60E-04	8926.1
E120H3	120	3	100	2.2585254097	2.2548931409	2.2511126934	1/10	1.23E-03	12440.1
E172H4	172	4	100	1.0767384436	1.0749063077	1.0727082268	1/10	5.74E-04	9578.5
E193H1	193	1	100	2.3343111553	2.3295235036	2.3214644932	2/10	1.97E-03	7944.6
E196H5	196	5	100	1.4138861609	1.4103438073	1.4065984035	1/10	1.23E-03	10583.1
E203H3	203	3	100	0.8564731696	0.8539301996	0.8521302869	1/10	6.78E-04	5641.0
E60H1	60	1	100	0.3582466332	0.3578719449	0.3572269441	1/10	1.39E-04	11181.3
E81H3	81	3	100	1.1485277749	1.1471834164	1.1455728502	1/10	4.88E-04	6745.9
E82H3	82	3	100	1.0474433895	1.0470108082	1.0464577206	2/10	1.59E-04	5507.8
E84H3	84	3	100	1.0872842260	1.0860224488	1.0830364389	1/10	5.83E-04	9933.0
E93H4	93	4	100	1.5939701632	1.5925545584	1.5911251813	1/10	5.50E-04	11966.2
E101H2	101	2	150	1.0155772115	1.0143834682	1.0128444937	1/10	4.04E-04	21128.3
E101H3	101	3	150	1.4759039711	1.4737548666	1.4681973948	1/10	1.03E-03	16911.7
E106H3	106	3	150	1.2307360530	1.2276956567	1.2250162675	1/10	7.08E-04	15194.5
E106H5	106	5	150	1.1372298361	1.1364212055	1.1347835054	1/10	3.69E-04	19205.7
E107H3	107	3	150	2.3577371747	2.3531119086	2.3472667062	1/10	1.61E-03	21671.7
E120H3	120	3	150	1.7975153377	1.7955887570	1.7924087476	1/10	7.07E-04	20227.9
E172H4	172	4	150	0.8525962904	0.8509029887	0.8494290019	1/10	4.90E-04	17591.8
E193H1	193	1	150	1.8620687888	1.8561542253	1.8499537490	1/10	2.10E-03	20907.5
E196H5	196	5	150	1.1096023844	1.1007478968	1.0851449447	1/10	3.46E-03	21476.2
E203H3	203	3	150	0.6801848429	0.6786975696	0.6774633897	1/10	4.19E-04	19141.6
E60H1	60	1	150	0.2828959343	0.2826403084	0.2823869775	1/10	6.12E-05	22588.3
E81H3	81	3	150	0.9155216189	0.9141138269	0.9134880776	1/10	2.96E-04	21073.8
E82H3	82	3	150	0.8324378837	0.8314429821	0.8305306328	1/10	2.79E-04	18772.3
E84H3	84	3	150	0.8629312866	0.8618367930	0.8607773850	1/10	3.75E-04	23571.1
E93H4	93	4	150	1.2601949020	1.2588156611	1.2567688378	1/10	5.40E-04	22484.4

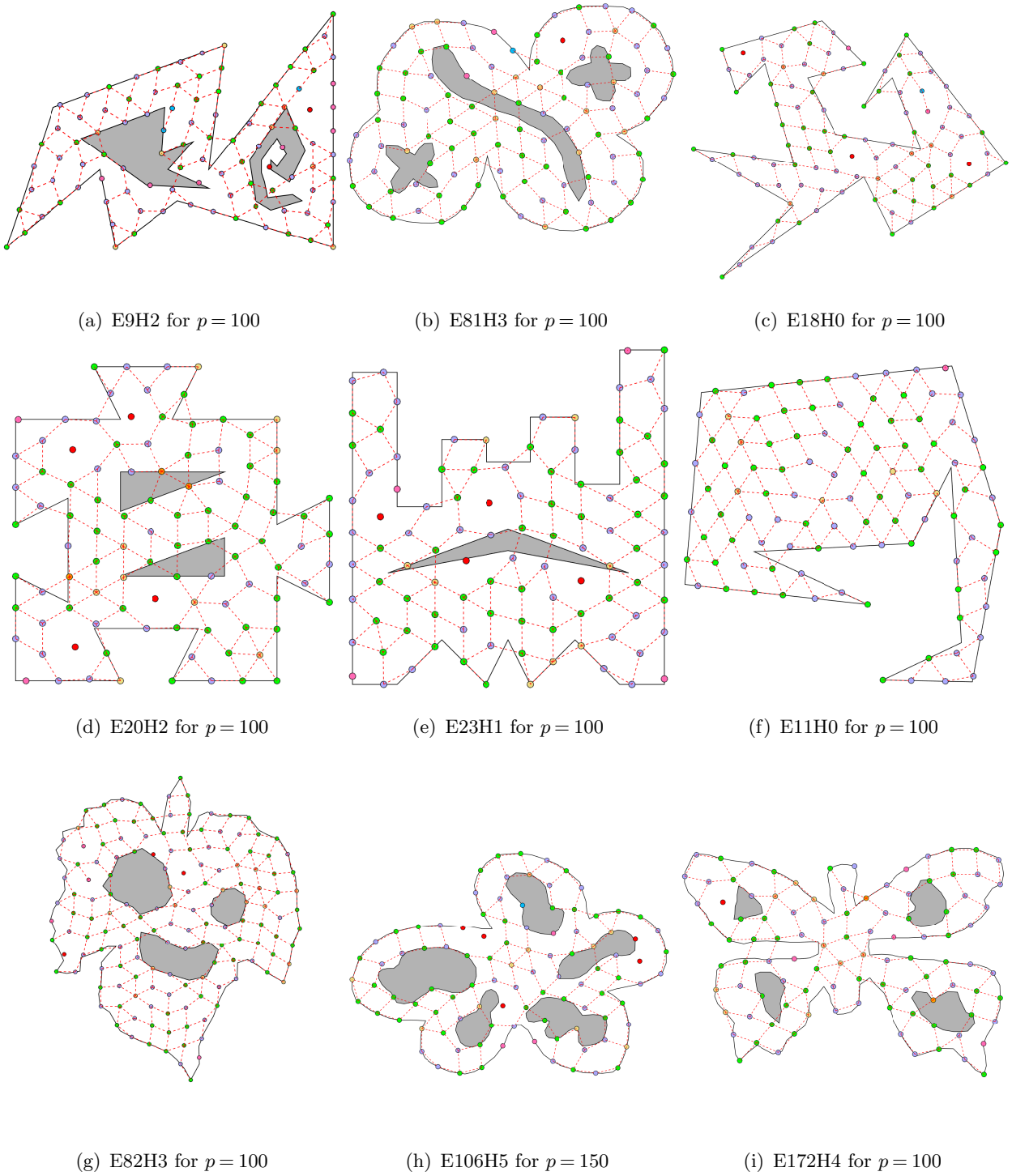


(a) E12H0a for $p = 150$ (b) E18H0 for $p = 150$ (c) E23H1 for $p = 150$

(d) E81H3 for $p = 200$ (e) E93H4 for $p = 200$ (f) E82H3 for $p = 200$

(g) E107H3 for $p = 200$ (h) E172H4 for $p = 200$ (i) E196H5 for $p = 200$

Figure 5 **Best configurations found in this work for other representative instances, where the circles are colored according to their number of neighbors.**



(a) E9H2 for $p = 100$ (b) E81H3 for $p = 100$ (c) E18H0 for $p = 100$

(d) E20H2 for $p = 100$ (e) E23H1 for $p = 100$ (f) E11H0 for $p = 100$

(g) E82H3 for $p = 100$ (h) E106H5 for $p = 150$ (i) E172H4 for $p = 100$

Figure 6 **Best configurations found in this work for other representative instances, where two points are connected by a dotted line if the distance between them equals the minimum distance found D_{best}.**