




Near optimal minimal convex hulls of disks

Josef Kallrath¹ · Joonghyun Ryu² · Chanyoung Song³ · Mokwon Lee² ·
Deok-Soo Kim^{2,3} 

Received: 3 July 2020 / Accepted: 20 February 2021 / Published online: 19 March 2021
© The Author(s) 2021

Abstract

The minimal convex hulls of disks problem is to find such arrangements of circular disks in the plane that minimize the length of the convex hull boundary. The mixed-integer non-linear programming model, named *MinPerim* [17], works only for small to moderate-sized problems. Here we propose a polyolithic framework of the problem for big problem instances by combining the following algorithms and models: (i) A fast disk-packing algorithm *VOROPACK-D* based on Voronoi diagrams, non-linear programming (NLP) models for packing disks, and an NLP model *minDPCH* for minimizing the discretized perimeter of convex hull; (ii) A fast convex-hull algorithm *QuickhullDisk* to compute the convex hulls of disk arrangements and their perimeter lengths; (iii) A mixed-integer NLP model *MinPerim* taking the output of *QuickhullDisk* as its input. We present complete analytic solutions for small problems up to four disks and a semi-analytic mixed-integer linear programming model which yields exact solutions for strip packing problems with up to one thousand congruent disks. It turns out that the proposed polyolithic approach works fine for large problem instances containing up to 1,000 disks. Monolithic and polyolithic solutions using *minDPCH* usually outperform other approaches. The polyolithic approach yields better solutions than the results in [17] and provides a benchmark suite for further research.

✉ Joonghyun Ryu
jhryu@hanyang.ac.kr

✉ Deok-Soo Kim
dskim@hanyang.ac.kr

Josef Kallrath
jkallrath@ufl.edu; josef.kallrath@web.de

Chanyoung Song
cysong.vdrc@gmail.com

Mokwon Lee
mwlee.vdrc@gmail.com

¹ Department of Astronomy, University of Florida, Gainesville, FL, USA

² Voronoi Diagram Research Center/HYU-HPSTAR-CIS High Pressure Research Center, Hanyang University, Seoul, Korea

³ School of Mechanical Engineering, Hanyang University, Seoul, Korea

Keywords Convex hulls · Global optimization · Non-convex nonlinear programming · Polyhedral · Voronoi diagram · VOROPACK-D · QuickhullDisk

1 Introduction

The **minimal convex hull of disks problem** in the plane, hereafter referred to the **minimal convex hull problem**, is to find arrangements of a finite set of 2D circular disks in the plane so that the perimeter length of the convex hull of the disks is minimized. The disks, either congruent or non-congruent, are placed in the arrangements in such a way that they do not overlap each other although two disks may contact. Some problems in logistics, such as the container loading [4] and placement of cylindrical drums on trucks [17], require computing minimal convex hulls of disks. Instead of packing objects into a container of explicitly specified shape such as rectangle and circle, the container in this study is implicitly defined: It is the convex hull of the objects.

The minimal convex hull problem was formally introduced by [17], hereafter abbreviated as KF19, and that of minimal convex hulls of 3D spheres by [16]. The problem was solved by formulating a mixed-integer non-linear programming (MINLP) model named MinPerim and approached from the deterministic global optimization point of view. However, as a MINLP problem is NP-hard, the MinPerim model can only be solved for small problem instances. In this paper, we want to find near optimal solutions of the minimal convex hull problems of considerably large problem instances using a *polyhedral* approach by combining techniques from both mathematical programming and computational geometry. The motivation is to combine the advantages of both disciplines: computational efficiency of computational geometry techniques and solution quality of mathematical programming. Two computational geometry algorithms are employed: VOROPACK-D for packing a set of disks [35,39] and QuickhullDisk for constructing the convex hulls of the arrangements of disks [26,33].

The idea of this study is simple as follows (Fig. 1). Given a large set of input disks, it first makes an initial disk arrangement by solving the disk packing problem with VOROPACK-D or an NLP model (in Step I) and constructs its convex hull with QuickhullDisk (in Step III). The convex hull is then used to build an MINLP model to minimize the perimeter of the convex hull boundary (in Step IV). If the input disk set is of a small-to-moderate size, Step I is followed by an improvement step (in Step II) with an NLP model based on the discretization of the boundary of an approximated convex hull (in Step II) before reaching Step III. The discretization is based on a finite set of rays emanating from a common interior point of the convex hull. The length of each ray, *i.e.*, the terminating point on each ray, follows from the tangential condition that all disks are “below” that tangential line, *i.e.*, the union of all tangentials provide an outer approximation of the boundary of the convex hull.

Contributions

The major contributions and highlights of this paper are three-fold.

1. A combined view of solving the minimal convex hull problems from both computational geometry and mathematical programming for problem instances containing up to 1,000 disks:
 - Polyhedral approaches computing the arrangement of non-overlapping disks based on VOROPACK-D or non-linear programming (NLP).
 - Output of VOROPACK-D providing input for QuickhullDisk, possibly through minDPCH (minimum discretized perimeter of convex hull).

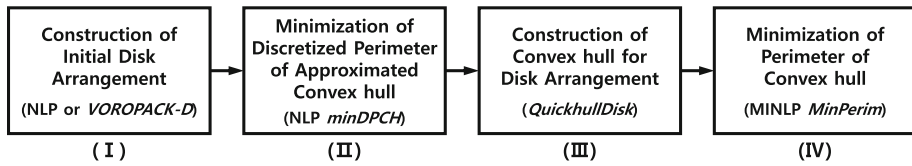


Fig. 1 Overview of the proposed polyolithic approach. Given a set of input disks, it first makes an initial disk arrangement (in Step I) which may be used to improve the solution by solving an NLP model to minimize a discretized perimeter of the approximated convex hull (in Step II). The convex hull of the output from Step II is constructed (in Step III) which is then used to build an MINLP model to minimize the convex hull boundary. The NLP model of Step II can work without an initial disk arrangement in Step I. For small to mid-sized problems, this could be more efficient

- Output of `QuickhullDisk` providing initial values for an MINLP model `MinPerim`.
- 2. Introducing a novel NLP model `minDPCH` which provides initial values for `MinPerim` via `QuickhullDisk`.
- 3. Analytic solutions for smaller cases up to four non-congruent disks and semi-analytic solutions involving a mixed-integer linear programming (MILP) model to solve strip packing problems with congruent disks.

The term **polyolithic** has been coined by [13,14] to refer to tailor-made modeling and solution approaches to solve optimization problems exploiting several models where output of one is input to another. Hence, polyolithic methods consist of a set of models which are linked to each other with regard to their inputs and outputs, *i.e.*, model $i+1$ can use the results of the first i models. This can be used, for instance, to initialize variables or to put tighter bounds on variables. On the other hand, a **monolithic** model consists of a model with data, variables and constraints and a call to a solution algorithm to solve optimization problems, *i.e.*, one model with one solve statement. This keeps the structure of the model and its solution relatively simple and clear. Let \mathcal{C} be a **container** hosting all input disks and \mathcal{H} be the convex hull of the input disks. Suppose that \mathcal{C} is convex. Then, $\mathcal{H} \subseteq \mathcal{C}$. Circular, elliptic, oval, and rectangular containers are typical examples of such convex containers. We call a container a **design container** if we seek a disk arrangement which determines the parameters of the container in such a way that an objective function is minimized. The area of the container is an example objective function. This optimization problem could be constrained by a **target domain** Ω in which both the disks and the container have to fit. Hence, $\mathcal{C} \subseteq \Omega$. Usual target domains are also of a convex shape such as circular, elliptic, oval, and rectangular. In this paper, both *disk* and *circle* denote a circular object and its boundary, respectively.

Two computational geometry algorithms are used to take advantage of the geometric nature of the problem, particularly for large problem instances. First, the `VOROPACK-D` algorithm packs circular disks in a container of circular or rectangular shape [35]. There were many studies on packing and cutting problems with mathematical programming and/or heuristic methods [6,8,11,27,34,37,41]. This study uses `VOROPACK-D` which can quickly find sufficiently good solutions for big problem instances. `VOROPACK-D` takes an argument denoting the container shape: `CC` or `RC` for a circular or rectangular container, respectively. Hence, with a slight abuse of notation, `VOROPACK-D(CC)` and `VOROPACK-D(RC)` pack input disks in a circular and rectangular container, respectively. Note that `VOROPACK-D(CC)` actually executes the `Shrink&Shake` algorithm which shrinks a sufficiently big circular container and shakes the disks intersecting the container, if any, by repositioning in the container [35,39]. Taking advantage of the powerful spatial reasoning property

of the Voronoi diagram of disks, Shrink&Shake can quickly reach a local optimum. `VOROPACK-D(RC)` uses a similar algorithm. Secondly, the `QuickhullDisk` algorithm constructs the convex hulls of disk arrangements by adapting the idea of the well-known quick sort algorithm [26]. `VOROPACK-D` and `QuickhullDisk` are freely available as both web servers and standalone programs (for both Linux and MS Windows) from the Voronoi Diagram Research Center, Hanyang University (<http://voronoi.hanyang.ac.kr/voropack>, <http://voronoi.hanyang.ac.kr/quickhulldisk>).

The remainder of the paper is organized as follows. Section 2 introduces the ordinary and minimal convex hull of disks and their related notations. Section 3 presents the non-linear programming basis for the minimal convex hull problems. Section 4 derives analytic solutions for small problems up to four disks and provides a semi-analytic mixed-integer linear programming model for a special strip packing problem. Section 5 presents theoretical bounds and gaps. Section 6 presents the proposed polyhedral framework and Sect. 7 its validation with numerical experiments. Section 8 concludes this paper.

2 The ordinary and minimal convex hulls of disks

Let $\mathbf{x} \in \mathbb{R}^2$ be a column vector and \mathbf{x}^T be its transpose. The two dimensions of the plane are referenced by $d \in \mathcal{D} = \{1, 2\}$, where 1 and 2 represent the first (x -axis) and second dimension (y -axis), respectively. Let \mathcal{I} be a finite set of n disks in the plane where the center of each disk $i \in \mathcal{I}$ is $\mathbf{x}_i^0 = (x_{i1}, x_{i2})^T$ and its radius R_i . Two coordinate frames are employed depending on the situation. The first one uses only the positive quadrant with vectors $\mathbf{x} \in \mathbb{R}^2$, $\mathbf{x} \geq \mathbf{0}$. On the other hand, in the second one, we enforce

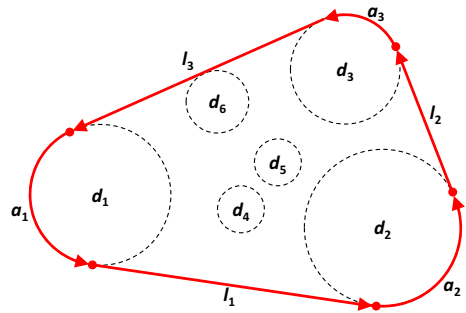
$$\mathbf{x}_c = \frac{1}{\sum_i R_i} \sum_i R_i \mathbf{x}_i^0 = \mathbf{0}. \quad (2.1)$$

This implies that \mathbf{x}_c coincides the origin of the coordinate system and relaxes the non-negativity constraint. Given two points $\mathbf{x}_1 = (x_{11}, x_{12})^T$ and $\mathbf{x}_2 = (x_{21}, x_{22})^T$ in the plane, the distance between \mathbf{x}_1 and \mathbf{x}_2 in this study is defined by the L_2 -norm, *i.e.*, the ordinary Euclidean distance, $\|\mathbf{x}_1 - \mathbf{x}_2\|_2 = \sqrt{\sum_{d \in \mathcal{D}} (x_{1d} - x_{2d})^2}$.

Suppose that a convex hull boundary of a set of disks is represented by $\partial\mathcal{H} = \{a_1, l_1, a_2, l_2, \dots, a_M, l_M\}$, where a_i and l_i denote an arc and a line segment on the boundary, respectively. As illustrated in Fig. 2, $\partial\mathcal{H}$ is counterclockwise oriented and thus every arc and line segment are also accordingly oriented. Particularly, each arc $a_i \in \partial\mathcal{H}$ is counterclockwise oriented around its center. According to the convex hull condition, arcs and line segments tangentially contact if they contact. An oriented line segment $l_j \in \partial\mathcal{H}$ is represented by an ordered tuple $(\mathbf{s}_j, \mathbf{e}_j)$ for the start and end vertices, respectively. Note that \mathbf{s}_j and \mathbf{e}_j are points on two adjacent disks on $\partial\mathcal{H}$. Let d_k be a disk which contributes to $\partial\mathcal{H}$. An oriented arc a of disk d_k is represented by an ordered triplet $(\mathbf{x}_k^0, \mathbf{s}_k, \mathbf{e}_k)$ of the disk center, the start and end vertices of a on ∂d_k , respectively.

Suppose that a subset $\mathcal{I}^{\text{out}} \subseteq \mathcal{I}$ of disks contributes to $\partial\mathcal{H}$. Then, the disks in \mathcal{I}^{out} are called outer disks (or extreme disks), while all other disks are inner disks (non-extreme disks). In Fig. 2, d_1, d_2 , and d_3 are outer (or extreme) disks and d_4, d_5 , and d_6 are inner (or non-extreme) disks. The arcs a_1, a_2 , and a_3 contribute to the convex hull boundary. Note that all entities are oriented. If $|\mathcal{I}^{\text{out}}| = k$ and $|\mathcal{I}| = n$, $n - k$ disks are either in the interior of \mathcal{H} or just touch $\partial\mathcal{H}$ from the inside of \mathcal{H} . A disk touching $\partial\mathcal{H}$ is not an outer disk but an inner one.

Fig. 2 Illustration of a convex hull of six disks. $d_1, d_2,$ and d_3 are outer (or extreme) disks and $d_4, d_5,$ and d_6 are inner (or non-extreme) disks. Be aware of the orientations. $a_1, a_2,$ and a_3 are arcs and $l_1, l_2,$ and l_3 are line segments contributing to the convex hull boundary



In a general setting, a disk d can contribute more than one arc to $\partial\mathcal{H}$. Consider the case that tiny disks are placed around a big one in the center in such a way that the tangential line between each tiny one and the big one is a supporting hyperplane of the entire disk set. In this case, given n tiny disks, the big disk can contribute to the convex hull boundary with up to n arcs and in such a case, there are $2n$ line segments on \mathcal{H} . In Fig. 3a, we have a big disk, say d_1 , in the middle and two small disks, say d_2 and d_3 , are tangentially placed around d_1 in such a way that d_2 and d_3 are antipodal.

However, in the minimal convex hull, a disk contributes to $\partial\mathcal{H}$ at most once if a domain Ω does not constrain the placement of disks. See Fig. 3b: d_2 and d_3 are clustered together. It is easy to prove that the length of $\partial\mathcal{H}$ in Fig. 3b is shorter than that in Fig. 3a. In addition, it is not difficult to prove that the placement in Fig. 3b is the minimal convex hull. This observation extends to an arbitrary number of tiny disks. For details, see KF19 [17]. Hence, minimal perimeter length convex hulls have outer disks which contribute *one and only one* arc to the convex hull boundary. Be aware that, however, there are alternative solutions as the rotation of the arrangement in Fig. 3b with an arbitrary angle around the center of the big disk d_1 results in an identical length.

Suppose that there is a constraint such that disks can be placed within a **rectangular domain** Ω as shown in Fig. 3c. Here we have a big disk d_1 in the center of Ω so that the disk is inscribed in Ω . Suppose that $\partial d_1 \cap \partial\Omega$ yields four distinct points, *i.e.*, the center of d_1 is equidistant from $\partial\Omega$ and $\partial\Omega$ is in fact a square. Suppose that we have four more disks $d_2, d_3, d_4,$ and d_5 , where no two disks can be placed in a single corner of Ω without violating the non-overlapping constraint, *i.e.*, unless the interior of two disks have a non-empty intersection. In such a case, d_1 contributes to the minimal convex hull boundary with four arcs. Hence, it can be easily proved that a disk d can contribute to the minimal convex hull boundary with M arcs if and only if $\partial d \cap \partial\Omega$ has M points. Figure 3d shows that the placement of two non-clustered tiny disks in a single corner cannot be the solution of the minimal convex hull. Here the same reasoning in Fig. 3a and b holds. This discussion results in the following observation: If a disk d has M tangential contacts with a domain Ω , d contributes to $\partial\mathcal{H}$ with M arcs.

3 Non-linear programming for the minimal convex hull problems

Sections 3.1 and 3.2 discuss an NLP model called minDPCH to minimize the discretized perimeter of an approximated convex hull. The output of minDPCH is used to construct the convex hull using QuickhullDisk algorithm which is then used to build an MINLP model

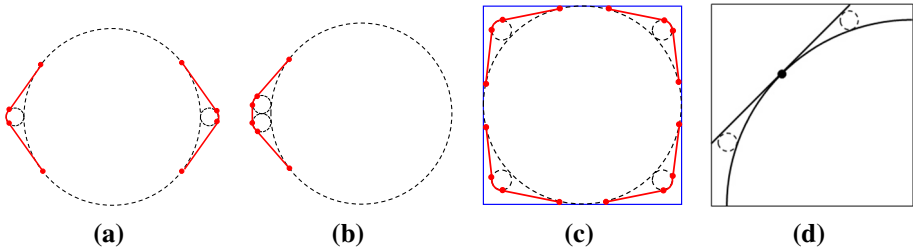


Fig. 3 The number of arcs that a disk can contribute to the convex hull boundary. **a** Two tiny disks, not clustered. The convex hull is not minimal and the big disk contributes to the convex hull boundary with two arcs. **b** Two tiny disks clustered together. The convex hull is minimal and every disk contributes to the convex hull boundary with one arc. **c** The domain Ω is rectangular. There are four disconnected regions at the corners where each can host only one small disk. The big disk (d_1) contributes to the boundary of minimal convex hull with four arcs. Note that $\partial d_1 \cap \partial \Omega$ has four points. **d** Two tiny disks separated by a tangential point around a corner. The convex hull is not minimal. In this case the big disk contributes to the convex hull boundary with an additional arc between the two tiny disks

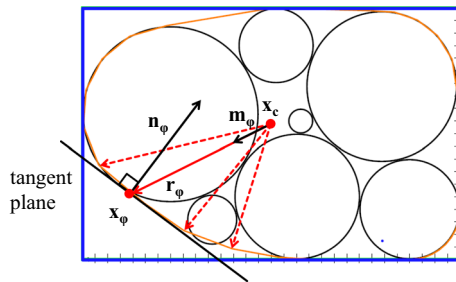


Fig. 4 The geometric idea of minDPCH. A coordinate origin \mathbf{x}_c is defined as the averaged radius-weighted center using Eq. (2.1). Then a set of points are sampled using polar coordinates. Given coordinate origin \mathbf{x}_c , a uniform grid of unit direction vectors \mathbf{m}_φ is defined onto the tangent line at point \mathbf{x}_φ using polar coordinates. $\mathbf{r}_\varphi = r\mathbf{m}_\varphi$ is the radial vector which corresponds to \mathbf{m}_φ . \mathbf{n}_φ is a normal vector of tangent plane at the point \mathbf{x}_φ which corresponds to \mathbf{r}_φ . Note that the orange curve is not the convex hull perimeter but an inner approximation

to minimize the convex hull boundary. Section 3.3 presents various NLP models to initialize minDPCH for larger problem instances.

3.1 The idea of minDPCH

Similarly to the idea developed in [16] using polar coordinates, we cover the perimeter of convex hull by a grid of points distributed uniformly on the boundary $\partial\mathcal{H}$ of the convex hull. Suppose that we define a coordinate origin as the averaged radius-weighted center \mathbf{x}_c using Eq. (2.1). Over the angular index domain φ we generate a uniform grid of unit direction vectors \mathbf{m}_φ using \mathbf{x}_c . Then each point is sampled considering the radial distance from \mathbf{x}_c as shown in Fig. 4.

To each \mathbf{m}_φ we associate a non-negative variable r_φ and we describe $\partial\mathcal{H}$ based on this polar coordinate $\mathbf{x}_\varphi = (r_\varphi, \varphi)$. The points on $\partial\mathcal{H}$ are subject to the condition that the distance of all disk centers \mathbf{x}_i^0 is greater or equal to their radii,

$$\mathbf{n}_\varphi^T \mathbf{x}_i^0 - n^D \geq R_i, \tag{3.1}$$

where the normal vector \mathbf{n}_φ and origin-distance n^D describe the tangent plane at \mathbf{x}_φ ,

$$\mathbf{n}_\varphi^T \mathbf{x}_\varphi = n^D. \tag{3.2}$$

Note that the angle between \mathbf{n}_φ and \mathbf{m}_φ has to be in the range of 90 and 180 degrees, or

$$\mathbf{n}_\varphi^T \mathbf{m}_\varphi \leq 0, \tag{3.3}$$

which means that the normal vector of the tangential plane points into the interior of the convex hull. We minimize the line integral $\int_0^{2\pi} r_\varphi d\varphi$, or its discretized version

$$\sum_{\nu=1}^{N_\varphi} r_{\varphi_\nu} \Delta\varphi, \tag{3.4}$$

with N_φ equidistant φ -angles, the φ -increments $\Delta\varphi = \frac{2\pi}{N_\varphi}$, and the φ -grid points $\varphi_\nu = (\nu - \frac{1}{2}) \Delta\varphi$. To keep the non-convex NLP problem computationally tractable, we maintain the total number of grid points at a reasonable level of no more than forty points. However, if we want to integrate only the unit disk (i.e., $r_{\varphi_\nu} = 1$), we need about a hundred grid points to obtain the approximate value of 6.2831853 for 2π . The approach has been implemented and works for up to two hundred disks. Beyond this size we cannot find feasible points. In such cases, we compute initial arrangements of disks using the disk packing models in Sect. 3.3.

Sampling of grid points For convex hulls with near-circular boundaries (with no target domain), the equidistant angular grid is fine. However, for rectangular target domain, it is more efficient to use non-equidistant angular grid. We have applied a few individual numerical tests and see an improved efficiency due to the smaller number of grid points. However, due to the complexity and various technical complications (distribution of the grid points, filling degree of the rectangle, smoothing effects in the non-zero curvature part of the convex hull boundary), we have not systematically followed this track.

3.2 NLP formulation of minDPCH

The ideas in Sect. 3.1 yield the following intuitive NLP formulation. The key decision variables are the center coordinates $\mathbf{x}_i^0 = (x_{i1}, x_{i2})^T \in \mathbb{R}^2$ of the disks. Consider the disks $i, j \in \mathcal{I}$ with radii R_i and R_j , where $R_i \geq R_j$ for $i > j$. The non-overlap condition for disks i and j produces the following non-convex constraints.

$$\|\mathbf{x}_i^0 - \mathbf{x}_j^0\|_2^2 = \sum_{d \in \mathcal{D}} (x_{id}^0 - x_{jd}^0)^2 \geq (R_i + R_j)^2, \quad \forall \{(ij) | i < j\}. \tag{3.5}$$

Note that there are $n(n - 1)/2$ inequalities of type Eq. (3.5) for n disks. For each direction vector \mathbf{m}_φ with its center at \mathbf{x}_c using Eq. (2.1), we seek the value of the non-negative variable r_φ which describes the boundary $\partial\mathcal{H}$ of the convex hull based on the polar coordinate $\mathbf{x}_\varphi = (r_\varphi, \varphi)$. The convex hull vector points are subject to the condition that the distances of disk centers \mathbf{x}_i^0 is greater or equal to their radii using the constraints (3.1), (3.2), and (3.3). We minimize the discretized perimeter length $\ell_D = \sum_{\nu=1}^{N_\varphi} \|\mathbf{x}_\varphi - \mathbf{x}_{\varphi+1}\| \Delta\varphi$ of $\partial\mathcal{H}$ in Eq. (3.4).

The structure of the problem This NLP model contains bilinear and square root relations. It does not provide strong lower bounds. The only strict lower bound we can provide is the radius of the smallest disk: $r_\varphi \geq \min_i R_i, \forall \varphi$.

Symmetry and optimality Symmetry is a problem when using deterministic global solver and trying to close the **gap** between the upper and lower bounds, and thus proving global optimality. Therefore, we want to reduce the observed symmetries: translational, rotational, and mirror symmetry. We can partially reduce translational symmetry by fixing \mathbf{x}_c . We can destroy these symmetries by selecting the coordinate frame 2 without fixing \mathbf{x}_c and instead placing the disk 1 at the origin to break translational symmetry, i.e., $\mathbf{x}_1 = 0$. We place disk 2 on the positive x_2 -axis such that $x_{21} = 0, x_{22} \geq 2R_1 + R_2$ to destroy rotational symmetry. We break mirror symmetry by requesting the third disk to be placed above the x_1 -axis, i.e., $x_{32} \geq 0$. This approach helps us to at least solve small instances with only congruent disks to optimality when we use MinPerim directly and only minimize the sum ℓ_L of line segments. However, there is always a trade-off with deterministic global solvers. Without symmetry reducing techniques, i.e., with fewer constraints, they find better initial solutions in shorter time. Symmetry reducing techniques only pay out when one wants to close the gap, which is usually possible only for smaller problems.

3.3 NLP models for disk packing

This section discusses the NLP models for disk packing arrangements to initialize either minDPCH or MinPerim. The disk packing arrangements have to satisfy two constraints: i) the non-overlap constraint and ii) if a rectangular target domain Ω is given, all input disks should fit into Ω . The non-overlap constraints for disks i and j with arbitrary radii R_i and R_j correspond to Eq. (3.5). For congruent disks, we add the symmetry breaking inequality

$$x_{i1}^0 \leq x_{j1}^0, \quad \forall \{(i, j) | i < j\}. \tag{3.6}$$

Fitting the disks inside the rectangular target domain requires

$$x_{id}^0 \geq R_i, \quad \forall \{i, d\} \tag{3.7}$$

and

$$x_{id}^0 + R_i \leq x_d^p \leq E_d, \quad \forall \{i, d\}. \tag{3.8}$$

E_d specifies the length ($d = 1$) and width ($d = 2$) of the rectangle. x_d^p is the free length and width of the rectangle if the rectangle is considered as a design container whose area or length of perimeter is to be minimized. Inequality (3.7) assumes that the rectangular container has its lower-left corner at the origin.

Inequalities in Eqs. (3.7) and (3.8) will only be used in the coordinate frame 1; in this case all disks are hosted in a target domain located in the first octant; $\mathbf{x} \geq 0$. The model established by Eqs. (3.5), (3.6), (3.7) and (3.8) is called CutDisks which uses either the area $a = x_1^p x_2^p$ or perimeter length $\ell_R = 2(x_1^p + x_2^p)$ of the rectangle hosting the disks. If we want to fit the disks into a circular container of minimal radius $r_{\min R}^{cc}$, it is better to use the coordinate frame 2 with $-\infty \leq \mathbf{x} \leq +\infty$. For practical reasons one tries to locate the radius-weighted center \mathbf{x}_c of the disks near or in the origin of the coordinate system as discussed in Eq. (2.1). The condition for fitting all disks into the circular container of radius $r_{\min R}^{cc}$ is

$$\|\mathbf{x}_i^0\|_2 + R_i = \sqrt{\sum_d (x_{id}^0)^2} + R_i \leq r_{\min R}^{cc}. \tag{3.9}$$

Rotational symmetry is broken by placing disk 1 in the first quadrant, i.e., $\mathbf{x}_1 \geq 0$. The model established by Eqs. (3.5), (3.6), (3.9), and $\mathbf{x}_1 \geq 0$ is called minRadiusCC; we use it

Table 1 Summary of non-linear programming models

Model	Type	Objective(s) to be minimized	References
minDPCH	NLP	Length of the discretized perimeter $\partial\mathcal{H}$	Section 3.2
CutDisks	NLP	Area or perimeter of a rectangle	Section 3.3
minRadiusCC	NLP	Radius of a circular container	Section 3.3
minSDC	NLP	Sum of distances of disk centers to \mathbf{x}_c	Section 3.3
MinPerim	MINLP	Length of the perimeter $\partial\mathcal{H}$	KF19

CutDisks, minRadiusCC, and minDPCH is used for minimizing the discretized perimeter of convex hulls. MinPerim is used for the constructing minimal convex hulls by taking the output of other models as its input

to compute disk arrangements with minimal $r_{\min R}^{cc}$. It produces good disk arrangements for computing minimal convex hulls of a large number of disks.

In the coordinate frame 2, we also consider a packing problem in which we minimize the sum of distances from all disks to the center \mathbf{x}_c . That model with (3.5), (3.6) and the objective function

$$\min \sum_i \|\mathbf{x}_i^0 - \mathbf{x}_c\|_2 \tag{3.10}$$

is called minSDC for a circular container. minSDC can be applied to the rectangular domain using the additional constraints (3.7) and (3.8). The MINLP and NLP models for polyhithic approaches of Sect. 7 are summarized in Table 1.

4 Analytic and semi-analytic solutions

In this section, we provide various analytic solutions and a semi-analytic solution based on an MILP model to solve the examples in Sect. 7.2.2 to global optimality within seconds—even for instances up to one thousand disks. For the evaluation of numerical experiments in Sect. 7, it helps us to compare the numerical results to analytic solutions.

4.1 Analytic methods for optimal solutions

4.1.1 Two disks

Suppose that two disks with radii R_1 and R_2 , $R_1 \geq R_2$, are given. We find the disk sector angles, α_1 and α_2 , in KF19 as

$$\alpha_1 = 2\pi - 2 \arccos \frac{R_1 - R_2}{R_1 + R_2}, \quad \alpha_2 = 2 \arccos \frac{R_1 - R_2}{R_1 + R_2} \tag{4.1}$$

or, alternatively,

$$\alpha_{1,2} = \pi \pm 2 \arccos \frac{2\sqrt{R_1 R_2}}{R_1 + R_2}.$$

Thus the length ℓ_2 of $\partial\mathcal{H}$ of two touching disks is given by

$$\ell_2 = \ell_{L2} + \ell_{A2} = 4\sqrt{R_1 R_2} + R_1\alpha_1 + R_2\alpha_2. \tag{4.2}$$

Note that $R_2 = R_1 = R$ implies that $\alpha_2 = \alpha_1 = \pi$, *i.e.*, the contributed length ℓ_{A2} of the arcs is $2\pi R$ as geometrically expected.

4.1.2 Three disks

For three disks with radii R_1, R_2 and $R_3, R_1 \geq R_2 \geq R_3$, we need to distinguish two cases to compute the length of the perimeter ℓ_3 : In case 1, the radius R_3 of the smallest disks is so small that this disk does not contribute an arc to ∂S (they may touch $\partial \mathcal{H}$); in this case we have $\ell_3 = \ell_2$, where ℓ_2 depends only on R_1 and R_2 .

In case 2, all three disks contribute an arc to $\partial \mathcal{H}$ and establish the tour 1-2-3-1. The minimal sum of the lengths of the line segments is given by

$$\ell_{L3} = 2 \left[\sqrt{R_1 R_2} + \sqrt{R_2 R_3} + \sqrt{R_3 R_1} \right]. \tag{4.3}$$

To calculate the contribution of the three arcs, we need the sector angles α_i . As displayed in Fig. 5, they can be obtained as

$$\alpha_i = 2\pi - \bar{\alpha}_i - \theta_{i,i-1} - \theta_{i,i+1},$$

where $\bar{\alpha}_i$ is the inner triangle angle (opposite to the sector angle α_i) corresponding to α_i . Those angles $\bar{\alpha}_i$ are established by the center coordinates of the disks $i, i - 1$, and $i + 1$, or, equivalently, sides of size $R_1 + R_2, R_2 + R_3$, and $R_3 + R_1$. The angles $\theta_{i,i-1}$ and $\theta_{i,i+1}$ are the trapezoid angles at the center of disk i to the adjacent disks $i - 1$ and $i + 1$ (similar to Fig. 4 of KF19) obtained by

$$\theta_{i,i-1} = \arccos \frac{R_i - R_{i-1}}{R_i + R_{i-1}}, \quad \theta_{i,i+1} = \arccos \frac{R_i - R_{i+1}}{R_i + R_{i+1}},$$

in detail

$$\begin{aligned} \theta_{13} &= \arccos \frac{R_1 - R_3}{R_1 + R_3}, & \theta_{31} &= \pi - \theta_{13}, \\ \theta_{12} &= \arccos \frac{R_1 - R_2}{R_1 + R_2}, & \theta_{21} &= \pi - \theta_{12}, \\ \theta_{23} &= \arccos \frac{R_2 - R_3}{R_2 + R_3}, & \theta_{32} &= \pi - \theta_{23}. \end{aligned}$$

The angles $\bar{\alpha}_i$ – their sum adds up to π – are given as

$$\bar{\alpha}_i = \arccos \frac{(R_i + R_{i-1})^2 + (R_i + R_{i+1})^2 - (R_{i-1} + R_{i+1})^2}{2(R_i + R_{i-1})(R_i + R_{i+1})}, \tag{4.4}$$

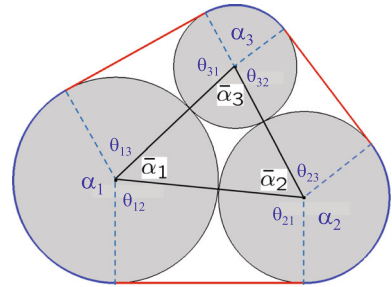
in detail

$$\begin{aligned} \bar{\alpha}_1 &= \arccos \frac{(R_1 + R_3)^2 + (R_1 + R_2)^2 - (R_3 + R_2)^2}{2(R_1 + R_3)(R_1 + R_2)}, \\ \bar{\alpha}_2 &= \arccos \frac{(R_2 + R_1)^2 + (R_2 + R_3)^2 - (R_1 + R_3)^2}{2(R_2 + R_1)(R_2 + R_3)}, \end{aligned}$$

and

$$\bar{\alpha}_3 = \arccos \frac{(R_3 + R_2)^2 + (R_3 + R_1)^2 - (R_2 + R_1)^2}{2(R_3 + R_2)(R_3 + R_1)}.$$

Fig. 5 Derivation of the sector angles for three arbitrary disks



Finally, the length ℓ_3 of $\partial\mathcal{H}$ of three touching disks is given by

$$\ell_3 = \ell_{L3} + \ell_{A3} = \ell_{L3} + R_1\alpha_1 + R_2\alpha_2 + R_3\alpha_3. \tag{4.5}$$

4.1.3 Four disks

For four disks with radii R_1, R_2, R_3 and R_4 , where $R_1 \geq R_2 \geq R_3 \geq R_4$, we only compute the length of the perimeter ℓ_4 for the case in which all four disks contribute an arc to the convex hull. All other cases can be reduced to two or three disks. As shown in KF19 we need to consider three possible counter-clockwise tours: 1–2–3–4–1, 1–2–4–3–1, and 1–3–2–4–1. The lengths of the lines segments had been given by KF19. To use the idea illustrated in Fig. 1 of KF19, we arrange the disks 1 and 2 horizontally, disk 3 on top touching disks 1 and 2, and disks 4 below disks 1 and 2. This can be understood as tour 1-4-2-3-1, which is the return tour corresponding to 1-3-2-4-1. For the upper part with disks 1, 2 and 3, we can use the formulae for three disks provided in Sect. 4.1.2 to compute $\bar{\alpha}_i$ and θ_{13}, θ_{23} as well as θ_{31} and θ_{32} . For the lower part with disks 1, 2, and 4, we denote the angles $\bar{\alpha}_i$ by $\bar{\gamma}_i$ and replace R_3 by R_4 leading to similar formulae as in Sect. 4.1.2:

$$\bar{\gamma}_1 = \arccos \frac{(R_1 + R_2)^2 + (R_1 + R_4)^2 - (R_1 + R_4)^2}{2(R_1 + R_2)(R_1 + R_4)},$$

$$\bar{\gamma}_2 = \arccos \frac{(R_2 + R_1)^2 + (R_2 + R_4)^2 - (R_1 + R_4)^2}{2(R_2 + R_1)(R_2 + R_4)},$$

and

$$\bar{\gamma}_4 = \arccos \frac{(R_4 + R_2)^2 + (R_4 + R_1)^2 - (R_2 + R_1)^2}{2(R_4 + R_2)(R_4 + R_1)}.$$

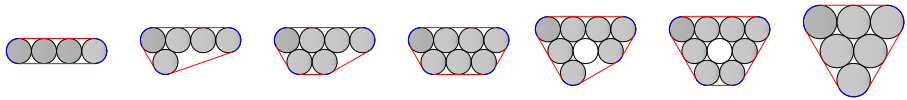
Now we get

$$\alpha_1 = 2\pi - \bar{\alpha}_1 - \bar{\gamma}_1 - \theta_{13} - \theta_{14},$$

$$\alpha_2 = 2\pi - \bar{\alpha}_2 - \bar{\gamma}_2 - \theta_{23} - \theta_{24},$$

$$\alpha_3 = 2\pi - \bar{\alpha}_3 - \theta_{32} - \theta_{31},$$

$$\alpha_4 = 2\pi - \bar{\gamma}_4 - \theta_{42} - \theta_{41},$$



(a) Block-1 (b) Block-2 (c) Block-3 (d) Block-4 (e) Block-5 (f) Block-6 (g) Block-7
Fig. 6 Block configurations 1 to 7 with 4, 5, 6, 7, 8, 9, and 6 congruent disks

with

$$\begin{aligned} \theta_{13} &= \arccos \frac{R_1 - R_3}{R_1 + R_3}, & \theta_{31} &= \pi - \theta_{13}, \\ \theta_{14} &= \arccos \frac{R_1 - R_4}{R_1 + R_4}, & \theta_{41} &= \pi - \theta_{14}, \\ \theta_{23} &= \arccos \frac{R_2 - R_3}{R_2 + R_3}, & \theta_{32} &= \pi - \theta_{23}, \\ \theta_{24} &= \arccos \frac{R_2 - R_4}{R_2 + R_4}, & \theta_{42} &= \pi - \theta_{24}. \end{aligned}$$

The minimal sum of the lengths of the line segments is given by

$$\ell_{L4} = 2 \left[\sqrt{R_1 R_4} + \sqrt{R_4 R_2} + \sqrt{R_2 R_3} + \sqrt{R_3 R_1} \right]. \tag{4.6}$$

Finally, the length ℓ_4 of $\partial\mathcal{H}$ of four extreme disks is given by

$$\ell_4 = \ell_{L4} + \ell_{A4} = \ell_{L4} + \sum_i R_i \alpha_i. \tag{4.7}$$

The analytic solutions have been compared to the numerical results of the test instances DC04, TC04, TC04a and TC04c defined in Tables 9 and 10. Note that TC04b cannot be used for this comparison as the rectangular target constraints become active; in this case, the optimal analytic solution is not feasible.

4.2 Minimal convex hulls for congruent disks in a strip packing problem

The task of this problem is to arrange a set of congruent disks of radius $R = 0.5$ in a rectangle with width $W = 4$ and arbitrary, or at least non-limiting length L in such a way that the length of the perimeter of the convex hull becomes minimal. As the disks have radius $R = 0.5$, we can have at most four disks in a layer. The solutions are established by a first block (bottom) which consists of layers with one, two, three, or four disks, and a final block (upper) embracing a main body of layers consisting of three or four disks. We have seven dominant configurations for the first block as demonstrated in Fig. 6. The upper block is just the up-side-down configuration of the first block.

Let us now build the model for the optimal configuration. Basically, this model is a partitioning model which covers n disks by the first block, the layers of the main body and the last block. For each block b we derive *a priori* its contribution L_b to the length of perimeter of \mathcal{H} , if it is selected as first or last block. The binary variables δ_b^{FB} and δ_b^{LB} indicate their selection. Note that different blocks can be selected as the first and the last block. The length contribution of blocks has been worked out in Appendix C.2.

The selection of layers inside the main body is governed by the binary variables δ_l^3 and δ_l^4 indicating whether layer l contains three or four disks. The objective function of the model, hereafter named minLSP, is to minimize the length of the perimeter of the convex hull.

Table 2 Lengths $L_b/R - \pi$ of the lower and upper blocks as worked out in Appendix C. The specification parameter S indicates the number of disks in the last layer (seen from bottom or top)

b	1	2	3	4	5	6	7
$L_b/R - \pi$	8	$4 + 2\sqrt{7}$	$6 + \sqrt{12}$	10	$6 + \sqrt{12}$	12	12
S	4	4	4	4	4	4	3

$$\ell = \sum_b L_b \delta_b^{FB} + \sum_l L_{3D} \delta_l^3 + \sum_l L_{4D} \delta_l^4 + \sum_b L_b \delta_b^{LB}, \tag{4.8}$$

where L_b follows from Table 2 and $L_{3D} = 2(\sqrt{3} - 1)R$, $L_{4D} = 2 \cdot 2R = 4R$.

L_{3D} and L_{4D} denote the lengths of a layer with three or four disks, respectively, contributed to the length ℓ . The number of disks in all layers equals the number n of disks to be placed, *i.e.*,

$$\sum_b L_b \delta_b^{FB} + \sum_l 3\delta_l^3 + \sum_l 4\delta_l^4 + \sum_b L_b \delta_b^{LB} = n. \tag{4.9}$$

Select one block for the first layer and the other for the last layer, *i.e.*, $\sum_b \delta_b^{FB} = 1$ and $\sum_b \delta_b^{LB} = 1$. A layer $l > 1$ can be a 3-layer, a 4-layer, or the last layer (last block): $\delta_l^3 + \delta_l^4 + \delta_l^L = \delta_l^A$, $\forall l$, where binary variable δ_l^A indicates whether layer l is active. Active layers are established by

$$\delta_l^{A+1} \leq \delta_l^A, \quad \forall l. \tag{4.10}$$

The last active layer is identified by $\delta_l^L = \delta_l^A - \delta_l^{A+1}$, $\forall l$. We also need to connect the number n_l of disks in layer l to the activity of that layer. As there are not more than 9 disks in an active layer (including blocks), we have

$$n_l \leq 9\delta_l^A, \quad \forall l. \tag{4.11}$$

Note that the first layer is always active, *i.e.*, $\delta_1^A = 1$ as it is associated with the first block. The last layer is associated with the last block.

It is never optimal if a 3-layer follows a 3-layer. Therefore we exclude two subsequent 3-layers by

$$\delta_i^3 + \delta_{i+1}^3 \leq 1, \quad \forall i. \tag{4.12}$$

Following the same argument it is never optimal if a 3-layer follows first block 7 (the highest layer of that block has three disks) and therefore we require

$$\delta_7^{FB} + \delta_2^3 \leq 1, \quad \forall i. \tag{4.13}$$

Similarly, it is never optimal if the second-last layer is a 3-layer and that one is followed by last block 7, *i.e.*,

$$\delta_{i-1}^3 + \delta_i^L + \delta_7^{LB} \leq 2, \quad \forall i. \tag{4.14}$$

Comments on the implementation Initially, for each block b we store x_b^0 , the x_1 -coordinate of the highest layer of disks (seen from bottom), and the number N_b^S of disks at this highest layer. To initialize the computation of the x_1 -coordinate of the main body we compute $x_1^L =$

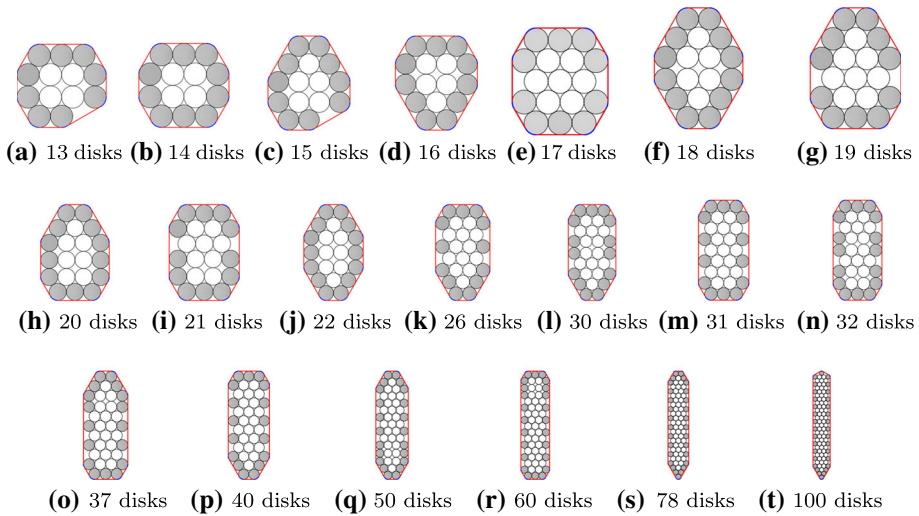


Fig. 7 Selected configurations for 13 to 100 disks

$\sum_b x_{1b}^0 \delta_b^{FB}$ and then iterate subsequently

$$x_1^L \rightarrow \begin{cases} x_1^L + \sqrt{3}R, & S = 3 \\ x_1^L + 2R, & S = 4 \end{cases}, \tag{4.15}$$

where S is a specification parameter indicating the number of disks in the previous layer. The x_1 -coordinates of the layer of disks in the upper block b are computed by a transformation which sets the x_1 -coordinates of the last layer to zero, the second-last layer to $x_{b, \text{LAST}}^0 - x_{b, \text{LAST}-1}^0$, and the first layer to $x_{b, \text{LAST}}^0 - x_{b, \text{LAST}-1}^0$. Then we use (4.15) to compute the x_1 -coordinates of all disks in the very layer. This model is solved easily even for large instances of several hundred disks within seconds. Figure 7 shows the selected configurations up to 100 disks.

5 Theoretical bounds and gaps

In this section, we provide some theoretical bounds and gaps for the perimeter length of minimal convex hull. We solve the disk packing problem which minimizes the radius of the circular container hosting all not necessarily congruent disks using `minRadiusCC`, `minSDC`, or `VOROPACK-D (CC)`. These initial arrangements of disks are input to `minDPCH` producing a minimal discretized perimeter of the convex hull \mathcal{H} , or directly input to `QuickhullDisk` (Refer to PL4 in Sect. 6). `QuickhullDisk` computes the perimeter length ℓ^{QH} of \mathcal{H} and generates the extreme disks and vertices which are required to initialize the binary variables δ_j^A , δ_{ij}^S , δ_{ij}^D , and δ_{ij} and to establish $\partial\mathcal{H}$ for `MinPerim` as described in Appendix C.1. With these input we follow up with `MinPerim` to compute ℓ^{MP} .

As expected, we observe $\mathcal{L}^{\text{II}} \leq \ell^{\text{MP}} \leq \ell^{\text{QH}} \leq l^{\text{CC}}$, where l^{CC} is the length of circumference for smallest circular container from `minRadiusCC`, `VOROPACK-D (CC)`, or benchmark data from `Packmomania` (Refer to Sect. 7). ℓ^{QH} is the length of the perimeter of the convex hull constructed by `QuickhullDisk` and ℓ^{MP} is the length of the perimeter of the convex hull obtained by `MinPerim`. \mathcal{L}^{II} is the lower bound derived from the isoperimetric inequality

$4\pi a \leq \ell^2$ [30] relating the square of the circumference ℓ of a closed curve and the area a of the region it encloses on the plane. Let $a = Area(\mathcal{H})$, i.e., the area of a convex hull \mathcal{H} , and $A = \pi \sum_i R_i^2 < a$, where R_i 's are the radii of input disks. Hence, the following weak lower bound \mathcal{L}_{lb}^{ii} can be easily established:

$$\mathcal{L}_{lb}^{ii} = \sqrt{4\pi A} = 2\pi \sqrt{\sum_i R_i^2}. \tag{5.1}$$

Especially, for n disks with $R_i = n - i + 1, i = 1, 2, \dots$, the lower bound \mathcal{L}_{lb}^{ii} on the length of $\partial\mathcal{H}$ is reduced to

$$\mathcal{L}_{lb}^{ii} = 2\pi \sqrt{\frac{n(n+1)(2n+1)}{6}}. \tag{5.2}$$

For congruent disks, on the other hand, a tighter lower bound can be derived from Wegner’s inequality which establishes a lower bound of the area A of the convex hull \mathcal{H} of n unit disks (i.e., every disk has a unit radius) as follows: $A \geq \sqrt{12}(n-1) + (2 - \sqrt{3}) [\sqrt{12n-3} - 3] + \pi$ [3,18]. Let ℓ be the length of the boundary $\partial\mathcal{H}$ of the convex hull \mathcal{H} . Given the isoperimetric inequality $4\pi A \leq \ell^2$, we get the **Wegner lower bound** \mathcal{L}_{lb}^{WV} on ℓ as follows.

$$\mathcal{L}_{lb}^{WV} = \sqrt{4\pi \left[\sqrt{12}(n-1) + (2 - \sqrt{3}) [\sqrt{12n-3} - 3] + \pi \right]} \tag{5.3}$$

which is much stronger than the lower bound \mathcal{L}_{lb}^{ii} derived from isoperimetric inequality. For the congruent disks with radius 0.5, divide both sides of Eq. (5.3) by $\sqrt{4.0}$. For instance, if we have $n = 13$ such disks, we get $\mathcal{L}_{lb}^{WV} \approx 12.2017$. Table 5 shows two lower bounds \mathcal{L}_{lb}^{ii} and \mathcal{L}_{lb}^{WV} for some congruent disks with radius 0.5 using Eqs. (5.1) and (5.3).

Let ℓ_{tot} be the total length of the convex hull perimeter. Let ℓ_L and ℓ_A be the subtotal lengths of the linear and circular segments on the convex hull boundary, respectively. Hence, $\ell_{tot} = \ell_L + \ell_A$. Let \mathcal{L}_{lb}^{best} be the best lower bound of the length of the convex hull boundary. Let

$$\Delta = \frac{\ell_{tot} - \mathcal{L}_{lb}^{best}}{\mathcal{L}_{lb}^{best}}. \tag{5.4}$$

Then, Δ defines the gap between the best solution obtained for the perimeter and the best lower bound. The column $D(\Delta^{ii}(\%))$ of Table 5 shows the gap between the best solution $G(\ell_*)$ in analytic form and $B(\mathcal{L}_{lb}^{ii})$. Note that the optimal solution in column $G(\ell_*)$ of Table 5 for $n = 13$ is $8 + \sqrt{3} + \pi = 12.8736$. Hence, the corresponding gap is $\Delta = \frac{12.8736 - 11.3272}{11.3272} \times 100 = 13.6521\%$. Similarly, the column $E(\Delta^{WV}(\%))$ shows the gap between the best solution $G(\ell_*)$ obtained and $C(\mathcal{L}_{lb}^{WV})$. See columns $\mathcal{L}_{lb}^{ii}, \mathcal{L}_{lb}^{WV}, \Delta^{ii}(\%),$ and $\Delta^{WV}(\%)$ in Table 5.

6 Polyolithic framework of the minimal convex hull problems

In this section, we propose a polyolithic approach to modeling and solving the minimal convex hull problems using the MINLP and NLP models in Table 1, VOROPACK-D, and QuickhullDisk. We recommend readers to refer to Appendix B for VOROPACK-D, QuickhullDisk, and Voronoi diagrams. Figure 1 is redrawn in Fig. 8 with algorithmic details. Step I constructs an initial disk arrangement and Step II minimizes the discretized perimeter of the approximated convex hull using minDPCH. This minimization can be from

scratch, using minDPCH on its own, or by initializing minDPCH with the initial arrangement of the disks obtained in Step I. Step III constructs the convex hull of the disk arrangement and computes the perimeter length of the convex hull, and Step IV constructs the minimal convex hulls of the disks via an MINLP model MinPerim. Step I finds a non-overlapping disk arrangement. Depending on the existence of a target domain, Step I may use NLP model(s) or either VOROPACK-D(CC) or VOROPACK-D(RC). Step II minimizes the discretized perimeter of the approximated convex hull using an NLP model minDPCH. There are two alternatives in Step II: minDPCH may take the output of Step I as its input or minDPCH may be used as a monolith for Step III. In Step III, QuickhullDisk is used to construct the convex hull of the disk arrangement resulting from Step I or II. QuickhullDisk provides the initial values for MinPerim in Step IV. Note that two different versions of minSDC appear in Step I: One model with Eqs. (3.5), (3.6), and (3.10) used for a circular container (with no target domain), and the other with Eqs.(3.5), (3.6), (3.7), (3.8), and (3.10) for a rectangular target domain. We have validated the proposed PolyLithic approach (PL) using the following algorithmic settings:

- PL1. Verification of VOROPACK-D algorithm: We have done this by comparing its solutions to those of the NLP models.
- (a) VOROPACK-D(CC) (circular container, i.e. no target domain): Comparing the solutions to the global minima of minRadiusCC; Table 3.
 - (b) VOROPACK-D(RC) (rectangular target domains): Comparing the solutions to the global minima of CutDisks; Table 5.
- PL2. Initial disk arrangements: We have obtained from the following NLP models and algorithm.
- (a) NLP models with various objective functions minimizing
 - i. the radius of a circular container using minRadiusCC (good for larger numbers of disks),
 - ii. the sum of radius weighted distances from all disks to the averaged center using minSDC,
 - iii. the area of a rectangle using CutDisks, and
 - iv. the perimeter length of a rectangle using CutDisks.
 - (b) minDPCH for the minimal discretized perimeter of the convex hull (in monolithic or polyolithic version).
 - (c) VOROPACK-D algorithm.
- PL3. Monolithic version of minDPCH: Output of minDPCH is provided for input of QuickhullDisk; Tables 4, 5, 6, 7, 8 (Flow C2–C3 in Fig. 8).
- PL4. Polyolithic versions of minDPCH: NLP model(s) or VOROPACK-D precedes minDPCH as follows.
- (a) minRadiusCC: Table 4 (Flow A11-A21-A3).
 - (b) minSDC: Table 4 (Flow A12-A21-A3).
 - (c) CutDisks: Tables 6, 7, and 8 (Flow B11-B12-B21-B3).
 - (d) VOROPACK-D(CC): Tables 3 (Flow A13-A22-A3) and 4 (Flow A13-A21-A3).
 - (e) VOROPACK-D(RC): Tables 6, 7, and 8 (Flow B13-B22-B3).
- PL5. Polyolithic for solving MinPerim (The first polyolithic approach P1 proposed in KF19): We solve the disk packing problem minimizing the area or perimeter length of the design rectangle hosting all disks. Then this initial disk arrangement is used for

initializing MinPerim. We provide this to compare the results of P1 with those of this paper (Tables 5, 6, 7, 8).

7 Validation of the polyolithic framework via numerical experiments

We have verified and validated the solution quality and performance of the polyolithic approach comparing its experimental results to analytic solutions, theoretical bounds, and some benchmark data including the best known results from both KF19 and the Packomania website (www.packomania.com, visited on Jun 29, 2018, maintained by E. Specht). Good initial disk arrangements for the polyolithic approach were obtained using both the NLP models and VOROPACK-D. minDPCH is the strongest NLP model and thus almost all experiments contain the results obtained by minDPCH. The congruent disk experiments summarized in Table 5 enable us to evaluate and compare the quality of minDPCH to the analytic solutions.

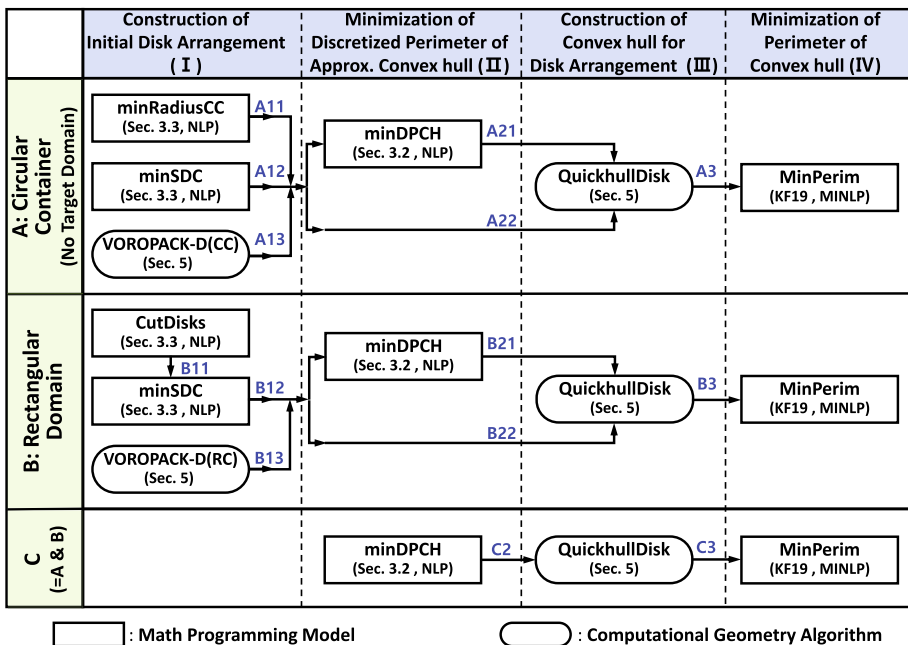


Fig. 8 The proposed polyolithic framework consisting of up to four computational steps: (I) Initial disk arrangement, (II) discretized perimeter of the approximated convex hull, (III) convex hull of the disk arrangement computed by QuickhullDisk, and (IV) perimeter minimization of convex hulls using MinPerim. Depending on the existence of a target domain, Step I uses either VOROPACK-D (CC) or VOROPACK-D (RC) as well as NLP models. They may be used to initialize minDPCH to compute the discretized perimeter, or minDPCH is used as a monolith in Step II. In Step III, QuickhullDisk is used to compute the convex hull of the initial disk arrangement available after Step I or II. Step III finishes with providing intermediate data allowing to compute initial values for MinPerim in Step IV. Note that minSDC comes in two slightly different versions: One model with (3.5), (3.6), and (3.10) used in the circular container context, and the other with (3.5), (3.6), (3.7), (3.8), and (3.10) subject to a rectangular target domain

7.1 Experimental environment: data, software, and computing platforms

Data sets

We have performed the computational tests for both circular containers (no target domain) and rectangular domains. For circular containers, we used one of the Packomania data sets where the disk radii are defined as $R_i = n - i + 1$, $n = 1, 2, \dots, 1,000$. This data enables us to compare our results with those of Packomania for $n \leq 200$. Note that the original Packomania set has $R_i = i$, but some of our models and algorithms require that the radii of the disks are sorted in descending order. The computational resources used for getting the best-known solutions of Packomania are not known. For rectangular domains, two different instance types Cx or Dx were used in KF19. Instances with congruent disks start with the prefix “C” while instances with non-congruent disks start with “D”. The parameter x stands for the number of disks in each instance, e.g., D03 represents an instance with three non-congruent disks. If the final gap for the minimization is smaller than 10^{-4} , the instance is labeled with an *. We used the following data sets for experiments.

SET-P: Packomania data.

SET-A: C05-C90 (in Table 5).

SET-B: DC03-DC10 (in Table 9).

SET-C: TC03-TC28 (in Table 10).

SET-D: D series (in Table 11).

The instances in SET-D of Table 11 are established by combining some disk sets of B and C, e.g., D21 = $3 \times$ TC07 (Refer to Appendix D for Tables 9, 10, and 11). Note that only SET-B and SET-C have been used by KF19.

Software and hardware

The mathematical optimization models as well as the polyolithic approaches were implemented in GAMS 28.2 using two global solvers, BARON (with CPLEX for the LP relaxation and MINOS for the NLP problem) and LINDO. VOROPACK-D was used to pack circular disks in a container of circular or rectangular shape [35] and QuickhullDisk was used to construct the convex hulls of disk arrangements [26]. Computations were executed on three similar machines: i) PC1: a 64 bit machine with an Intel(R) Core(TM) i7 CPU 2.8 GHz, 16 GB, RAM running Windows 7, ii) PC2: a 64 bit machine with an Intel(R) Core(TM) i7-7700 (3.6 GHz), 16GB RAM running Windows 10 Pro, iii) PC3: a 64 bit machine with an Intel(R) Core(TM) i7-7700 (3.6 GHz), 16GB RAM running Windows 10 Pro. All computations were done using a single core processor. However, the proposed polyolithic approaches could exploit parallel computer hardware (multi-core system, clusters of computers, etc.) by applying the technique introduced by Kallrath *et al.* [15]. This allows us to select a polyolithic method and its parameters and to solve it on a selected core or computer. By doing this in parallel, we may pick the best solution obtained within a given time limit.

Computation time limits

Computation time limits for minRadius, CutDisks, and minSDC were set to 3,600 sec; Those of MinPerim and minDPCH were set to 36,000 sec. An algorithm terminated when the gap was reached. The computation times for VOROPACK-D and QuickhullDisk were negligible (*i.e.* a few seconds, at most). For details on their computation time, see [35] and [26]. The computation times for the best-known solutions of the Packomania data SET-P are not publicly available.

Assumption

Let N_i^{ls} be the maximal number of line segments of the convex hull boundary which is incoming (ending) to disk i or outgoing (starting) from disk i . For example, a disk d_1 of Fig. 2 has an incoming line segment l_3 and outgoing line segment l_1 . All numerical experiments in this section have been performed with $N_i^{ls} = 1$, *i.e.*, each disk has at most one incoming and one outgoing line segment on the convex hull boundary unless we explicitly express some different setting.

7.2 Experiments and discussions

7.2.1 Circular container (No target domain)

We performed experiments for the circular container problem using the Packomania data SET-P. Tables 3 and 4 summarize the results. For smaller number of disks, as is expected, the minimal circular container solutions are not strong initial solutions for minimal convex hulls. However, for 20 or more disks they are reasonable initial solutions. From about $n = 50$, VOROPACK-D(CC) provides better initial solutions than the NLP models based on minRadiusCC and minSDC. The monolithic approach works only reasonably well up to a certain problem size of 170 disks in Table 4. Within the polyolithic approach (PL4 in Sect. 6), minRadiusCC and minSDC - in terms of the quality of the configuration—are competitive up to 40 disks; for larger problem instances VOROPACK-D(CC) outperforms them.

Figure 9 shows the arrangements of non-congruent disks in circular container which are produced by applying different methods to Packomania data SET-P ($n=5, 10, 15, 20, 30, 50$, and 100). The columns of the table are as follows. Column (A): n : The number of disks. Columns (B - E): The disk arrangements by four different methods and their convex hull boundaries as follows. Column (B): VOROPACK-D(CC) (This column corresponds to Column B of Table 3). Column (C): minRadiusCC (Column D of Table 3). Column (D): VOROPACK-D(CC) \rightarrow QuickhullDisk \rightarrow MinPerim (Column G of Table 3). Column (E): minRadiusCC \rightarrow QuickhullDisk \rightarrow MinPerim (Column H of Table 3). We have the following observations.

- In all cases except two (Column (D) of $n = 50, 100$), MinPerim improves initial solutions from both VOROPACK-D(CC) and minRadiusCC.
- Improvements become smaller as the problem size increases.
- The methods using only NLP and MINLP models do not work for the large problem instances (Refer to Tables 3 and 4). In this case, the computational geometry algorithm such as VOROPACK-D(CC) could be a good alternative.

Monolithic experiments

Table 3 reports the radii of the containers obtained from VOROPACK-D(CC) (r_V^{cc}), the best-known radii reported in Packomania (r_P^{cc}), and those of minRadiusCC (r_{minR}^{cc}) followed by the relative gap (Δ^{ii}) in percent between column D(r_{minR}^{cc}) and the best lower bound, ℓ_{lb} , provided by BARON (not in the table), *i.e.*, $(D(r_{minR}^{cc}) - \ell_{lb}) / \ell_{lb} * 100$. The next column shows the length $l_{peri}^{cc} = 2\pi \min\{r_V^{cc}, r_P^{cc}, r_{minR}^{cc}\}$ of the perimeter of the smallest circular container. Columns $\ell(r_V^{cc})$ and $\ell(r_{minR}^{cc})$ show the lengths of perimeters of the convex hull when feeding the initial solutions obtained by VOROPACK-D(CC) or minRadiusCC through QuickhullDisk into MinPerim, respectively (*i.e.*, VOROPACK-D(CC) \rightarrow QuickhullDisk \rightarrow MinPerim or minRadiusCC \rightarrow QuickhullDisk \rightarrow MinPerim). The last two columns display $\ell(KF19)$, the value obtained by MinPerim using the polyolithic mode P1 described in KF19 and the lower bound \mathcal{L}_{lb}^{ii} derived from the isoperimetric inequal-

ity in Eq. (5.2). The entry marked *green* in each row indicates the smallest perimeter length found for that problem instance.

Note that for smaller problem instances $n \leq 8$, the followings were observed: (i) we are able to close the gap and prove the global optimality of $r_{\min R}^{cc}$; (ii) $r_{\min R}^{cc} = r_{\mathbf{P}}^{cc}$; (iii) the monolithic computations of the minimal length of the perimeter using `minRadiusCC` produce disk arrangements which fit into a circular container of the radius $r_{\min R}^{cc} = r_{\mathbf{P}}^{cc}$. The computed disk is proven to be optimal. This leads us to formulating a conjecture in Appendix E which relates the optimal solution of minimal circular container to that of the minimal convex hull.

Polyolithic experiments The first column ℓ_b of Table 4 is the best perimeter length of Table 3, i.e., $\ell_b = \min\{\ell(r_{\mathbf{V}}^{cc}), \ell(r_{\min R}^{cc}), \ell(\text{KF19})\}$. The column ℓ_m is obtained by initializing `MinPerim` (through `QuickhullDisk`) by the monolithic solution found by `minDPCH` (i.e., `minDPCH` \rightarrow `QuickhullDisk` \rightarrow `MinPerim`). The next column ℓ_p^1 is the polyolithic value which is obtained by initializing `minDPCH` with `minSDC` (i.e., `minSCD` \rightarrow `minDPCH` \rightarrow `QuickhullDisk` \rightarrow `MinPerim`). The following two columns ℓ_p^2 and ℓ_p^3 are identical to ℓ_p^1 except that `minSDC` is replaced by `VOROPACK-D (CC)` and `minRadiusCC`, respectively. The last column is the lower bound $\mathcal{L}_{\text{lb}}^{\text{ii}}$ of the length of the convex hull perimeter derived from the isoperimetric inequality and is a duplication of the last column of Table 3. For most cases, ℓ_p^2 using `VOROPACK-D (CC)` provides the best initialization of `minDPCH`. For many instances in the circular container experiments we found that the arrangements of disks leading to the minimal circular container radius also had the minimal perimeter convex hulls and vice versa. This observation inspired us to attempt to formulate the conjecture in Appendix E.

7.2.2 Rectangular domain

We performed experiments for rectangular domain against SET-A, SET-B, SET-C, and SET-D of Sect. 7.1.

The experimental results are compiled as follows: Table 5 for congruent disks, Tables 6, 7, and 8 for non-congruent disks. For congruent disks, in most cases, the monolithic solutions using `minDPCH` and polyolithic solutions using `VOROPACK-D` outperform the previous results reported in KF19. For non-congruent disks except SET-C (Table 7), the monolithic solutions using `minDPCH` (PL3 of Sect. 6) outperform others in most cases. For SET-C, the polyolithic solutions using `minDPCH` based on `CutDisks` and `minSDC` (PL4 of Sect. 6) outperform others.

Congruent disks Table 5 shows the results for the congruent disks of radius $R = 0.5$. The width W of a rectangular domain is fixed by $W = 4$ and its length $L = 8$ for $n \leq 30$, $L = 14$ for $30 < n \leq 60$, and $L = 25$ for $n > 60$. The columns of the table are as follows. Column Instance: The instance of congruent disk models as named in KF19 (e.g., C20 involves 20 congruent disks). $\mathcal{L}_{\text{lb}}^{\text{ii}}$: The lower bound of the perimeter length derived from the isoperimetric inequality as $\mathcal{L}_{\text{lb}}^{\text{ii}} = \sqrt{n\pi}$. $\mathcal{L}_{\text{lb}}^{\text{W}}$: The Wegner lower bound in Eq. (5.3). $\Delta^{\text{ii}}(\%)$: The gap between the analytic solution ℓ_* in Column G and $\mathcal{L}_{\text{lb}}^{\text{ii}}$ using Eq. (5.4). $\Delta^{\text{W}}(\%)$: The gap between the analytic solution ℓ_* and $\mathcal{L}_{\text{lb}}^{\text{W}}$. $\Delta(\%)$: The gap between the analytic solution ℓ_* and $\min\{\ell_m, \ell_p^{\text{V}}, \ell_p^{\text{P1}}\}$. ℓ_* : The best solution (analytic and MILP solution of `minLSP` are identical up to 9 decimal places). CPU: The CPU time in seconds to compute the monolith solution ℓ_m using `minDPCH`. ℓ_m : The monolithic length of the convex hull perimeter obtained by solving the grid model using `minDPCH`. ℓ_p^{V} : The polyolithic length of the convex hull perimeter using `VOROPACK-D (RC)`. ℓ_p^{P1} : The polyolithic length of the convex hull perimeter using homotopy P1 in `MinPerim`. $\ell(\text{KF19})$: The perimeter reported in KF19. In most cases, $\min\{\ell_m, \ell_p^{\text{V}}\} \leq \ell(\text{KF19})$. Thus the current approach improves the

Table 3 Circular container

A	B	C	D	E	F	G	H	I	J
n	r_V^{cc}	r_C^{cc}	r_{minR}^{cc}	$\Delta^{ii}(\%)$	f_{cc}^{cc} f_{peri}	$\ell(r_V^{cc})$	$\ell(r_{minR}^{cc})$	ρ_P^{PI}	J L_{lb}^{ii}
1	5	9.0097	9.0014	0.00	56.5575	50.9570	50.9612	52.4476	46.5973
2	6	11.2784	11.0570	0.00	69.4734	65.9766	67.1218	68.1823	59.9378
3	7	13.6996	13.4621	0.00	84.5849	82.5075	82.0283	87.3210	74.3437
4	8	16.4066	16.2217	0.00	101.9242	99.6866	98.7975	100.9288	89.7418
5	9	19.5698	19.2332	11.61	120.8457	118.8822	117.1919	119.1669	106.0724
6	10	22.5468	22.0202	13.74	138.2313	135.1169	134.4312	144.8100	123.2850
7	11	25.7881	24.9606	16.50	156.8323	157.3633	154.9329	161.3144	141.3368
8	12	29.0594	28.3714	19.79	178.2627	177.1516	175.7230	180.3760	160.1904
9	13	32.5989	31.5459	21.50	198.2085	198.9966	197.0374	202.6618	179.8133
10	14	36.5604	35.0956	24.15	220.5125	224.7975	219.5618	227.0455	200.1764
11	15	39.6046	38.8380	26.18	244.0263	243.4188	242.2126	247.8414	221.2538
12	16	43.7347	42.4581	28.23	266.7722	270.2806	268.0676	275.7099	243.0220
13	17	47.9683	46.2913	30.24	290.8571	293.5291	292.0681	266.0064	265.4599
14	18	52.2834	50.1198	31.77	314.9118	322.3334	319.7210	328.3126	288.5481
15	19	56.4028	54.2403	33.45	340.8018	349.1889	347.1149	356.7371	312.2686
16	20	61.7502	58.4006	34.62	366.9416	373.0600	372.0307	376.7279	336.6052
17	30	109.4269	104.5404	45.68	656.8465	673.7421	672.7333	682.7785	610.9570
18	40	166.0467	159.0216	52.60	999.1620	1032.7911	1032.1717	1074.7765	934.9076
19	50	229.6400	220.5654	57.02	1385.8533	1433.4945	1427.2552	1520.8474	1301.7723
20	60	300.6587	289.3423	60.68	1817.9911	1879.0802	1882.3866	1922.6160	1707.0155
21	70	373.5568	363.5378	63.43	2284.1756	2342.6714	2368.3415	2515.3915	2147.2964
22	80	460.2343	442.7192	65.70	2781.6865	2883.4737	2884.9102	2992.1280	2620.0205
23	90	548.4759	526.9030	67.43	3310.6293	3440.1942	3440.9015	3576.3842	3123.0971
24	100	639.3226	615.8676	69.14	3869.6105	4004.5129	4026.2467	4199.1646	3654.7945
25	110	730.4575	710.1000	70.57	4461.6898	4584.9307	4649.4883	4872.6842	4213.6476

Table 3 continued

A <i>n</i>	B r_{V}^{cc}	C r_{P}^{cc}	D $r_{\text{minR}}^{\text{cc}}$	E Δ^{ih} (%)	F $r_{\text{per}}^{\text{cc}}$	G $\ell(r_{\text{V}}^{\text{cc}})$	H $\ell(r_{\text{minR}}^{\text{cc}})$	I $\ell_{\text{P}}^{\text{PI}}$	J $\mathcal{L}_{\text{lb}}^{\text{ii}}$
26	120	837.2709	808.3093	841.5741	71.90	5078.7573	5255.1838	n.s.f.	4798.3958
27	130	943.9314	910.0150	951.8927	73.00	5717.7928	5924.4589	n.s.f.	5407.9395
28	140	1055.5848	1016.2440	1061.3296	73.71	6385.2491	6624.9328	n.s.f.	6041.3080
29	150	1166.2653	1125.9622	1195.9126	75.00	7074.6294	7319.9560	n.s.f.	6697.6368
30	160	1286.1900	1238.9335	n.s.f.	7784.4487	8073.4930	n.s.f.	7376.1491	
31	170	1400.3923	1356.1791	n.s.f.	8521.1243	8790.6899	n.s.f.	8076.1420	
32	180	1525.4812	1476.5658	n.s.f.	9277.5363	9577.5359	n.s.f.	8796.9755	
33	190	1657.2579	1600.6699	n.s.f.	10057.3057	10400.1043	n.s.f.	9538.0636	
34	199	1773.7489	1714.6960	n.s.f.	10773.7530	11133.6737	n.s.f.	10221.9141	
35	200	1803.8520	1726.2219	n.s.f.	10846.1722	11321.7599	n.s.f.	10298.8672	
36	300	3271.9177	n.a.	n.s.f.	20558.0652	20548.1102	n.s.f.	18896.6733	
37	400	5038.1656	n.a.	n.s.f.	31655.7281	31634.2137	n.s.f.	29075.1982	
38	500	7023.5847	n.a.	n.s.f.	44130.4843	44116.8861	n.s.f.	40618.6184	
39	600	9216.3811	n.a.	n.s.f.	57908.2302	57892.3023	n.s.f.	53381.2339	
40	700	11584.3174	n.a.	n.s.f.	72786.4132	72771.7496	n.s.f.	67256.0473	
41	800	14158.7510	n.a.	n.s.f.	88962.0563	88943.0673	n.s.f.	82160.1381	
42	900	16922.2138	n.a.	n.s.f.	106325.4050	106306.1568	n.s.f.	98026.7829	
43	1000	19768.3650	n.a.	n.s.f.	124208.3005	124184.8200	n.s.f.	114800.7767	

Packomania data SET-P (Radii: $R_i = n - i + 1$). Column A(*n*): The number of disks. B(r_{V}^{cc}): The radii of the circular container obtained by VOROPACK-D(CC). C(r_{P}^{cc}): The radii of the circular container reported in Packomania. D($r_{\text{minR}}^{\text{cc}}$): The radii of the circular container obtained by minRadiusCC. Monolithic solutions using minRadiusCC. E(Δ^{ih} (%)): The relative gap in percent between column D($r_{\text{minR}}^{\text{cc}}$) and the best lower bound, ℓ_{lb} , provided by BARON (not in the table), (D($r_{\text{minR}}^{\text{cc}}$) - ℓ_{lb}) / ℓ_{lb} * 100. F($r_{\text{per}}^{\text{cc}}$): The perimeter length of the smallest circular container ($\approx 2\pi \min(r_{\text{V}}^{\text{cc}}, r_{\text{P}}^{\text{cc}}, r_{\text{minR}}^{\text{cc}})$). G($\ell(r_{\text{V}}^{\text{cc}})$): The perimeter length of the convex hull obtained by MinPerim with the initial solution from VOROPACK-D(CC) (VOROPACK-D(CC) → QuickhullDisk → MinPerim). H($\ell(r_{\text{minR}}^{\text{cc}})$): The perimeter length of the convex hull obtained by MinPerim with the initial solution from minRadiusCC (minRadiusCC → QuickhullDisk → MinPerim). I($\ell_{\text{P}}^{\text{PI}}$): Polyhedral length of the perimeter using homotopy P1 in MinPerim. J($\mathcal{L}_{\text{lb}}^{\text{ii}}$): The lower bound derived from the isoperimetric inequality resulting in Eq. (5.2). n.a.: not available. n.s.f.: no solution has been found during the allowed CPU time. The entries marked *bold* indicate the smallest values found for the problem instance. Computation time limits for each run for each problem instance: 10h for minRadiusCC in order to find the monolithic solutions (Note, in other tables, minRadiusCC is set to 1h in order to find only initial solutions); 1h for CutDisks and minSDC; 10h for MinPerim

Table 4 Circular container

	A	B	C	D	E	F	G
	n	ℓ_b	ℓ_m	ℓ_p^1	ℓ_p^2	ℓ_p^3	\mathcal{L}_{ib}^{ii}
1	5	50.9570	50.9239	50.9239	50.9239	51.6695	46.5973
2	6	65.9766	65.2320	65.2320	65.2320	65.3374	59.9378
3	7	82.0283	84.6088	81.5129	80.8543	81.0607	74.3437
4	8	98.7975	98.5126	98.3966	97.7753	98.7391	89.7418
5	9	117.1919	117.3312	116.0138	115.9662	115.9211	106.0724
6	10	134.4312	135.6620	134.9659	134.2246	134.4611	123.2850
7	11	154.9329	154.2528	154.2777	154.3223	154.6954	141.3368
8	12	175.7230	175.8432	174.5132	175.0881	175.1553	160.1904
9	13	197.0374	197.4531	196.9732	195.8129	197.4724	179.8133
10	14	219.5618	218.6733	220.0829	218.9954	218.1065	200.1764
11	15	242.2126	243.3812	242.2063	242.1223	242.6845	221.2538
12	16	268.0676	266.8615	266.8742	266.4584	265.6643	243.0220
13	17	266.0064	290.1897	291.2531	290.7442	290.7412	265.4599
14	18	319.7210	316.9774	316.4747	316.2817	315.4277	288.5481
15	19	347.1149	337.2536	341.4287	341.5054	342.5435	312.2686
16	20	372.0307	368.0723	368.2716	368.4323	367.3466	336.6052
17	30	672.7333	670.9710	692.5670	679.8542	673.6457	610.9570
18	40	1032.1717	1027.0800	1063.5384	1032.8375	1025.5774	934.9076
19	50	1427.2552	1437.4821	1464.9400	1433.6786	1437.5462	1301.7723
20	60	1879.0802	1892.7856	1922.2069	1865.7066	1869.5897	1707.0155
21	70	2342.6714	2380.0346	2451.6021	2319.4709	2362.1225	2147.2964
22	80	2883.4737	2908.6887	2967.4288	2861.4114	2865.5523	2620.0205
23	90	3440.1942	3463.1676	3525.0382	3401.3496	3444.2215	3123.0971
24	100	4004.5129	4254.8663	4151.8010	3969.4806	4022.6399	3654.7945
25	110	4584.9307	4608.5208	4686.0451	4551.9120	4631.2520	4213.6476
26	120	5255.1838	5268.0139	5284.9654	5206.3910	5261.7400	4798.3958
27	130	5924.4589	6019.1557	5928.4277	5872.4614	n.s.f.	5407.9395
28	140	6624.9328	6709.8229	6632.5330	6554.0464	n.s.f.	6041.3080
29	150	7319.9560	7505.8314	7403.2186	7265.3988	n.s.f.	6697.6368
30	160	8073.4930	8118.3524	8161.2211	8118.3524	n.s.f.	7376.1491
31	170	8790.6899	8867.1624	8896.4340	8723.9592	n.s.f.	8076.1420
32	180	9577.5359	n.s.f.	9688.8061	9577.5359	n.s.f.	8796.9755
33	190	10400.1043	n.s.f.	10561.5863	10325.0666	n.s.f.	9538.0636
34	199	11133.6737	n.s.f.	n.s.f.	11079.6713	n.s.f.	10221.9141
35	200	11321.7599	n.s.f.	n.s.f.	11205.9881	n.s.f.	10298.8672
36	300	20548.1102	n.s.f.	n.s.f.	20374.3794	n.s.f.	18896.6733
37	400	31634.2137	n.s.f.	n.s.f.	31373.8078	n.s.f.	29075.1982
38	500	44116.8861	n.s.f.	n.s.f.	43866.1295	n.s.f.	40618.6184
39	600	57892.3023	n.s.f.	n.s.f.	57884.6491	n.s.f.	53381.2339
40	700	72771.7496	n.s.f.	n.s.f.	i.f	n.s.f.	67256.0473

Table 4 continued

	A	B	C	D	E	F	G
	n	ℓ_b	ℓ_m	ℓ_p^1	ℓ_p^2	ℓ_p^3	\mathcal{L}_{lb}^{ii}
41	800	88943.0673	n.s.f.	n.s.f.	i.f.	n.s.f.	82160.1381
42	900	106306.1568	n.s.f.	n.s.f.	106070.8167	n.s.f.	98026.7829
43	1000	124184.8200	n.s.f.	n.s.f.	i.f.	n.s.f.	114800.7767

Packomania data SET-P (Radii: $R_i = n - i + 1$). Column A(n): The number of disks. B(ℓ_b): The best perimeter length in Table 3, *i.e.*, $\ell_b = \min\{\ell(r_{\sqrt{V}}^{cc}), \ell(r_{\min R}^{cc}), \ell(\text{KF19})\}$. C(ℓ_m): The solution of (minDPCH \rightarrow QuickhullDisk \rightarrow MinPerim). D(ℓ_p^1): The polyolithic solution of (minSDC \rightarrow minDPCH \rightarrow QuickhullDisk \rightarrow MinPerim). E(ℓ_p^2): The polyolithic solution of (VOROPACK-D (CC) \rightarrow minDPCH \rightarrow QuickhullDisk \rightarrow MinPerim). F(ℓ_p^3): The polyolithic solution of (minRadiusCC \rightarrow minDPCH \rightarrow QuickhullDisk \rightarrow MinPerim). Refer to PL3 in Sect. 6 for ℓ_m . Refer to PL4 in Sect. 6 for ℓ_p^1 , ℓ_p^2 , and ℓ_p^3 . G(\mathcal{L}_{lb}^{ii}): The lower bound derived from the isoperimetric inequality. *i.f.*: The initialization of minDPCH based on the output of VOROPACK-D (CC) failed, *i.e.*, turned out to be infeasible. *n.s.f.*: No feasible solution found. The bold entries indicate the smallest perimeter length found for that problem instance. Computation time limits for each run for each problem instance: 1h for minRadius, CutDisks, and minSDC; 10h for MinPerim and minDPCH

results by KF19. The numerical solutions of the instances marked with an *, *e.g.*, C13*, are identical to the analytic results or semi-analytic ones based on the partitioning model. Zeros of column F($\Delta(\%)$) also shows the same information.

The last column (Best Arrangement): Layer arrangements for congruent disks (C05 through C20). As the target rectangle has width $W = 4$, at most 4 disks can find place in the width direction. Therefore, we are facing a sort of a strip packing problem for more than five disks in which disks are arranged in layers. The last column of Table 5 and Fig. 10 symbolically and graphically reveal the best arrangement patterns for up to 20 disks, respectively; see also other columns of Table 5 for the analytic and numeric results. For instance, to understand Table 5 better, read configuration C20 in Fig. 10 from left to right vertically. The first column of disks at the very left contains two disks, followed by a column of three disks, then three times four disks, and finally three disks at the very right.

Non-congruent disks

Tables 6, 7, and 8 show the results for non-congruent disks in a rectangular domain for SET-B, SET-C, and SET-D, respectively. The tables report the lower bound \mathcal{L}_{lb}^{ii} derived from the isoperimetric inequality Eq. (5.2) followed by the best known solution ℓ_b . The CPU-column gives the time in seconds to compute the length ℓ_m using the monolith version of minDPCH. ℓ_p^0 is obtained by using the polyolithic version of minDPCH which starts with CutDisks yielding input into minSDC, which in turn feeds into minDPCH followed by QuickhullDisk and, finally, MinPerim. Then we see the length ℓ_p^{P1} computed by the polyolithic mode P1 of MinPerim described in KF19. The next column labeled ℓ_p^V displays the convex hull perimeter obtained when feeding the configuration computed by VOROPACK-D (RC) into MinPerim. The last column of Table 7 displays $\ell(\text{KF19})$, the best value reported by KF19 using MinPerim. Figure 11 shows the best configurations for the non-congruent disks problem of SET-B, SET-C, and SET-D.

Some entries in the ℓ_p^V -column, in Tables 7 and 8, show *nsf*, which indicates that VOROPACK-D (RC) does not find a feasible configuration. It does not strictly mean the problem is infeasible, as the Voronoi approach embedded in VOROPACK-D (RC) is a heuristic. We expect this to happen when the target domain has not much more capacity than just

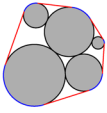
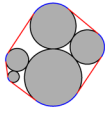
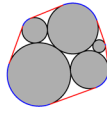
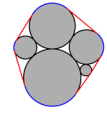
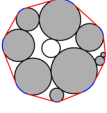
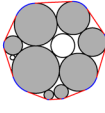
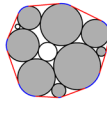
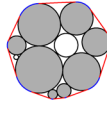
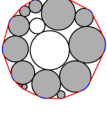
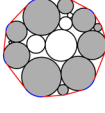
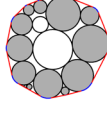
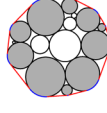
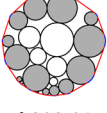
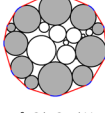
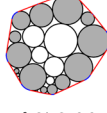
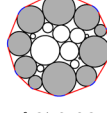
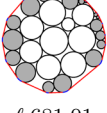
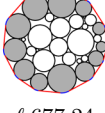
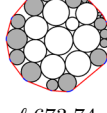
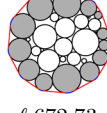
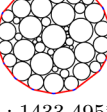
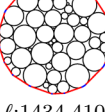
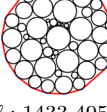
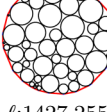
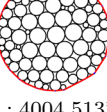
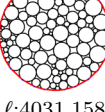
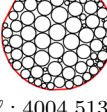
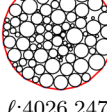
(A)	(B)	(C)	(D)	(E)
n	VOROPACK-D	minRadiusCC	MinPerim(\leftarrow B)	MinPerim(\leftarrow C)
5	 ℓ :53.12	 ℓ :52.99	 ℓ :50.96	 ℓ :50.96
10	 ℓ :138.88	 ℓ :134.60	 ℓ :135.12	 ℓ :134.43
15	 ℓ :245.72	 ℓ :242.60	 ℓ :243.42	 ℓ :242.21
20	 ℓ :383.81	 ℓ :373.45	 ℓ :373.06	 ℓ :372.03
30	 ℓ :681.01	 ℓ :677.24	 ℓ :673.74	 ℓ :672.73
50	 ℓ : 1433.495*	 ℓ :1434.410	 ℓ : 1433.495*	 ℓ :1427.255
100	 ℓ : 4004.513*	 ℓ :4031.158	 ℓ : 4004.513*	 ℓ :4026.247

Fig. 9 Circular container. The arrangements of non-congruent disks of Packomania data SET-P produced by different methods and their convex hull boundaries as follows. Column (B): VOROPACK-D(CC) (This column corresponds to Column B of Table 3). Column (C): minRadiusCC (Column D of Table 3). Column (D): VOROPACK-D(CC) \rightarrow QuickhullDisk \rightarrow MinPerim (Column G of Table 3). Column (E): minRadiusCC \rightarrow QuickhullDisk \rightarrow MinPerim (Column H of Table 3). 1st row: $n=5$. 2nd row: $n=10$. 3rd row: $n=15$. 4th row: $n=20$. 5th row: $n=30$. 6th row: $n=50$. 7th row: $n=100$. ℓ is the length of minimal convex hull boundary which is computed by each method. In all cases except two (Column (D) of $n = 50, 100$), MinPerim improves initial solutions from both VOROPACK-D(CC) and minRadiusCC. Improvements become smaller as the problem size increases. * means no improvement. As shown in Tables 3 and 4, the methods using only NLP and MINLP models do not work for the large problem instances. In this case, the computational geometry algorithm such as VOROPACK-D(CC) could be a good alternative

Table 5 Rectangular domain

A Instance	B \mathcal{C}_{lb}^{ii}	C \mathcal{C}_{lb}^{iv}	D $\Delta^{ii}(\%)$	E $\Delta^{iv}(\%)$	F $\Delta(\%)$	G $\ell_* - \pi$	H CPU	I ℓ_m	J ℓ_p^v	K ℓ_p^i	L $\ell(KF19)$	M Best Arrangement
1 C05*	7.0248	7.5901	15.8980	7.2660	0	5	3457	8.1416	8.1416	8.1416	n.a.	1-2-2
2 C06*	7.6953	8.3267	15.3119	6.5680	0	$4 + \sqrt{3}$	124	8.8736	8.8736	8.8746	8.8736	2-2-2
3 C07*	8.3119	8.9564	9.9821	2.0678	0	6	501	9.1416	10.1416	9.1416	n.a.	2-3-2
4 C08*	8.8858	9.5886	14.1327	5.7673	0	7	695	10.1416	10.1626	10.6057	n.a.	2-3-3
5 C09*	9.4248	10.1817	15.3722	6.7955	0	$6 + \sqrt{3}$	5032	10.8736	10.8736	10.8736	n.a.	3-3-3
6 C10*	9.9346	10.7028	12.1495	4.0999	0	8	100	11.1416	11.1416	11.1416	n.a.	3-4-3
7 C11*	10.4195	11.2372	13.9556	5.6633	0	$7 + \sqrt{3}$	3113	11.8736	12.1295	12.6057	12.6057	2-3-4-2
8 C12*	10.8828	11.7114	11.5669	3.6733	0	9	2870	12.1416	12.1416	12.6875	n.a.	2-3-4-3
9 C13*	11.3272	12.2017	13.6521	5.5066	0	$8 + \sqrt{3}$	58	12.8736	12.8736	12.8736	12.8736	2-4-4-3
10 C14*	11.7548	12.6398	11.7977	3.9700	0	10	63	13.1416	13.4872	13.6057	n.a.	3-4-4-3
11 C15*	12.1673	13.0954	14.0237	5.9425	0	$9 + \sqrt{3}$	8912	14.6057	13.8736	13.8736	n.a.	2-4-4-3-2
12 C16*	12.5664	13.5045	12.5350	4.7177	0	11	7953	14.1416	14.3821	14.6057	n.a.	3-4-3-4-2
13 C17*	12.9531	13.9318	12.7583	4.8371	0	$8 + 2\sqrt{3}$	2248	14.6057	14.6057	14.6067	14.6067	3-4-3-4-3
14 C18*	13.3286	14.3171	13.6023	5.7588	0	12	1211	15.1416	15.1416	16.0698	n.a.	3-4-4-4-3
15 C19*	13.6939	14.6922	13.9610	6.2176	0	$9 + 2\sqrt{3}$	7586	15.6057	16.2800	16.6057	16.6057	2-3-4-3-4-3
16 C20*	14.0496	15.086	14.8901	6.9972	0	13	92	16.1526	16.3377	16.1416	17.1416	2-3-4-4-4-3
17 C30*	17.2072	18.4103	22.4476	14.4457	0	$11 + 4\sqrt{3}$		nsf	21.5339	21.0698	21.5208	
18 C40	19.8692	21.2201	30.8457	22.5159	3.0918	$9 + 8\sqrt{3}$	414	27.9054	26.8018	26.9980	26.9980	
19 C50	22.2144	23.6991	39.5401	30.7982	0.2316	$14 + 8\sqrt{3}$		n.s.f.	31.0698	31.0698	31.6089	

Table 5 continued

A Instance	B C_{lb}^{ii}	C C_{lb}^{IV}	D $\Delta^{ii}(\%)$	E $\Delta^{IV}(\%)$	F $\Delta(\%)$	G $\ell_* - \pi$	H CPU	I ℓ_m	J ℓ_p^V	K ℓ_p^{P1}	L $\ell(KF19)$	M Best Arrangement
20 C75	27.2070	28.9469	59.2182	49.6481	1.6067	$9 + 18\sqrt{3}$		n.s.f.	44.1015	44.0145	44.0974	
21 C85	28.9641	30.7967	66.8220	56.8951	1.5284	$14 + 18\sqrt{3}$		n.s.f.	49.2791	49.0570	49.0570	
22 C90	29.8038	31.6811	70.3897	60.2930	0.974	$13 + 20\sqrt{3}$		n.s.f.	51.6029	51.2772	51.2772	

SET-A. Congruent disks with the radii $R_i = 0.5$. Dimension of the rectangular target domain: width $W = 4$; length $L = 8$ for $n \leq 30$, $L = 14$ for $30 < n \leq 60$, and $L = 25$ for $n > 60$. Column A(Instance): Instances as named in KF19 with the integer denoting the number of disks (e.g., C20 involves 20 disks). $B(C_{lb}^{ii})$: The lower bound derived from the isoperimetric inequality as $C_{lb}^{ii} = \sqrt{\pi}\pi$ using Eq. (5.1). $C(C_{lb}^{IV})$: The Wegner lower bound in Eq. (5.3). $D(\Delta^{ii}(\%))$: The gap between the best solution $G(\ell_*)$ obtained and $B(C_{lb}^{ii})$ using Eq. (5.4). $E(\Delta^{IV}(\%))$: The gap between the best solution $G(\ell_*)$ and $C(C_{lb}^{IV})$. $F(\Delta(\%))$: The gap between the analytic solution $G(\ell_*)$ and $\min\{\ell_m, \ell_p^V, \ell_p^{P1}\}$. $G(\ell_*)$: The best solution (analytic and MILP solution of minLSP are identical up to 9 decimal places). $H(CPU)$: The CPU times in seconds to compute the monolithic length of perimeter $I(\ell_m)$ using minDPCH. $I(\ell_m)$: Monolithic length of the perimeter using minDPCH. $J(\ell_p^V)$: Polyhithic length of the perimeter using VOROPACK-D(RC). $K(\ell_p^{P1})$: Polyhithic length of the perimeter using homotopy P1 in MinPerim. $L(\ell(KF19))$: The length of perimeter reported in KF19. Note that the results $\min\{\ell_m, \ell_p^V\} \leq \ell(KF19)$, i.e., the current approach improves the results by KF19. Small CPU times indicate that ℓ_m has been found during presolve. The numerical solutions of instances marked with an *, e.g., C13*, are identical to the analytic results or semi-analytic ones based on the partitioning model. Computation time limits for each run for each problem instance: 1h for CutDisks and minSDC; 10h for MinPerim and minDPCH. Column M(Best Arrangement): Layer arrangements for congruent disks (C05 through C20). Refer to Fig. 10 for the corresponding disk arrangements. The each number of sequence denotes the number of disks of the best arrangement in layers(columns) from left to right. The layer pattern is enforced by the limiting width $W = 4$ of the rectangular target domain

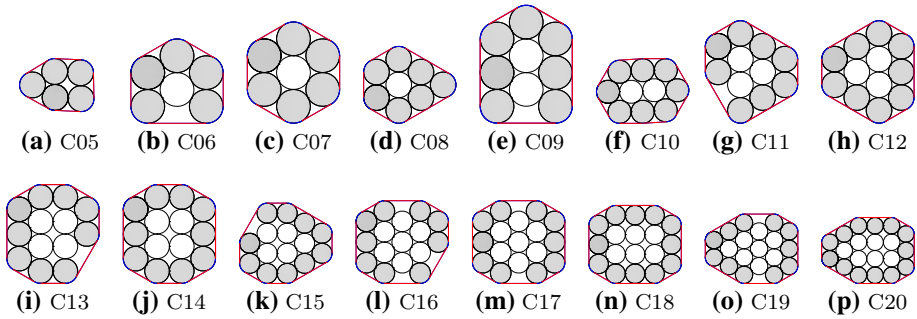


Fig. 10 Congruent disk configurations up to 20 disks. All instances from SET-A

Table 6 Rectangular domain

	A Instance	B \mathcal{L}_{lb}^{ii}	C ℓ_b	D CPU	E ℓ_m	F ℓ_p^0	G ℓ_p^{P1}	H ℓ_p^V
1	DC03*	7.6953	8.5408	400	8.5408	8.5408	8.5408	8.5408
2	DC04	9.9346	10.9376	830	10.9376	10.9376	11.9400	11.2705
3	DC05	8.6036	9.6644	3489	9.6644	9.8771	13.0755	9.9004
4	DC06	9.8096	10.9720	4702	10.9720	10.9720	15.9482	10.9904
5	DC07	10.3004	11.4674	6148	11.4674	11.5387	17.7127	11.7091
6	DC08	11.3272	12.4915	4772	12.4925	12.4915	27.7444	12.4915
7	DC09	11.9628	13.1971	5879	13.1971	13.2369	17.5352	13.1971
8	DC10	12.3685	13.8543	6811	13.8543	13.9611	18.1949	13.9222

Non-congruent disks in SET-B (Data definition in Table 9). A(Instance): Instances. B(\mathcal{L}_{lb}^{ii}): The lower bound of the length of the convex hull perimeter in Eq. (5.1) which is derived from the isoperimetric inequality. C(ℓ_b): The best known solution. D(CPU): The computing time in seconds after the length ℓ_m has been found using the monolith version of minDPCH. E(ℓ_m): The length of the convex hull perimeter obtained by the monolithic version of minDPCH. F(ℓ_p^0): The length of the convex hull perimeter obtained by using the polyolithic version of minDPCH which starts with CutDisks yielding input into minSDC, which in turn feeds into minDPCH followed by QuickhullDisk and, finally, MinPerim. G(ℓ_p^{P1}): The length of the convex hull perimeter computed by the polyolithic mode P1 of MinPerim described in KF19. H(ℓ_p^V): The length of the convex hull perimeter obtained by feeding the configuration computed by VOROPACK-D(RC) into MinPerim. The bold marked entries indicate the smallest perimeter length found for that problem instance. Computation time limits for each run for each problem instance: 1h for CutDisks and minSDC; 10h for MinPerim and minDPCH

hosting the disks to be placed. This weakness can possibly be overcome by connecting it with a metaheuristic such as simulated annealing.

8 Conclusions and outlook

This paper studies solution methods for the minimal convex hull of disks problem which is to find the arrangement of a finite set of 2D circular disks such that the perimeter length of the convex hull of the disks is minimized. In the arrangement, disks are not allowed to overlap each other but may contact. To solve the problem, we have developed a polyolithic framework which combines various NLP models and computational geometry algorithms to provide good initial disk arrangements. These arrangements - the best ones result from

Table 7 Rectangular domain

A Instance	B \mathcal{C}_{lb}^{ii}	C ℓ_b	D CPU	E ℓ_m	F ℓ_p^0	G ℓ_p^{P1}	H ℓ_p^V	I $\ell(KF19)$
1 TC05a	14.8422	16.4723	20883	16.4723	16.4723	17.7532	17.2764	n.a.
2 TC06	16.9413	19.9014	4574	19.9014	20.2734	20.1889	19.9014	n.a.
3 TC06a	8.2881	9.4902	4099	10.1447	9.4902	13.5519	9.5442	n.a.
4 TC06b	9.0181	9.8385	306	10.7365	9.8385	12.2815	10.3867	n.a.
5 TC06c	9.5289	9.8385	7603	9.8385	10.7362	11.3975	10.9750	n.a.
6 TC07	17.3557	20.4319	6148	20.5809	20.4319	20.8287	n.s.f.	20.4378
7 TC08	21.4274	27.7444	1609	28.0904	27.7444	28.5017	28.5017	27.8233
8 TC09	21.7565	28.5821	17371	28.5821	29.5076	28.8927	28.7970	30.7433
9 TC10	23.3495	32.0599	11539	33.0531	32.9176	32.0599	32.2396	32.4350
10 TC20	336.6052	368.6220	5790	368.6370	368.6220	402.9884	394.5359	n.a.
11 TC20a	168.3026	200.5066	9708	200.5870	200.5870	216.9346	205.4742	n.a.
12 TC28	17.3557	21.3314	95	21.3860	21.3314	22.0681	22.0403	42.5991

Non-congruent disks in SET-C (Data definition in Table 10), Column B(\mathcal{C}_{lb}^{ii}): The lower bound derived from the isoperimetric inequality resulting in Eq. (5.1), C(ℓ_b): The best known solution. D(CPU): The computing time in seconds after the lengths ℓ_m has been found using the monolith version of minDPCH. E(ℓ_m): The length of the convex hull perimeter obtained by the monolithic version of minDPCH. F(ℓ_p^0): The length of the convex hull perimeter obtained by using the polyhithic version of minDPCH which starts with CutDisks yielding input into minSDC, which in turn feeds into minDPCH followed by QuickhullDisk and, finally, MinPerim. G(ℓ_p^{P1}): The length of the convex hull perimeter computed by the polyhithic mode P1 of MinPerim described in KF19. H(ℓ_p^V): The length of the convex hull perimeter obtained when feeding the configuration computed by VOROPACK-D (RC) into MinPerim. I($\ell(KF19)$): The best value reported by KF19 using MinPerim. Computation time limits for each run for each problem instance: 1h for CutDisks and minSDC; 10h for MinPerim and minDPCH. The bold marked entries in each row indicate the smallest perimeter length found for that problem instance. n.a.: Not available. nsf: No feasible solution found (Do not know whether the problem is infeasible or not)

Table 8 Rectangular domain

	A Instance	B \mathcal{L}_{lb}^{ii}	C ℓ_b	D CPU	E ℓ_m	F ℓ_p^0	G ℓ_p^{P1}	H ℓ_p^Y
1	D05a	14.8422	16.4723	5367	16.4723	16.7791	16.4723	17.2764
2	D05b	14.8422	16.1530	2985	16.1530	nsf	16.1530	16.9678
3	D06	16.9413	19.9141	1710	19.9141	20.0116	19.9141	20.4138
4	D07	17.3557	20.4319	2525	20.4319	20.4378	20.4319	n.s.f.
5	D08	22.6195	30.6847	8423	32.7080	31.9413	32.7080	30.6847
6	D09	22.9315	28.7999	383	33.3962	31.9413	33.3962	28.7999
7	D10	11.6747	15.4844	7266	16.7094	15.9766	16.7094	15.4844
8	D12	23.9586	26.1780	4089	26.1780	26.3009	26.2187	27.9876
9	D12b	23.9586	30.3580	184	30.3580	30.3836	30.3580	30.9477
10	D14	24.5447	28.8327	1844	28.8486	29.8068	28.8327	29.5064
11	D16	12.7955	15.1224	10118	15.1784	15.1903	15.1224	16.3107
12	D18	29.3432	35.0109	10868	35.0109	35.9798	35.0109	35.6353
13	D21	15.0305	18.1231	17333	18.1231	18.1886	18.1231	18.6061
14	D24a	39.1781	47.5074	20601	47.5074	48.6168	47.5074	56.3276
15	D24b	33.8826	41.8584	15231	41.8584	42.2681	41.8584	41.8729
16	D32	45.2389	57.2146	1689	57.8740	57.2146	57.8740	63.2966

Non-congruent disks in SET-D (Data definition in Table 11). Column B(\mathcal{L}_{lb}^{ii}): The lower bound of the length of the convex hull derived from the isoperimetric inequality in Eq. (5.1). C(ℓ_b): The best known solution. D(CPU): The computing time in seconds after the length ℓ_m has been found using the monolith version of minDPCH. E(ℓ_m): The length of the convex hull perimeter obtained by the monolithic version of minDPCH. F(ℓ_p^0): The length of the convex hull perimeter obtained by using the polyolithic version of minDPCH which starts with CutDisks yielding input into minSDC, which in turn feeds into minDPCH followed by QuickhullDisk and, finally, MinPerim. G(ℓ_p^{P1}): The length of the convex hull computed by the polyolithic mode P1 of MinPerim described in KF19. H(ℓ_p^Y): The length of the convex hull perimeter obtained when feeding the configuration computed by VOROPACK-D (RC) into MinPerim. The bold marked entries in each row indicate the smallest perimeter length found for that problem instance. nsf: No feasible solution found. Computation time limits for each run for each problem instance: 1h for CutDisks and minSDC; 10h for MinPerim and minDPCH

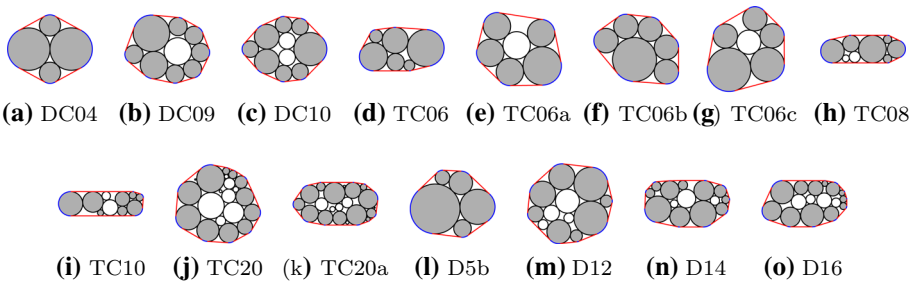


Fig. 11 Configurations for the non-congruent disks problems of SET-B, SET-C, and SET-D. **a, b, c**: from SET-B. **(d through k)**: from SET-C. **(l through o)**: from SET-D

minimizing the discretized perimeter or from minimal circular containers obtained by the `VOROPACK-D` algorithm (which is based on the Voronoi diagram) - have been fed into the `QuickhullDisk` algorithm to construct the convex hull and to compute the length of its perimeter. The output of `QuickhullDisk` is transformed into initial values which are used by `MinPerim` to improve the solution. For up to 1,000 disks, `VOROPACK-D` was used to compute the non-overlapping disk arrangement with convex hulls of almost circular shape. Monolithic and polyhedral solutions using `minDPCH` usually outperform other approaches. Analytic and semi-analytic solutions helped us to verify that the NLP based algorithm and `VOROPACK-D` produce near optimal solutions over a broad range of test cases. It turns out that the polyhedral approach yields better solutions than the results in [17] and the test cases and results could serve as a benchmark suite for further research.

From circular container experiments, we observed that the disk arrangement with minimal circular container radius gave the minimal perimeter convex hull. Thus one of the future researches would be to apply the techniques used in `VOROPACK-D` to the computation of minimal convex hull. Another research path is to extend the current activities to 3D problems, *i.e.*, computing the convex hull of spheres, ellipsoids, and polytopes. We believe that the polyhedral framework can be similarly applied to other hard optimization problems.

Acknowledgements This work was supported by the National Research Foundation of Korea (MSIT) [Nos. 2017R1A3B1023591 and 2016K1A4A3914691].

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

A Notation

We provide the symbols which is introduced in the derivation of the models and is used in the Voronoi diagrams (Appendix B); they are not necessarily used in the models directly.

$VD(P)$	the ordinary Voronoi diagram of a point set P in \mathbb{R}^2 .
$VD(D)$	the Voronoi diagram of a circular disk set D in \mathbb{R}^2 .
$\mathcal{VD}(\kappa, D)$	the Voronoi diagram of a set D of non-intersecting circular disks contained in a container κ
Δ	the difference between the upper and lower bound from the LP relaxation for the MINLP provided by the solver.
$\partial\mathcal{H}$	the perimeter of the convex hull hosting all disks.
\mathcal{H}_i	hyperplane induced by the circular line segment of disk i connecting a pair of incoming (ending) and outgoing (starting) vertices \mathbf{v}_j^{la} and $\mathbf{v}_{j+1}^{\text{al}}$ located on that disk.
\mathcal{H}_j	hyperplane induced by line segment j connecting and tangential to two adjacent disks.
m	the number of disks touching the convex hull; $m \leq n$.
n	the number of disks to be placed.

N_i^{ls}	for disk i , the maximal number of incoming or outgoing line segments on the convex hull boundary.
\mathbf{n}_j^H	the normal vector onto hyperplane \mathcal{H}_j induced by line segment j connecting two disks.
\mathcal{H}	the convex hull hosting all disks.
S_{ij}	specifying whether arc on disk i induced by line segment j is a major ($S_{ij} = 1$) or minor sector ($S_{ij} = 0$); $S_{ij} \in \{0, 1\}$.
\mathbf{v}_j^{al}	vertex connecting a source arc to line segment j ; $\mathbf{v}_j^{al} = (v_{1j}^{al}, v_{2j}^{al})$
\mathbf{v}_j^{an}	vertex connecting a source arc line segment $j + 1$; $\mathbf{v}_j^{an} = (v_{1j}^{an}, v_{2j}^{an})$.
\mathbf{v}_j^{la}	vertex connecting line segment j to a destination arc; $\mathbf{v}_j^{la} = (v_{1j}^{la}, v_{2j}^{la})$.
\mathbf{x}^c	radius-weighted center of all disks.

The symbols used in the explanations of the models are summarized in the following subsections.

A.1 Indices and sets

$d \in \mathcal{D}$	index for the dimension with $\mathcal{D} = \{1, 2\}$; $d = 1$ represents the length-axis, and $d = 2$ the width-axis of the rectangle.
$i \in \mathcal{I}$	objects (disks) to be packed; $\mathcal{I} := \{1, \dots, n\}$.
$j \in \mathcal{J}$	line segments potentially connecting disks and tangential to the convex hull; $\mathcal{J} := \{1, \dots, m \leq N^J \leq n\}$. Note that the number m of active line segments is identical to the number of circular arcs contributed to $\partial\mathcal{H}$.

A.2 Data

E_d	length ($d = 1$) and width ($d = 2$) of the rectangle.
L	length of the rectangle; also called E_1 .
R_i	radius of disk i to be packed.
S_i	indicator specifying to use a major ($S_i = 1$) or minor ($S_i = 0$) sector of disk i contributing an arc to the boundary of the convex hull.
W	width of the rectangle; also called E_2 .

A.3 Decision variables

d_j^H	distance of hyperplane \mathcal{H}_j induced by line segment j to the origin of the coordinate system.
m_{ij}^D	distance of hyperplane \mathcal{H}_{ij} induced by circle segment i to the origin of the coordinate system.
\mathbf{m}_{dij}^H	a specific orthogonal vector onto the <i>circle line segment</i> of disk i connecting \mathbf{v}_j^{la} and \mathbf{v}_j^{an} .
\mathbf{n}_{dj}^H	the normal vector onto hyperplane \mathcal{H}_j induced by line segment j connecting two disks (direction d).
x_d^R	(continuous) extension of the rectangle in dimension d .
x_{id}^0	(continuous) coordinates of the center vector of disk i to be packed.

A.4 Decision variables used in MinPerim

d_j^H	distance of hyperplane \mathcal{H}_j induced by line segment j to the origin of the coordinate system.
m_{ij}^D	distance of hyperplane \mathcal{H}_{ij} induced by circle segment i to the origin of the coordinate system.
\mathbf{m}_{dij}^H	a specific orthogonal vector onto the <i>circle line segment</i> of disk i connecting \mathbf{v}_j^{la} and \mathbf{v}_j^{an} .
n_{dj}^H	the normal vector onto hyperplane \mathcal{H}_j induced by line segment j connecting two disks (direction d).
x_d^R	(continuous) extension of the rectangle in dimension d .
x_{id}^R	(continuous) coordinates of the center vector of disk i to be packed.
α_{ij}	(continuous) sector angle of disk i induced by line segment j .
δ_{ij}	(binary) indicates whether disk i has incoming (ending) or outgoing (starting) line segment j .
δ_j^A	(binary) indicates whether vertices \mathbf{v}_j^{al} and \mathbf{v}_j^{la} are active, <i>i.e.</i> , line segment j is used.
δ_{ij}^D	(binary) indicates whether disk i is the destination of line segment j .
δ_j^L	(binary) indicates whether line segment j is the last active one used.
δ_{ij}^S	(binary) indicates whether disk i is the origin of line segment j .
ℓ	(continuous) length of the convex hull perimeter.
μ	(integer) the number of active line segments.

B Computational geometry basis for the minimal convex hull problems

If the size of a given problem becomes so large that it cannot be solved analytically, semi-analytically, or by mathematical programming alone, we resort to different computational methods. For this purpose, we use two computational geometry algorithms: The VOROPACK-D algorithm for packing circular disks in a container [35], and the QuickhullDisk algorithm for constructing the convex hulls of disk arrangements [26].

The VOROPACK-D algorithm

VOROPACK-D can pack input disks in a circular or rectangular design container by taking advantage of the powerful spatial reasoning capability of the Voronoi diagram of disks in a container. With the Voronoi diagram, VOROPACK-D can locate the vacancy information in a container such that no disk intersects both other disks and the container. In literature, the phi-function [5,31,34,37] and no-fit polygon [2,6,8] were exploited to incorporate the non-overlap condition among disks and container for packing and cutting problems. VOROPACK-D takes an argument denoting the container shape: A circular or rectangular design container. VOROPACK-D(CC) and VOROPACK-D(RC) are for a circular and rectangular container, respectively. The VOROPACK-D(CC) algorithm actually implements the Shrink&Shake algorithm which packs circular disks in a circular container by taking advantage of the Voronoi diagram of disks in the container [35,39]. The method solves a disk packing problem of either congruent or non-congruent disks. The idea of the algorithm is, beginning with a sufficiently large container, to repeat shrinking the container and shaking mutually disjoint disks to reposition in the shrunken container. With the correct implementation of the Voronoi diagram of disks in a circular container, the algorithm is extremely fast

compared to the other reported algorithms. The algorithm, during the shake process, pushes each protruding disk by repositioning every disk at a new position through an average $O(1)$ time decremental and incremental operations from and to the existing Voronoi diagram. With these enhancements, *Shrink&Shake* takes an $O(Mn \log n)$ time for each container shrinkage where $M \leq n$ represents the number of protruding disks which intersect the boundary of the shrunken container. M depends on input data and tends to increase until it reaches some constant as the algorithm iterates. The number of shrinkage also depends on input data. We note that *Shrink&Shake* takes full advantage of the vacancy information among the generators in the container.

In this study, we also use the *VOROPACK-D (RC)* algorithm for packing disks in a rectangular container. *VOROPACK-D (RC)* is designed to handle a rectangular container case using the basic idea of *Shrink&Shake*. *VOROPACK-D (RC)* begins with a sufficiently large container and repeatedly shrink the container and reposition all disks in the shrunken container. Due to the correct implementation of the Voronoi diagram of non-congruent circular disks in a rectangular container, the algorithm can take full advantage of the vacancy information among the generators in the container. Hereafter, if necessary, we will use *VOROPACK-D (CC)* to name an algorithm for packing disks in a circular container and *VOROPACK-D (RC)* for packing disks in a rectangular container instead of *Shrink&Shake*.

The *QuickhullDisk* algorithm

Convex hull is one of the most fundamental concepts in geometry and its construction has been extensively studied, particularly the convex hull of points. Here, we use the recently reported simple and fast *QuickhullDisk* algorithm for the construction of the convex hull of a set of disks in \mathbb{R}^2 by generalizing the quickhull algorithm for points [26]. The *QuickhullDisk* algorithm is a divide-and-conquer algorithm and is based on the idea of the well-known quick sort algorithm. It constructs the convex hull of a disk set D by dividing it into two subsets and quickly conquering the results of the subsets to get the solution of the entire set D . The algorithm recurs until one or two disks are left in the set so that a stopping-condition for a further recursion is encountered. *QuickhullDisk* takes $O(n \log n)$ time on average and $O(mn)$ time in the worst case where m represents the number of extreme disks which contribute to the boundary of the convex hull of n disks. Experimental result shows that the proposed *QuickhullDisk* algorithm runs significantly faster than the $O(n \log n)$ time incremental algorithm, proposed by [7], particularly for big data. *QuickhullDisk* is approximately 2.6 times faster than the incremental algorithm for random disks.

Voronoi diagrams

Voronoi diagrams are powerful geometric constructs which are used to solve diverse problems related with spatial reasoning. We briefly introduce Voronoi diagrams because of their critical uses in the proposed algorithm. For Voronoi diagrams in general, we recommend readers to refer to [1,29]. Hereafter “V-” denotes “Voronoi” for notation simplicity. We limit the discussion in \mathbb{R}^2 unless otherwise stated. We store all Voronoi diagrams in \mathbb{R}^2 in the winged-edge or half-edge data structure which takes $O(n)$ memory for n entities because the Voronoi diagram is a planar subdivision [1,28,29,32]. The geometry of the V-edges in this paper is either linear, parabolic, hyperbolic, or elliptic which are in fact quadratic polynomial curves that can be all represented as a rational quadratic Bézier curve in a unified manner [21].

The ordinary Voronoi diagram of points

The ordinary Voronoi diagram $VD(P)$ of a point set P in \mathbb{R}^2 is a tessellation where each V-cell of the tessellation is a set of locations in the plane which is closer to the associated

point, called a **generator**, in P than to the other generators. Each V-edge is equidistant from two generators, is a subset of a line, and is the boundary between two adjacent V-cells; Some V-edges may be unbounded to emanate to infinity while the others are bounded. Each V-vertex is equidistant from three points. Figure 12a shows an example of $VD(P)$.

In the ordinary Voronoi diagram $VD(P)$ of n point generators in \mathbb{R}^2 , there are $O(n)$ V-vertices, $O(n)$ V-edges, and n V-cells. $VD(P)$ can be constructed in the optimal $O(n \log n)$ time using the divide-and-conquer algorithm [1,29,32]. However, we prefer to use the robust topology-oriented incremental algorithm which was introduced by [38] which takes $O(n)$ time on average (although $O(n^2)$ time in the worst case). Ordinary Voronoi diagrams of approximately 50,000 points in the plane can be robustly constructed in a second on an ordinary desktop computer.

The Voronoi diagram of disks

The Voronoi diagram $VD(D)$ of a circular disk set $D = \{d_1, d_2, \dots, d_n\}$ in \mathbb{R}^2 is a tessellation of the plane so that every location in a V-cell is closer to its generating disk than to other disks. Each V-edge is the locus of the center of circular probe that simultaneously contacts the boundaries of two generating disks: If the generating disks are of different sizes, the V-edge is hyperbolic and if they are of an identical size, it is linear. Hence, they can be all represented as a rational quadratic Bézier curve. The Voronoi diagram of congruent disks is identical to the ordinary Voronoi diagram of disk centers. A V-vertex is the center of circular probe that simultaneously contacts three generating disks. If two generator disks intersect each other, their V-edge passes through the two intersection points between the boundaries of the two disks. Figure 12b shows an example of $VD(D)$.

$VD(D)$ has $O(n)$ V-vertices, $O(n)$ V-edges, and n V-cells and can be constructed by an optimal $O(n \log n)$ time for n disks using the plane sweep method [12,40] or the divide-and-conquer method [24,36]. However, we prefer to use the topology-oriented incremental algorithm which guarantees robustness [25] (or the edge-flipping algorithm [22,23]) for its robust construction. Both algorithms take $O(n^2)$ time in the worst case but $O(n)$ time on average. VD for approximately 15,000 disks can be robustly constructed in a second on an ordinary desktop computer.

The Voronoi diagram of disks in a container

Let $\mathcal{VD}(\kappa, D)$ be the Voronoi diagram of a set D of non-intersecting circular disks contained in a container κ [19]. In this paper, κ is either a circle or a rectangle. We define \mathcal{VD} only in $\partial\kappa$, i.e., the interior of the container. \mathcal{VD} shares many similarities with the Voronoi diagram VD of D but it also has some differences, particularly near $\partial\kappa$.

\mathcal{VD} is a tessellation of the interior of κ , where every location of each V-cell is closer to its generating disk. The container κ itself is regarded as a generator but its interior is considered to be the outside of κ . In other words, the interior of $\partial\kappa$ is regarded as the entire Euclidean space of the outside of κ . Hence, a V-cell can also be well-defined for the container as the set of locations closer to $\partial\kappa$ than to boundaries of any input disks.

If κ is a circle, a rectangle, or a polygon [9,10,20], the V-edge defined between $\partial\kappa$ and an input disk is elliptical or parabolic, respectively. Note that both ellipse and parabola are quadratic. The V-edges between input disks are hyperbolic. Hence, all V-edges can be represented by a rational quadratic Bézier curve [21]. Figure 12c, d shows the examples of $\mathcal{VD}(\kappa, D)$, where κ is a circular and a rectangular container, respectively.

Both \mathcal{VD} and VD can be constructed with a similar efficiency. Even if an optimal algorithm taking $O(n \log n)$ time is known, we prefer to use the topology-oriented incremental algorithm (with an average $O(n)$ time and the worst case $O(n^2)$ time) [25] with the winged-edge data structure. This is because of the guaranteed robustness with a sufficiently good efficiency

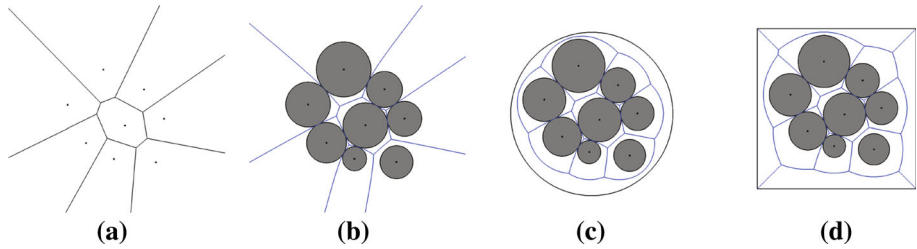


Fig. 12 Voronoi diagrams used in the proposed algorithm. **a** $\text{VD}(P)$: The ordinary Voronoi diagram of a point set P . **b** $\text{VD}(D)$: The Voronoi diagram of a disk set D . **c** $\text{VD}(D, \text{CC})$: The Voronoi diagram in a circular container. **d** $\text{VD}(D, \text{RC})$: The Voronoi diagram in a rectangular container

- actually significantly faster than the optimal algorithm for large problem instances. We skip the details of the combinatorial properties of \mathcal{VD} because they are identical or similar to VD .

Incremental maintenance of Voronoi diagram

Removing a disk $d \in D$ from one location and inserting it to another location, both in the container, is essential to *Shrink&Shake*. If we reconstruct the entire Voronoi diagram for each removal or insertion of d , it takes an optimal $O(n \log n)$ time for each reconstruction. As the removal and insertion of disks occur very frequently in *Shrink&Shake*, it is desirable to do it efficiently. We developed an average $O(1)$ -time (but a worst case $O(n)$ time) algorithms for the maintenance of Voronoi diagram in an incremental manner.

The insertion of a disk into a Voronoi diagram is done as follows. Let \mathcal{VD}_{i-1} be the Voronoi diagram of $i - 1$ disks in the container. We want to compute \mathcal{VD}_i including a new disk d_i . We try to reuse the information in \mathcal{VD}_{i-1} as much as possible. \mathcal{VD}_{i-1} is a planar subdivision: *i.e.*, the network of V-edges of \mathcal{VD}_{i-1} forms a planar graph. The basic idea of the topology-oriented increment is to maintain the planarity of the V-edge graph of \mathcal{VD}_i after the incremental insertion of d_i . Therefore, the topology-oriented increment is to consistently maintain the planarity of \mathcal{VD}_i by (i) identifying a tree subset of V-edge graph of \mathcal{VD}_{i-1} contained in the V-cell of the incrementing disk d_i , (ii) trimming the tree from \mathcal{VD}_{i-1} , (iii) creating new V-vertex(es), V-edge(s), and a new V-face corresponding to d_i , and (iv) properly establishing topology connections among the Voronoi entities remaining in \mathcal{VD}_i . While an insertion (and a delete, too) can be done in $O(i)$ time in the worst case for i disks in the container, its average time complexity is $O(1)$. This average $O(1)$ time holds particularly well during the disk packing process because most disks are in contact with a constant number of other disks on average. For details, see [25].

The removal of a disk from a Voronoi diagram is, roughly speaking, the reverse of the insertion of a disk to a particular location in a given Voronoi diagram and a removal can be done in $O(1)$ on average and in $O(i)$ time in the worst case for i disks in the container. In the incremental insertion, however, identifying the proper location for disk packing in the Voronoi diagram and the bookkeeping after the insertion requires at least $O(\log n)$ time with a priority queue.

C Detailed derivations and proofs

C.1 Importing results from QuickhullDisk

Here we provide analytic expression for importing the result of QuickhullDisk as an initial value to MinPerim. The analytic expression improves the usage of the solver BARON and LINDO, especially, if all variables used in MinPerim are initialized. QuickhullDisk provides a list of *hull disk vertices*, \mathbf{v}_j^{al} and \mathbf{v}_j^{la} . A subset of hull disk is called *extreme disks* in the context of QuickhullDisk. These extreme disks \mathcal{I}_e correspond to *outer disks* of MinPerim. Therefore, QuickhullDisk could provide more vertices than those required in MinPerim because some hull disk vertices can touch the boundary $\partial\mathcal{H}$ of the convex hull \mathcal{H} . In order to cope with this situation we consider those hull disks as well by adjusting $\varepsilon = 0$ in inequality (2.42) of KF19. Vertex \mathbf{v}_j^{al} is the outgoing (starting) vertex and tangential to the arc of that disk from which line segment j leaves. Line segment j ends in an ingoing (ending) vertex \mathbf{v}_j^{la} which is the extreme vertex of the arc of the adjacent outer disk. The arc ends in vertex $\mathbf{v}_{j+1}^{\text{al}}$ which is then the outgoing vertex for line segment $j + 1$. The line segment ends in an incoming vertex $\mathbf{v}_{j+1}^{\text{la}}$ which is the start vertex for the arc of the neighbored outer disk. We continue the construction in anti-clockwise order until line segment $j = m$ ends in vertex \mathbf{v}_m^{la} which is the start vertex of the arc ending in vertex \mathbf{v}_1^{al} . This construction closes $\partial\mathcal{H}$. From \mathbf{v}_j^{al} and \mathbf{v}_j^{la} we derive

$$\delta_j^A = \begin{cases} 1, & \sum_d |\mathbf{v}_j^{\text{al}}| \neq 0 \wedge \sum_d |\mathbf{v}_j^{\text{la}}| \neq 0 \\ 0, & \text{else} \end{cases}, \quad \forall j.$$

Note that for importing the results of QuickhullDisk into MinPerim, at first, we fix all binary variables, *i.e.*, we keep the selected hull disks, arc, and line segments. Once, we have an accepted initial point in LINDO, we relax this fixation. The last active line segment is computed by

$$\delta_j^L = \delta_j^A - \delta_{j+1}^A, \quad \forall j.$$

Whether disk $i \in \mathcal{I}_e$ is the source of line segment j is traced by

$$\delta_{ij}^S = \begin{cases} 1, & \left| \sum_d (x_{id} - \mathbf{v}_j^{\text{al}})^2 - R_i^2 \right| < \varepsilon_R \\ 0, & \text{else} \end{cases}, \quad \forall \{(ij)|i \in \mathcal{I}_e\}$$

which measures whether \mathbf{v}_j^{al} is a point on the circumference of disk i . For numerical purposes we set $\varepsilon_R = 10^{-4}$. Similarly, we proceed for tracing whether disk $i \in \mathcal{I}_e$ is the destination of line segment j

$$\delta_{ij}^D = \begin{cases} 1, & \left| \sum_d (x_{id} - \mathbf{v}_j^{\text{la}})^2 - R_i^2 \right| < \varepsilon_R \\ 0, & \text{else} \end{cases}, \quad \forall \{(ij)|i \in \mathcal{I}_e\}.$$

This allows us to derive

$$\delta_{ij} = \begin{cases} 1, & \delta_{ij}^D > 0.5 \vee \delta_{ij}^S > 0.5 \\ 0, & \text{else} \end{cases}, \quad \forall \{(ij)|i \in \mathcal{I}_e\}.$$

The vertex on the circular arc, which is the source of line segment $j + 1$, is given by

$$\mathbf{v}_j^{\text{an}} = \mathbf{v}_j^{\text{al}} + \mathbf{v}_{v01}^{\text{al}} \delta_j^L, \quad \forall j,$$

where v_{01} denotes the first line segment counted counterclockwise. In case we have more than 99 line segments, the first one would be named v_{001} . Note that it is also necessary to define upper bounds on the variables \mathbf{v}_j^{al} , \mathbf{v}_j^{la} , and \mathbf{v}_j^{an} . For simplicity, we put the dimension of the rectangle as upper bounds (the coordinate frame 1). The normal vector on line segment j (pointing into the interior of the convex hull) is given by

$$n_{dj}^{\text{H}} = - \sum_{i \in \mathcal{I}} \frac{x_{id}^0 - v_{dj}^{\text{al}}}{R_i} \delta_{ij}^{\text{S}}, \quad \forall \{dj\}. \tag{C.1}$$

The distance to the origin is

$$d_j^{\text{H}} = \mathbf{n}_j^{\text{H}} \mathbf{v}_j^{\text{al}}, \quad \forall j. \tag{C.2}$$

The orthogonal vector $\mathbf{m}_{ij}^{\text{H}}$ onto the *circle line segment* of disk i connecting \mathbf{v}_j^{la} and \mathbf{v}_j^{an} is constructed as the vector from the center of disk i to the midpoint of the chord

$$m_{dij}^{\text{H}} = \frac{1}{2} \left(v_{dj}^{\text{an}} + v_{dj}^{\text{la}} \right) \delta_{ij}^{\text{D}} - x_{di}^0 \delta_{ij}^{\text{D}}, \quad \forall \{i, j\}. \tag{C.3}$$

In the half-space inequality we use the negated vector pointing into the interior of \mathcal{H} . If norm $\|\mathbf{m}_{ij}^{\text{H}}\|_2 > 0$, the right-hand side value m_{ij}^{D} of the Hessian normal form $\mathbf{m}_{ij}^{\text{H}} \mathbf{x} = m_{ij}^{\text{D}}$ is computed as

$$m_{ij}^{\text{D}} = \frac{1}{\|\mathbf{m}_{ij}^{\text{H}}\|_2} \sum_d m_{dij}^{\text{H}} v_{dj}^{\text{la}} \delta_{ij}^{\text{D}}, \quad \forall \{i, j\}. \tag{C.4}$$

The angles α_{ij} of the circular arcs follow from

$$\sin \frac{\alpha_{ij}}{2} = \frac{\|\mathbf{v}_j^{\text{an}} - \mathbf{v}_j^{\text{la}}\|_2 \delta_{ij}^{\text{D}}}{2R_i}, \quad \forall \{i, j\} \tag{C.5}$$

and

$$\alpha_{ij} = 2\pi S_{ij} \delta_{ij}^{\text{D}} + 2(1 - 2S_{ij}) \arcsin \left(\frac{\|\mathbf{v}_j^{\text{an}} - \mathbf{v}_j^{\text{la}}\|_2 \delta_{ij}^{\text{D}}}{2R_i} \right), \quad \forall \sum \{i, j\}. \tag{C.6}$$

Finally, we also need to initialize the objective function variable, *i.e.*, the length ℓ of the perimeter of the convex hull

$$\ell = \sum_{j \in \mathcal{J}} \sqrt{\sum_{d \in \mathcal{D}} [v_{dj}^{\text{al}} - v_{dj}^{\text{la}}]^2} + \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} R_i \alpha_{ij}. \tag{C.7}$$

C.2 Computing the length in the partition model

To demonstrate how to compute the length contribution of blocks, let us inspect the solution corresponding to C13 of Table 5 in the main text. This solution is composed by block 3, which is a little complicated, and by block 4 (upside down). If we inspect block 3 as an independent arrangement of six congruent disks with radius R , the length ℓ_3 of its complete perimeter is given by

$$\ell_3 = \ell_L + 2\pi R,$$

Table 9 Non-congruent disks in SET-B (Called Set B in KF19)

Instance	L	W	n	Disk radii	n_D
DC03	8	4	3	$1, 2 \times 0.5$	2
DC04	8	4	4	$2 \times 1, 2 \times 0.5$	2
DC05	8	4	5	$2 \times 0.75, 3 \times 0.5$	2
DC06	8	4	6	$3 \times 0.75, 3 \times 0.5$	2
DC07	8	4	7	$1, 2 \times 0.75, 4 \times 0.5$	3
DC08	8	4	8	$1, 2 \times 0.75, 5 \times 0.5$	3
DC09	8	4	9	$1, 2 \times 0.75, 6 \times 0.5$	3
DC10	8	4	10	$1, 2 \times 0.75, 7 \times 0.5$	3

Instance: instance names. L and W : the length and width of target rectangular domain. n : the number of disks. Disk radii: a list of disk radii. n_D : the number of different-sized disks

where ℓ_L is the sum of lengths of all line segments given by

$$\ell_L = 5 \cdot (2R) + \left\| (R, 4R) - (R + [\sqrt{3}R], R + 3 \cdot 2R) \right\|.$$

The term $\left\| (R, 4R) - (R + [\sqrt{3}R], R + 3 \cdot 2R) \right\|$ represents the length of the line segment from the very right disk of the block’s lower layer with origin at $(R, 4R)$ to the very right disk of its upper layer centered at $(R + [\sqrt{3}R], R + 3 \cdot 2R)$. This length is identical to the distance of these disks. Disks in contact have line segments of length $2R$. Therefore, we have

$$\ell_L = \left[10 + \left\| (-\sqrt{3}, -3) \right\| \right] R = \left[10 + \sqrt{12} \right] R.$$

From this, inspecting the geometry, we further derive

$$\ell_3 = R + (\ell_L + 2\pi R) - (3 \cdot 2R + \pi R) + R.$$

The summands R ’s at the very left and very right are the contributions of a half layer while the term $-(3 \cdot 2R + \pi R)$ reflects that the upper layer of the block does not fully contribute to the length of the perimeter in the partition model; only the lower layer of block 3 and half the upper layer of disks contribute. Therefore, the total contribution of block 3 is

$$\ell_3 = \left[6 + \sqrt{12} + \pi \right] R. \tag{C.8}$$

Similarly, for the upper blocks ℓ_4 and ℓ_5 we obtain

$$\ell_4 = [10 + \pi] R \quad , \quad \ell_5 = \left[2 + \sqrt{12} + \pi \right] R. \tag{C.9}$$

D Tables for input disks

We provide Tables 9, 10, and 11 for non-congruent disks in SET-B (DC03-DC10), SET-C (TC03-TC28), and SET-D (D series) of Sect. 7 in the main text, respectively.

Table 10 Non-congruent disks in SET-C (Called Set C in KF19)

Instance	L	W	n	Disk radii	n_D
TC03	8	4	3	1, 0.6, 0.4	3
TC03a	8	4	3	$2 \times 1, 0.1$	2
TC03b	8	4	3	1, 0.8, 0.1	3
TC03c	8	4	3	$2 \times 2, 1.5$	2
TC04	8	4	4	1.0.4, 0.3.0.2	4
TC04a	8	4	4	$3 \times 1, 0.1$	2
TC04b	16	4	4	2, 1.7, 1.3, 1.2	4
TC04c	8	4	4	1, 1, 1, 1	4
TC05	8	4	5	1, 0.4, 0.3.0.2, 0.1	5
TC05a	8	4	5	1.7, 1.2, 0.8, 0.6, 0.5	5
TC05b	2.0625	2.0625	5	$1, 4 \times (3 - \sqrt{8})$	2
TC06_0	8	4	6	1.7, 1.3, 1.2, 0.8, 0.6, 0.5	6
TC06a	8	4	6	$0.7, 5 \times 0.5$	2
TC06b	8	4	6	$0.9, 5 \times 0.5$	2
TC06c	4	4	6	$0.9, 0.7, 4 \times 0.5$	3
TC07	8	4	7	1.7, 1.3, 1.2, 0.8, $2 \times 0.6, 0.5$	6
TC08	16	4	8	2, 1.7, 1.3, 1.2, 0.8, $2 \times 0.6, 0.5$	7
TC09	20	4	9	2, 1.7, 1.3, 1.2, 0.8, $3 \times 0.6, 0.5$	7
TC10	16	4	10	2, 1.7, $2 \times 1.3, 1.2, 0.8, 0.7, 2 \times 0.6, 0.5$	8
TC20	130	130	20	$R_i = 21 - i$	20
TC20a	80	40	20	$R_i = (21 - i)/2$	20
TC28	8	4	28	$7 \times (1.7, 1.3, 1.2, 0.8, 2 \times 0.6, 0.5)/2$	6

Instance: instance names. L and W : the length and width of target rectangular domain. n : the number of disks. Disk radii: a list of disk radii. n_D : the number of different-sized disks

Table 11 Non-congruent disks in SET-D (Called Set D in KF19)

Instance	L	W	n	Disk radii	n_D
D5b	18	5	5	1.7, 1.2, 0.8, 0.6, 0.5	5
D12	12	8	12	$2 \times (1.7, 1.3, 1.2, 0.8, 0.6, 0.5)$	6
D12b	9	5	12	$2 \times (1.7, 1.3, 1.2, 0.8, 0.6, 0.5)$	6
D14	12	6	14	$2 \times (1.7, 1.3, 1.2, 0.8, 2 \times 0.6, 0.5)$	7
D16	20	8	16	$2 \times (2, 1.7, 1.3, 1.2, 0.8, 2 \times 0.6, 0.5)$	7
D18	14	7	18	$3 \times (1.7, 1.3, 1.2, 0.8, 0.6, 0.5)$	6
D21	7	3.5	21	$3 \times (1.7, 1.3, 1.2, 0.8, 2 \times 0.6, 0.5)/2$	6
D24a	30	9	24	$3 \times (2, 1.7, 1.3, 1.2, 0.8, 2 \times 0.6, 0.5)$	7
D24b	16	8	24	$4 \times (1.7, 1.3, 1.2, 0.8, 0.6, 0.5)$	3

Instance: instance names. L and W : the length and width of target rectangular domain. n : the number of disks. Disk radii: a list of disk radii. n_D : the number of different-sized disks

E Conjecture

Let \mathcal{C} be set of all arrangements of disks fitting into the minimal convex container (with radius r_* in the case of a circular container, otherwise in more general situations with a finite set of variables \mathbf{x}_*^s for rectangular, polygonal, ellipse, or oval containers) and let \mathcal{P} be the set of all arrangements of disks whose convex hull has minimal perimeter with length ℓ_* . In both sets \mathcal{C} and \mathcal{P} we only consider irreducible arrangements, *i.e.*, not translated, rotated or arrangements obtained by symmetry operations.

Then the following statements hold:

ST1: The intersection of \mathcal{C} and \mathcal{P} is not empty, *i.e.*, $\mathcal{C} \cap \mathcal{P} \neq \emptyset$.

ST2: There exists at least one element $\mathbf{c} \in \mathcal{C}$, which is also an element of \mathcal{P} , *i.e.*, $\mathbf{c} \in \mathcal{C} \wedge \mathbf{c} \in \mathcal{P}$.

ST3: There exists at least one element $\mathbf{p} \in \mathcal{P}$, which is also an element of \mathcal{C} , *i.e.*, $\mathbf{p} \in \mathcal{P} \wedge \mathbf{p} \in \mathcal{C}$.

ST4: There exists an arrangement \mathbf{a} , whose convex hull has minimal perimeter ℓ_* and fits into the minimal convex container, $\mathbf{a} \in \mathcal{C} \cap \mathcal{P}$, *i.e.*,

$$\ell_* = \min_{\mathbf{c} \in \mathcal{C}} \ell_{\mathbf{c}}.$$

While the equivalences are obvious, it is not easy to see how to prove one of them easily. ST1 through ST4 basically express that minimal container configurations can be found within the set of minimal perimeter length configurations, and, the other way round, that minimal perimeter length configurations can be found within the set of minimal container configurations.

Note that \mathcal{C} and \mathcal{P} may have the dimensionality of the continuum if smaller disks (named *orphans* in this context) can be placed anywhere into the empty areas between other disks with changing the shape and size of either \mathcal{C} and \mathcal{P} . The other extremes we expect to find are the cases with cardinalities $|\mathcal{C}| = 1$, $|\mathcal{P}| = 1$, or $|\mathcal{C}| = |\mathcal{P}| = 1$. If we neglect the orphans in the arrangements, we almost always found $|\mathcal{C}| = |\mathcal{P}| = 1$, *i.e.*, the minimal perimeter length configuration and the minimal container configuration coincide for the unique arrangement \mathbf{a} . In one case, we found $|\mathcal{C}| = 1$ and $|\mathcal{P}| > 1$ (Refer to Fig. 13).

In the formulation of the conjecture we have implicitly used the assumption that minimal configurations are really realized and not only approached in the sense of the infimum. This assumption is justified by the extreme value theorem on compact sets (Weierstraß). It is important to note that while the disks do not overlap they are allowed to touch in one common point. Without this possibly touching point we would not have a compact set. If the conjecture is true, we have the following two conclusions:

1. In an iterative procedure, we could compute the minimal radius of the container (not necessarily optimal), and add this as a constraint to the minimal perimeter problem extended by the inequalities of the minimal container problem. For the convex hull obtained that way (not necessary optimal), we can compute the minimal radius (very cheaply), and see whether that improves what we have.
2. If we are able to compute the solution of the minimal convex hull problem and know for some reason that it is unique (problems with all disks having different radii are good candidates), all the other specific-type minimal convex container follow immediately from the arrangements of the convex objects within the minimal convex hull. We just need to solve the minimal container problem with the variables describing the container but not the positions of the convex objects hosted. Examples are: The radius $r_{\min R}^{\text{cc}}$ of the

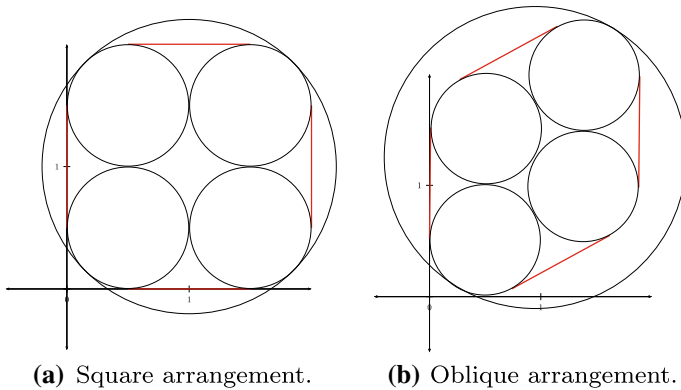


Fig. 13 The left figure shows the square arrangement of four congruent disks with radius $R = 0.5$, $\ell = \ell_* = 4 + \pi$, and circular container with radius 1.2062. The right one: An oblique arrangement of the same disks, the same $\ell = \ell_* = 4 + \pi$ but larger, circular container with radius 1.3608

circular container, the sides a and b of the rectangular container as well as the angle θ of its orientation, the semi-major axis a and b of the ellipse container as well as the angle θ of its orientation.

If the opposite of the conjecture was true, the implication

$$\ell_* < \min_{c \in \mathcal{C}} \ell_c,$$

would hold, *i.e.*, all arrangements with the perimeters of the minimal convex hull are outside the set of minimal container arrangements. Of course, it could also be that for some problem instances we have $\ell_* = \min_{c \in \mathcal{C}} \ell_c$, and for others $\ell_* < \min_{c \in \mathcal{C}} \ell_c$.

Let us focus a little more on the special case of uniqueness. If an arrangement \mathcal{A}_* of disks is the only arrangement which leads to the minimal perimeter length convex hull, then the smallest enclosing circle of \mathcal{A}_* is the minimal enclosing circle of all possible arrangements \mathcal{A} of those disks in the plane. In other words, a unique disk arrangements with the minimal convex hull is also the disk arrangement with minimal circular containers, *i.e.*,

$$\min \ell \Rightarrow \min r_{\min R}^{\text{cc}}.$$

Equivalently, a necessary condition for ℓ to be minimal is that $r_{\min R}^{\text{cc}}$ is minimal. Note that this holds only when the disk arrangement is not subject to any target container constraint. The conjecture does not hold if we allow several arrangements leading to the same minimal ℓ_* as seen in the counter example displayed in Fig. 13. Both have minimal length $\ell_* = 4 + \pi$, but the radius of arrangement in Fig. 13a is $r_{\min R}^{\text{cc}} = 1.2062$ (the global minimum of $r_{\min R}^{\text{cc}}$) while the one in the oblique arrangement (Fig. 13b) has radius $1.3608 > r_{\min R}^{\text{cc}}$.

If this conjecture holds, we can check easily whether a given disk arrangement is not minimal w.r.t. ℓ . For a given disk arrangement with perimeter length ℓ (quickly computed by `QuickhullDisk`) we compute $r_{\min R}^{\text{cc}}(\ell)$ for this disk arrangement using `minRadiusCC` (this problem has only two free variables, $r_{\min R}^{\text{cc}} = r_{\min R}^{\text{cc}}(\ell)$ and x_c , and solves in seconds to global optimality). Then we compare $r_{\min R}^{\text{cc}}(\ell)$ by the value $r_{\min R}^{\text{cc}}$ obtained for the free disk arrangement obtained by `minRadiusCC` or `VOROPACK-D(CC)`. If $r_{\min R}^{\text{cc}}(\ell) > r_{\min R}^{\text{cc}}$, ℓ is not minimal. If $r_{\min R}^{\text{cc}}(\ell) = r_{\min R}^{\text{cc}}$, we cannot conclude anything as the minimal circular container property is not sufficient for ℓ to be minimal.

This conjecture, if true, can also be the basis for an efficient heuristic iteration scheme for larger problem instances in which we are not able to compute $r_{\min R}^{\text{cc}}$ to global optimality by `minRadiusCC`: We plan to develop a simulated annealing enhanced version of `VOROPACK-D(CC)` and subsequently solve for ℓ subject to $r_{\min R}^{\text{cc}}(\ell) \leq r_{\mathbf{V}}^{\text{cc}}$. `VOROPACK-D(CC)` might follow up on a disk arrangement produced by `MinPerim` and further improve w.r.t. to $r_{\min R}^{\text{cc}}$ producing a new value $r_{\mathbf{V}}^{\text{cc}}$. The conjecture motivates us to perform also a few numerical experiments using the discrete perimeter approach with the objective function $z = r_{\min R}^{\text{cc}} + \ell$ and inequality (3.9). For CC05, ..., CC10 of SET-P in Sect. 7.1 of the main text, this simultaneous minimization of $r_{\min R}^{\text{cc}}$ and ℓ yields indeed the minimal circular container and perimeter minimal convex hull. For CC11 and higher this is not the case as the values of $r_{\min R}^{\text{cc}}$ are larger than the Packomania values $r_{\text{Pack}}^{\text{cc}}$. Authors would welcome the opportunity for other researchers to either prove or disprove the conjecture.

References

1. Aurenhammer, F., Klein, R., Lee, D.T.: Voronoi Diagrams and Delaunay Triangulations. World Scientific, Singapore (2013)
2. Bennell, J.A., Oliveira, J.F.: The geometry of nesting problems: a tutorial. *Eur. J. Oper. Res.* **184**(2), 397–415 (2008)
3. Böröczky J.R.K.: Finite Packing and Covering. Cambridge Tracts in Mathematics. Cambridge University Press (2004)
4. Bortfeldt, A.W.G.: Constraints in container loading: a state-of-the-art review. *Eur. J. Oper. Res.* **1**, 1–20 (2013)
5. Chernov, N., Stoyan, Y., Romanova, T.: Mathematical model and efficient algorithms for object packing problem. *Comput. Geom.* **43**(5), 535–553 (2010)
6. Cherri, L.H., Mundim, L.R., Andretta, M., Toledo, F.M.B., Oliveira, J., Carravilla, M.A.: Robust mixed-integer linear programming models for the irregular strip packing problem. *Eur. J. Oper. Res.* **253**(3), 570–583 (2016)
7. Devillers, O., Golin, M.J.: Incremental algorithms for finding the convex hulls of circles and the lower envelopes of parabolas. *Inform. Process. Lett.* **56**(3), 157–164 (1995)
8. Gomes, A.M., Oliveira, J.F.: Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *Eur. J. Oper. Res.* **171**(3), 811–829 (2006)
9. Held, M.: On the Computational Geometry of Pocket Machining. Lecture Notes in Computer Science, vol. 500. Springer, New York (1991)
10. Held, M., Huber, S.: Topology-oriented incremental computation of Voronoi diagrams of circular arcs and straight-line segments. *Comput. Aided Des.* **41**(5), 327–338 (2009)
11. Hifi, M., M'Hallah, R.: A dynamic adaptive local search algorithm for the circular packing problem. *Eur. J. Oper. Res.* **183**(3), 1280–1294 (2007)
12. Jin, L., Kim, D., Mu, L., Kim, D.S., Hu, S.M.: A sweepline algorithm for Euclidean Voronoi diagram of circles. *Comput. Aided Des.* **38**(3), 260–272 (2006)
13. Kallrath, J.: Combined strategic design operative planning in the process industry. *Comput. Chem. Eng.* **33**, 1983–1993 (2009)
14. Kallrath, J.: Polyolithic modeling and solution approaches using algebraic modeling systems. *Optim. Lett.* **5**(3), 453–466 (2011)
15. Kallrath J., Blackburn R., Näumann J.: Grid-enhanced polyolithic modeling and solution approaches for hard optimization Problems. In: Bock H.G., Jäger W., Kostina E., Phu H.X. (eds) Modeling, Simulation and Optimization of Complex Processes HPSC 2018, pp.83–96, Springer, Cham. (2021). https://doi.org/10.1007/978-3-030-55240-4_4
16. Kallrath, J., Frey, M.M.: Minimal surface convex hulls of spheres. *Vietnam J. Math.* **46**(4), 883–913 (2018)
17. Kallrath, J., Frey, M.M.: Packing circles into perimeter-minimizing convex hulls. *J. Glob. Optim.* (2018). <https://doi.org/10.1007/s10898-018-0724-0>
18. Böröczky, K.J., Ruzsa, I.Z.: Note on an inequality of Wegner. *Disc. Comput. Geom.* **37**, 245–249 (2007)
19. Kim, D., Kim, D.S., Sugihara, K.: Euclidean Voronoi diagram for circles in a circle. *Int. J. Comput. Geom. Appl.* **15**(2), 209–228 (2005)

20. Kim, D.S.: Polygon offsetting using a Voronoi diagram and two stacks. *Comput. Aided Des.* **30**(14), 1069–1076 (1998)
21. Kim, D.S., Hwang, I.K., Park, B.J.: Representing the Voronoi diagram of a simple polygon using rational quadratic Bézier curves. *Comput. Aided Des.* **27**(8), 605–614 (1995)
22. Kim, D.S., Kim, D., Sugihara, K.: Voronoi diagram of a circle set from Voronoi diagram of a point set: I. Topology. *Comput. Aided Geom. Des.* **18**, 541–562 (2001)
23. Kim, D.S., Kim, D., Sugihara, K.: Voronoi diagram of a circle set from Voronoi diagram of a point set: II. Geometry. *Comput. Aided Geom. Des.* **18**, 563–585 (2001)
24. Lee, D.T., Drysdale, R.L.: Generalization of Voronoi diagrams in the plane. *SIAM J. Comput.* **10**(1), 73–87 (1981)
25. Lee, M., Sugihara, K., Kim, D.S.: Topology-oriented incremental algorithm for the robust construction of the Voronoi diagrams of disks. *ACM Trans. Math. Softw.* **43**(2), 14:1–14:23 (2016)
26. Linh, N.K., Song, C., Ryu, J., An, P.T., Hoang, N.D., Kim, D.S.: Quickhulldisk: a faster convex hull algorithm for disks. *Appl. Math. Comput.* **363**, 124626 (2019)
27. López, C.O., Beasley, J.E.: A formulation space search heuristic for packing unequal circles in a fixed size circle in a fixed size circular container. *Eur. J. Oper. Res.* pp. 1–10 (2015)
28. Mäntylä, M.: *An Introduction to Solid Modeling*. W.H. Freeman & Company, New York (1988)
29. Okabe, A., Boots, B., Sugihara, K., Chiu, S.N.: *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd edn. Wiley, Chichester (1999)
30. Osserman, R.: The isoperimetric inequality. *Bull. Am. Math. Soc.* **84**(6), 1182–1238 (1978)
31. Pankratov, A., Romanova, T., Litvinchev, I.: Packing ellipses in an optimized rectangular container. *Wireless Netw.* <https://doi.org/10.1007/s11276-018-1890-1> (2018)
32. Preparata, F.P., Shamos, M.I.: *Computational Geometry: An Introduction*. Springer, New York (1985)
33. Rappaport, D.: A convex hull algorithm for discs, an application. *Comput. Geom. Theory Appl.* **3**(1), (1992)
34. Romanova, T., Litvinchev, I., Pankratov, A.: Packing ellipsoids in an optimized cylinder. *Eur. J. Oper. Res.* <https://doi.org/10.1016/j.ejor.2020.01.051> (2020)
35. Ryu, J., Lee, M., Kim, D., Kallrath, J., Sugihara, K., Kim, D.S.: VOROPACK-D: Real-time disk packing algorithm using Voronoi diagram. *Appl. Math. Comput.* **375**, 125076 (2020). <https://doi.org/10.1016/j.amc.2020.125076>
36. Sharir, M.: Intersection and closest-pair problems for a set of planar discs. *SIAM J. Comput.* **14**(2), 448–468 (1985)
37. Stoyan, Y., Pankratov, A., Romanova, T.: Quasi-phi-functions and optimal packing of ellipses. *J. Glob. Optim.* **65**(2), 283–307 (2016)
38. Sugihara, K., Iri, M.: A solid modelling system free from topological inconsistency. *J. Inf. Process.* **12**(4), 380–393 (1989)
39. Sugihara, K., Sawai, M., Sano, H., Kim, D.S., Kim, D.: Disk packing for the estimation of the size of a wire bundle. *Jpn. J. Ind. Appl. Math.* **21**(3), 259–278 (2004)
40. Yap, C.K.: An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. *Dis. Comput. Geom.* **2**, 365–393 (1987)
41. Zeng, Z., Yu, X., He, K., Huang, W., Fu, Z.: Iterated tabu search and variable neighborhood descent for packing unequal circles into a circular container. *Eur. J. Oper. Res.* **250**(2), 615–627 (2016)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.