Applied Mathematics and Computation 375 (2020) 125076

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

VOROPACK-D: Real-time disk packing algorithm using Voronoi diagram

Joonghyun Ryu[a], Mokwon Lee[a], Donguk Kim[c], Josef Kallrath[d], Kokichi Sugihara[e], Deok-Soo Kim[a,b,*]

[a] Voronoi Diagram Research Center, Hanyang University, Republic of Korea
[b] Department of Mechanical Engineering, Hanyang University, Republic of Korea
[c] Department of Industrial Engineering, Gangneung-Wonju National University, Republic of Korea
[d] Department of Astronomy, University of Florida, Gainesville, FL 32611, USA
[e] Meiji Institute for Advanced Study of Mathematical Sciences, Japan

ARTICLE INFO

Article history:
Received 1 November 2019
Accepted 19 January 2020
Available online 12 February 2020

MSC:
00-01
99-00,

Keywords:
Tessellation
NP-hard
Heuristic
Optimization
Computational geometry
Monotone convergence

ABSTRACT

The disk packing problem (DPP) is to find an arrangement of circular disks within the smallest possible container without any overlap. We discuss a DPP for polysized disks in a circular container. DPP is known NP-hard and reported algorithms are slow for finding good solutions even with the problem instances of small to moderate sizes. Here we introduce a heuristic algorithm which finds sufficiently good solutions in realtime for small to moderate-sized problem instances and in pseudo-realtime for large problem instances. The proposed algorithm, VOROPACK-D, takes advantage of the spatial reasoning property of Voronoi diagram and finds an approximate solution of DPP in $O(n\log n)$ time with $O(n)$ memory by making incremental placement of n disks in the order of non-increasing disk size, thus called a big-disk-first method. The location of a placement is determined using the Voronoi diagram of already-placed disks. If needed, we further enhance a big-disk-first realtime packing solution using the Shrink-and-Shake algorithm by taking an additional $O(Mn^2)$ time for each shrinkage where $M \ll n$ is the number of protruding disks for each shrinkage. Experimental results show that the proposed algorithm is faster than other reported ones by several orders of magnitude, particularly for large problem instances. Theoretical observations are verified and validated by a thorough experiment. This study suggests that Voronoi diagram might be useful for solving other hard optimization problems related with empty spaces. VOROPACK-D is freely available at Voronoi Diagram Research Center (http://voronoi.hanyang.ac.kr/software/voropack-d).

1. Introduction

Suppose that we are given a set $D = \{d_1, d_2, \ldots, d_n\}$ of disks where each disk $d_i = d_i(c_i, r_i) \in D$ is associated with center $c_i(x_i, y_i) \in \mathbb{R}^2$ and radius $r_i \in \mathbb{R}$. We want to find the smallest circular container $d_0 = d_0(c_0, r_0)$ which contains all disks in D

* Corresponding author at: School of Mechanical Engineering, Hanyang University, Republic of Korea.
E-mail address: dskim@hanyang.ac.kr (D.-S. Kim).

https://doi.org/10.1016/j.amc.2020.125076

without any overlap except on their boundaries. Hence, we want to determine the center c_0 and radius r_0 of the container together with the disk arrangement, *i.e.*, the coordinates of the disk centers $c_1, c_2, \ldots, c_n$. The *disk packing problem (DPP)* discussed in this paper can be formulated as an optimization problem as follows.

Formulation 1 (DPP-Polysized-Circular-container (DPP-PCc) Problem)**.**

$$\textit{Minimize } r_0 \tag{1}$$

Subject to

$$(x_i - x_j)^2 + (y_i - y_j)^2 \geq (r_i + r_j)^2 \tag{2}$$

$$(x_0 - x_i)^2 + (y_0 - y_i)^2 \leq (r_0 - r_i)^2 \tag{3}$$

$$r_0 \geq 0 \tag{4}$$

where $i \neq j,\ i,\ j = 1, 2, \ldots, n$.

Inequality (2) prevents that two disks overlap in more than one point. They are allowed to touch each other in one point. Inequality (3) enforces all disks to be contained within the container. Formulation 1 has $\frac{n(n+1)}{2} + 1$ constraints and $2n + 3$ decision variables. The number of decision variables can be reduced to $2n + 1$ by enforcing c_0 (or c_i of any $d_i \in D$) to coincide with the coordinate origin. The formulation has a linear objective function with one variable over nonlinear inequalities each defining a segment of the quadratic boundary of the non-convex feasible solution space.

DPP is NP-hard even if we ignore the rotational variety of disk arrangement [1]. There can be frequently infinitely many alternative optimal solutions. For example, consider $D = \{d_1, d_2, d_3\}$ where d_1 and d_2 are equal-sized and d_3 is tiny. Suppose that the optimal container d_0 is the circle containing both d_1 and d_2 and is tangent to d_1 and d_2 where they are in a tangential contact. Then, it is possible that the tiny d_3 can rattle in the interstitial region between d_1 and d_2 within the container, thus called a rattler. The rattler may take any one among infinitely many locations in the region. See Section 2 for literature review.

In this paper, we present a heuristic algorithm VOROPACK-D for solving DPP. The algorithm finds sufficiently good solutions of the problem instances of small to moderate size in realtime and those of large size in pseudo-realtime, *i.e.* very efficiently. The idea is simple as follows. With a sufficiently large container d_0, the algorithm increments disks in the order of non-increasing size, thus called the big-disk-first method. For each increment, we make sure that the constraints of Eqs. (2) and (3) are satisfied. We use the Voronoi diagram of polysized circular disks which are contained within the initial container. Taking advantage of the empty space information available from the Voronoi diagram, the algorithm finds a good location for an incrementing disk. As there are $O(i)$ vertices and edges in the Voronoi diagram of i disks, we can find an appropriate location for $(i+1)$th disk in $O(i)$ time using a straightforward linear scan of edges. After all disks are placed, the minimum enclosing circle can be found in $O(n)$ time. Hence, a good packing of n disks can be easily found in $O(n^2)$ time using the big-disk-first heuristic with the Voronoi diagram. We present the accelerated big-disk-first method to $O(n \log n)$ time using two priority queues: One for maintaining interstitial voids among disks in a sorted order of clearance and the other for maintaining the Voronoi edges in the free space (which is called the infinite void later) in a sorted order of distance from the center of the container. An example of the performance is as follows. Let A1, A2, and A3 be the sets with 50, 100, and 1000 disks where the ith disk has radius i, respectively; B1, B2, and B3 be those with 50, 100, and 1000 disks where the ith disk has radius $i^{-\frac{1}{2}}$, respectively. VOROPACK-D, which implements the $O(n^2)$ time algorithm, finds the packings of A1, A2, B1, and B2 in the solution containers in 18, 32, 12, and 35 msec where the container radii are 5.6%, 5.1%, 5.1%, and 3.3% larger than the best-known minimum container, respectively. It finds the packings of A3 and B3 in 757 and 806 msec where the packing densities are 0.8539 and 0.8940, respectively and the container radii 19782.500 and 2.894, respectively. Note that no known packing results are reported for A3 and B3. If it is necessary or desirable, the realtime packing solution of VOROPACK-D can be enhanced by the Shrink-and-Shake (S&S) method reported in 2004 in [2]. The accelerated S&S algorithm developed in this paper uses the Voronoi diagram of disks and takes an additional $O(Mn^2)$ time for each shrinkage where $M \ll n$ is the number of protruding disks for a shrunken container.

We would like to emphasize that the idea of the proposed algorithm for DPP using Voronoi diagram can be used to solve other NP-hard problems whose solutions are related with empty Euclidean spaces. This geometry-priority approach based on Voronoi diagram might introduce a new paradigm for designing heuristic algorithms of hard problems, particularly for large problem instances related with empty space in 2- and 3-dimensional Euclidean spaces.

The contributions of this paper are as follows.

- Introduction of a realtime disk packing algorithm VOROPACK-D.
- Introduction of Voronoi diagram to optimization problems which are related with empty space among particles.
- VOROPACK-D program which implements the proposed algorithm: It is freely available at the Voronoi Diagram Research Center (http://voronoi.hanyang.ac.kr/software/voropack-d).

All discussions are in the plane and all time complexities are in the worst case sense unless otherwise stated. We distinguish “disk” from “circle”. “Disk” is an input circular object to pack whereas “circle” is a derived object to be used for designing algorithm. Note. In literature, there exists another context of “circle packing problem” different from the problem

we discuss in this paper. In [3–5], a circle packing is a configuration of circles realizing a specified pattern of tangencies: E.g. Apollonian circle packing [6–8]. In this regard, we strictly distinguish the use of "disk" from "circle" in this paper.

The organization of this paper is as follows. Section 2 presents a literature review. Section 3 briefly presents the Voronoi diagram which is the most fundamental computational tool for the proposed algorithm. Section 4 presents the computational building blocks to design the proposed algorithm for solving disk packing problems. Section 5 presents the big-disk-first realtime disk packing algorithm, together with two others, for finding the solution of DPP in realtime. Section 6 presents the enhancement of the packing solution obtained by the big-disk-first realtime algorithm using the 2004 Shrink-and-Shake algorithm, together with its acceleration using the Voronoi diagram. Section 7 presents the experimental result with discussions. Then, the paper concludes.

2. Literature review

The disk packing problem is an old, important and hard problem, and derives its roots perhaps from the Tenth Problem of Apollonius of Perga. There are abundant prior works for packing disks. We recommend the following for information sources: (i) Good reviews [9,10]; (ii) The Packomania web site for benchmark problems with the best-known results (http://www.packomania.com); (iii) Al Zimmermann's programming contest in 2005 for best disk packing in a circular container (See http://recmath.com/contest/CirclePacking/index.php).

Two fundamental considerations in packing problems are the profile of disk sizes and the shape of the target container. For example, all disks can be of an equal radii, some may have equal radii but others unequal radii, all disks have distinct radii, the disk set consists of a mixture of only two (or three, four, etc.) different radii, etc. The target container shape may be a circle, a rectangle, an ellipse, a convex polygon (such as square, rectangle, or triangle), or a non-convex polygon.

To the best of our knowledge, the disk packing problem began with a convex polygonal container by Segre and Mahler in 1944 [11]. Even if we are interested in packing disks in a circular container, we also briefly review articles addressing relaxed problems.

Suppose that we want to place $p > 0$ congruent disks in a container of a unit square without any overlap between disks. What is the maximum disk radius $r > 0$? This problem can be formulated as the following.

Formulation 2 (DPP-Congruent-Square-container (DPP-CSc) Problem)**.**

$$\textit{Maximize } r \tag{5}$$

Subject to

$$(x_i - x_j)^2 + (y_i - y_j)^2 \geq 4r^2 \tag{6}$$

$$0 \leq x_i \leq 1, \quad 0 \leq y_i \leq 1, \quad i = 1, 2, \ldots, p \tag{7}$$

If we replace Eq. (7) with Eq. (3), we have a DPP-Congruent-Circular-container (DPP-CCc) Problem. If we replace Eq. (7) with

$$c_i \subset \Omega, \quad i = 1, 2, \ldots, p \tag{8}$$

where Ω is a polygon (either convex or non-convex) and $c_i = c_i(x_i, y_i)$ is the center of a disk $d_i(c_i, r_i)$, we have a DPP-Congruent-Polygon-container (DPP-CPc) Problem.

2.1. Congruent disks

Disk packing naturally began with congruent disks without any consideration of container. For congruent disks, if no constraint about container exists, a hexagonal packing with the kissing number six (also called as a honeycomb packing; every disk contacts six others of an identical size) is optimal (*i.e.*, the densest) with the packing density of $\pi/\sqrt{12} = 0.9069\ldots$. This seemingly obvious optimality of congruent disk packing problem was proved by Thue, initially in 1892 and the proof was later improved in 1910 [12,13] and eventually completed by Toth in 1940 [14].

With a container, the disk packing problem was first discussed with congruent disks w.r.t. a convex polygon in 1944 (DPP-CPc Problem) [11] and followed by studies w.r.t. a unit square container (DPP-CSc Problem) by Schaer in 1965 [15] and a circular container (DPP-CCc Problem). Then, discussions have been extended to more generalized models such as DPP-PCc, DPP-PPc, etc.

There were three major approaches: (i) Analytic approach (to find analytically the optimal or good packing of a small number of disks); (ii) Simulation approach (to use simulation for a large number of disks). (iii) Nonlinear programming (NLP) approach (to formulate a NLP optimization problem to find good packing of a large number of disks);

The analytic approach was the first appeared one. To the best of our knowledge, Schaer in 1965 was the first to report packing k congruent disks within a unit square in $\mathbb{R}^2$, $2 \leq k \leq 9$ [15]. In 1966, Schaer reported packing k congruent spheres within a unit cube for $k \leq 9$ in $\mathbb{R}^3$ [16]. In 1967, Kravitz reported the problem of finding the smallest possible circular container (we call it an *optimal container*) which can contain congruent disks and presented packings up to 19 disks (without optimality proofs) [17]. Graham in 1968 proved the optimality of the packing of up to 7 disks [18]. Goldberg, in

1971, presented packings of 14, 16, 17, and 20 disks [19]. Reis, in 1975, presented up to 25 disks including an improvement of Goldberg's result [20]. Melissen, in 1994, proved the optimality for the case of eleven disks [21]. In 1991, Dowsland reported a study of rules to compactly pack many congruent disks in pallet arrangements [22]. Fraser and George in 1994 [23] discussed packing congruent disks in a container of fixed size. Dowsland addressed the optimal size package [22,24]. Isermann in 1991 [25] outlined heuristics for packing congruent disks.

The simulation approach began in 1990 by Lubachevsky and Graham, with coworkers, using a bucket-based algorithm to simulate billiards with hard elastic balls moving on the plane and used to analyze the packing patterns of disk sets of moderate to big size [26–28]. The idea of the algorithm was to start with a set of random points with random velocities and to grow them at a uniform rate so that particles eventually jammed. The algorithm was able to solve congruent disk packing problems for many disks. Graham and Lubachevsky used the billiards algorithm to study packings of congruent disks in an equilateral triangular container [29,30], rectangular container [31,32], and circular container [33,34]. They even looked at an opposite problem of finding the minimum perimeter rectangles that enclose congruent non-overlapping disks [35]. The benchmark report by Gavrilova et al. among the algorithms to detect the contacts between polysized disks using dynamic power diagram and various subdivision methods is noteworthy [36].

Three important studies of the NLP approach were reported in 1995. First, Maranas, Floudas, and Pardalos reported an NLP formulation to maximize the minimum distance between all pairs of points within a unit square container [37]. Hence, this problem is equivalent to maximize the radius of the disks in DPP-CSc Model. They presented the packing results up to 30 disks (with neither optimality claim nor computation time statistics) using GAMS for modeling and the MINOS solver for computation. They used multiple initial points in order to span most of the parameter space because the solver provided no theoretical guarantee of the convergence to the global optimum.

Second, Drezner and Erkut introduced the continuous p-dispersion problem in a square and showed its equivalence to the problem of packing p disks in a square [38]. The problem was formulated to maximize the minimum distance between pairs of points within a container. They solved this problem for $p = 10, 11, \ldots, 23$ and were able to duplicate the best-known results. They used AMPL for modeling and used MINOS 5.4 as a nonlinear solver and solved each of the problems 1000 times with a different initial solution each time. They reported the time statistic for one case. They also solved the packing problem in a circular container (DPP-CCc Model) by replacing the constraint in Eq. (7) with

$$x_i^2 + y_i^2 \le (1 - r)^2 \quad \text{for} \quad i = 1, 2, \ldots, p \tag{9}$$

Third, George et al. reported the problem of fitting circular pipes of different diameters into a shipping container (*i.e.*, DPP-PSc Problem) [39]. They formulated the problem as a nonlinear mixed integer programming problem and reported the solutions (without time statistics) by a genetic algorithm based on several heuristic rules. They used two heuristic rules based on geometric observation to make a room for another disk: "spin-out" (to move existing disks to container boundary as far as possible) and "shake-down" (to move existing disks downward as much as possible).

Graham and Lubachevsky, together with coworkers in 1998 [34], presented a nonlinear optimization formulation to maximize the minimum pairwise distance between the points in the unit circle: Max Min$\{|c_i - c_j| : 1 \le i < j \le n\}$ where $C = \{c_1, c_2, \ldots, c_n\}$ is the set of disk centers in the container with a unit radius.

2.2. Polysized disks

From mathematical programming modeling point of view, the difference between packing polysized disks and congruent ones is to replace Eq. (6) with Eq. (2) (together with a slight variation of container constraint). We believe that the first explicit NLP formulation of a polysized disk packing problem was, as explained above, by George et al. in 1995 for a square container (*i.e.*, DPP-PSc) [39]. Huang in 1999 presented an NLP formulation of a polysized disk packing problem w.r.t. a circular container (*i.e.*, DPP-PCc) [40]. They looked at the same problem from the opposite viewpoint of the earlier works by attempting to minimize the total overlapping regions among disks and container using the gradient-based local search to reduce the overlap to zero. A rule for escaping from a local minimum was to choose a disk with the most overlapped distance and to move that to some other region within the container. Since then, Huang's group reported several heuristic observations such as corner placement, maximal hold degree, the self look-ahead strategy, and vacant degree [41–46]. Other studies followed: Simulated annealing with tabu search [47], NLP formulations [48,49], etc.

In 2007, Hifi and M'Hallah [50] presented an incremental algorithm that increments a new disk d_i, $i \le n$, in a greedy manner to an arrangement of already packed $i - 1$ disks so that d_i kept tangency with at least two disks. Before the actual placement of ith disk, all candidate two-tangency locations are identified and for each candidate location, the minimal container was found by solving another nonlinear minimization problem. Among the containers, the minimal one is selected. The minimum enclosing circle for n disks with a fixed arrangement needs to be mentioned. There are $O(n^3)$ enclosing circles because three disks may define a unique enclosing circle. As the inclusion test should be done for each of the $n - 3$ disks, the time complexity is $O(n^4)$ without a clever method such as [51,52]. This group extended the incremental algorithm by exploring different ways to increment each disk so that more solution space can be searched, of course finding better solution in the cost of computation time [53] and further studied the influence of limiting the number of children of each node in the search tree [54].

2.3. Explicit modeling of empty space in container

In 2004, Sugihara and Kim, together with coworkers, introduced the idea of using Voronoi diagrams for an explicit modeling of empty space in container to efficiently pack disks (We skip its detail as it is explained in Section 5). Wormser in 2008 used power diagram for an application of packing polysized disks [55]. Lu and coworkers in 2011 used power diagram in a similar effort for packing disks [56]. Specht in 2016 used the geometric property of the interstitial region in disk arrangement which is conceptually similar to the Voronoi diagram [57]. At each iteration, the algorithm generates a contact graph where a vertex corresponds to a disk and an edge corresponds to either two disks or a disk and container. The contact graph is used to determine the structure of interstitial region and to calculate the clearance of the interstitial regions. The contact information was modeled as a planar graph which was represented by the half-edge data structure that allows an efficient query. In the higher level, the algorithm adopted an iterative improvement based on a method similar to the billiards simulation algorithm [26]. The iterations were repeated until the temperature gradually cools down to a given numerical tolerance. When a jamming occurs during the process, annealing methods such as jumping disks, swapping two disks, or rotating three or more disks were attempted. The algorithm was efficient, largely due to the active use of geometric properties of disk arrangement. The trend of using more geometric properties is clear in several recent studies [44,46,50,54,58].

2.4. Generalized packing problems

It is interesting that the first reported study of a disk packing problem with a container was by Segre and Mahler in 1944 about the upper bound of the number of congruent disks in a convex polygon [11]. Since then, many reports followed on DPP-CSc or DPP-PSc problems [22,39,59–63]. Recent works include more generalized problems: Polysized disk packing into perimeter-minimizing convex hulls [64], congruent disk packing within a non-convex container defined by free-form curves [65], etc.

The problem has been even more generalized. Packing ellipses of different shapes into a rectangular container of minimal area [66–68] or into a circular container [69,70] and packing ellipsoids into three-dimensional containers of rectilinear or spherical shapes [71,72]. The packing of ellipses and ellipsoids has important applications in wireless communication, understanding matter [73–75], modeling explicit forces among elliptical particles [76], 3D printing [77], etc. It is noteworthy that ellipsoidal particle packing in $\mathbb{R}^3$ has recently significant applications for understanding matter [73,78], granular materials [79,80], geosystems [73], pore/tunnel analysis for tissue repair and regeneration to study jamming behavior of hydrogel microparticles [81].

Interestingly enough, the earliest possible reference to a packing problem is perhaps the *Kepler conjecture* about the packing of spherical balls in $\mathbb{R}^3$ stated in 1611 in his "De Nive Sexangula (On the six cornered snowflake)" as follows: No packing of identically-sized balls in $\mathbb{R}^3$ has density greater than that of the face-centered cubic packing (which is $\pi/\sqrt{18} = 0.74048\ldots$). This number occurs in FCC (face-centered cubic) and HCP (hexagonal closed-pack). More than 400 years later, Thomas Hales and coworkers in 2015 reported a proof of Kepler conjecture [82]. To be precise, they proved $\pi/\sqrt{18}$ is the highest packing density for congruent 3D spherical balls and in addition to the FCC packing, there are uncountably many packing configurations with the same density. Hales launched a worldwide cooperative project "Flyspeck," which was completed in August 2014, to prove his own proof by himself and Hales and 21 coworkers published "A formal proof of Kepler conjecture in 2017 [82]." Packing of higher-dimensional spheres is also being studied [83].

3. Voronoi diagram

Voronoi diagrams are powerful geometric constructs which are used to solve diverse problems related with spatial reasoning. We briefly introduce a necessary minimum of Voronoi diagrams with proper references because they are extensively used by VOROPACK-D. For Voronoi diagrams in general, we recommend readers to refer to [84,85]. Hereafter "V-" denotes "Voronoi" for notation simplicity.

Ordinary Voronoi diagram of points. The ordinary Voronoi diagram of a point set P is a tessellation where each V-cell of the tessellation is a set of locations in the space which is closer to the associated point, called a **generator**, in P than to the other generators. Each V-edge is equidistant from two points in P, 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.

In the ordinary Voronoi diagram of n point generators in $\mathbb{R}^2$, there are $O(n)$ V-vertices, $O(n)$ V-edges, and n V-cells and can be constructed in the optimal $O(n\log n)$ time using the divide-and-conquer algorithm with a winged-edge or half-edge data structure taking $O(n)$ memory [84–87]. However, we prefer to use the topology-oriented incremental algorithm which was introduced by Sugihara and Iri in 1989 [88] because it guarantees robustness with $O(n)$ time on average (although $O(n^2)$ time in the worst case). The ordinary Voronoi diagram of approximately 50,000 points in the plane can be robustly constructed in a second on an ordinary desktop computer.

The Voronoi diagram of circular disks. The Voronoi diagram VD of a circular disk set $D = \{d_1, d_2, \ldots, d_n\}$ in the plane 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-

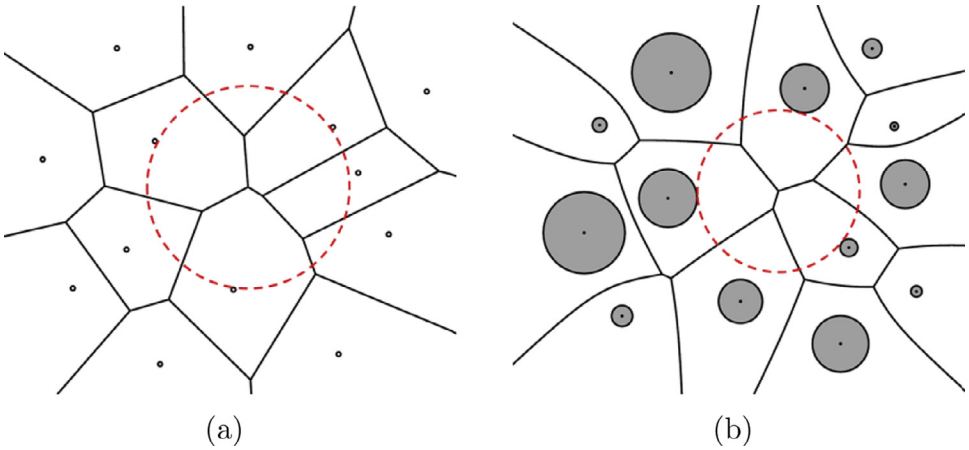

Fig. 1. Voronoi diagrams in $\mathbb{R}^2$. The red dotted circles are the biggest empty circles among generators which are found in $O(n)$ time for n input generators. (a) The ordinary Voronoi diagram of points. (b) The Voronoi diagram of disks. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

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. Some V-edges may be unbounded to emanate to infinity while the others are bounded. The Voronoi diagram of congruent disks is identical to the ordinary Voronoi diagram of disk centers. V-edges can be conveniently represented by rational quadratic Bézier curve because it can define all quadratic polynomial in a unified manner [89]. 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. If they contact each other, the corresponding V-edge passes through the contact point.

VD 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 [90,91] or the divide-and-conquer method [92,93]. However, we prefer to use the topology-oriented incremental algorithm [94] (or the edge-flipping algorithm [95,96]) with the winged-edge data structure (taking $O(n)$ memory) [86,87,97] for its robust construction. Both algorithms take $O(n^2)$ time in the worst case but $O(n)$ time on average but with guaranteed robustness. VD for approximately 15,000 disks can be robustly constructed in a second on an ordinary desktop computer. Hence, we can construct the Voronoi diagram of 100 disks approximately 150 times/sec. Our experience well confirms this estimation.

Fig. 1 (a) shows an example of the ordinary Voronoi diagram of points and the (red dotted) largest empty circle found in $O(n)$ time for n input points by scanning the V-vertices. Fig. 1(b) is an example of the Voronoi diagram of disks and the biggest empty circle, also found in $O(n)$ time for n input disks.

The Voronoi diagram of disks within a circular container. Let $\mathcal{VD}$ be the Voronoi diagram of a set D of non-intersecting circular disks contained within a circular container d_0 [98]. We define $\mathcal{VD}$ only within ∂d_0, *i.e.*, the interior of the container. $\mathcal{VD}$ shares many similarities with the Voronoi diagram VD of D but it also has some dissimilarities.

$\mathcal{VD}$ is a tessellation of the interior of the container where every location of each V-cell is closer to its generating disk. The container d_0 itself is regarded as a disk generator but its interior is considered to be the outside of d_0. In other words, the interior of ∂d_0 is regarded as the entire Euclidean space of the outside of d_0. Hence, a V-cell can also be well-defined for the container as the set of locations closer to ∂d_0 than to any input disks. The V-edge defined between ∂d_0 and an input disk is elliptic (which is also a quadratic curve). The V-edges between input disks are hyperbolic. So, the rational quadratic Bézier curve representation of V-edges holds good for all V-edges of $\mathcal{VD}$ [89]. Both $\mathcal{VD}$ and VD can be constructed with a similar efficiency. Even if an optimal algorithm is known, we prefer to use a slight variation of the topology-oriented incremental algorithm (with an average $O(n)$ time and the worst case $O(n^2)$ time) [94] with the winged-edge data structure because of the guaranteed robustness with a sufficiently good efficiency - actually significantly faster than the optimal algorithm for large problem instances. The variation is simply to consider the elliptic V-edge between a disk and the container. We skip the details of the combinatorial properties of $\mathcal{VD}$ because they are identical or similar to VD.

Fig. 2 shows examples of $\mathcal{VD}$. Fig. 2(a) shows $\mathcal{VD}$ of six points within a container: The V-edges between the point generators are linear whereas the V-edges between a point generator and the container is elliptic. Fig. 2(b) shows $\mathcal{VD}$ of three points and three disks of different sizes within a container: The V-edges between the disks are hyperbolic whereas the V-edges between the disks and the container are elliptic. Fig. 2(c) shows $\mathcal{VD}$ of six disks with tangential contacts in a container with three voids: The **infinite void** defined by the container and the input disks, a second one defined by five disks, and a third, tiny one defined by three disks. The second and third ones are **interstitial voids** which are defined by mutually tangent disks.

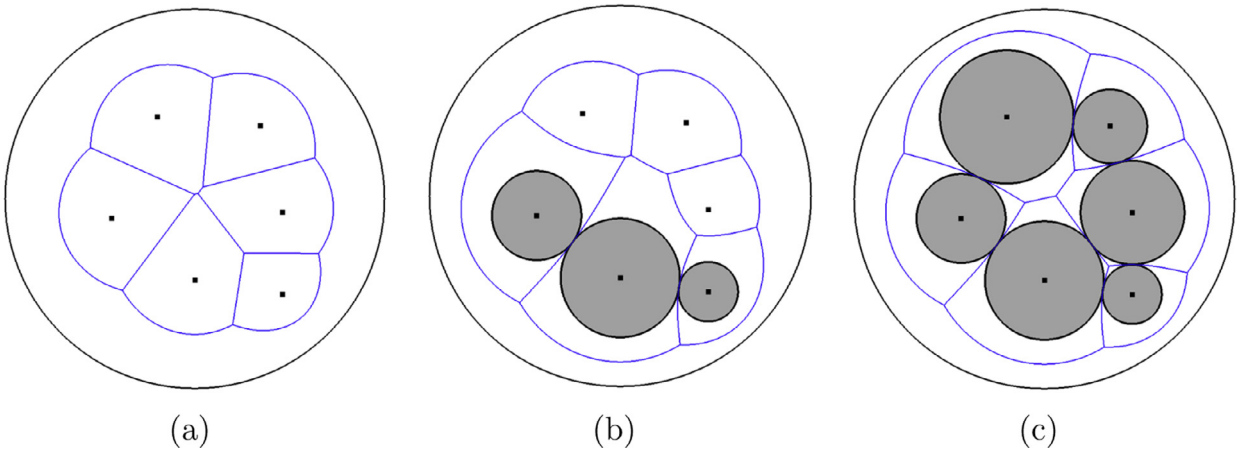

Fig. 2. Examples of the Voronoi diagram $\mathcal{VD}$ in a circular container. (a) Voronoi diagram of six points within a container. V-edges between point generators are linear whereas the V-edges between a point generator and the container is elliptic. (b) Voronoi diagram of three points and three disks within a container. V-edges between disks are hyperbolic where as the V-edges between the disks and container are elliptic. (c) Voronoi diagram of six disks in a container with three voids: One infinite void and two interstitial ones.

Lemma 1. *Let $\mathcal{VD} = \mathcal{VD}(V, E, C)$ be the Voronoi diagram of n disks within a container where V, E, and C are the sets of V-vertices, V-edges, and V-cells, respectively. Then, $|V| = |E| = O(n)$ and $|C| = n$.*

Decrement/increment operations. Removing a disk $d \in D$ from one location and inserting it to another location, both within the container, is essential to the accelerated S&S algorithm. If we reconstruct the entire Voronoi diagram for each removal of 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 the accelerated S&S algorithm, it is desirable to do it efficiently. We thus developed and implemented an average $O(1)$-time decrement and a worst case $O(\log n)$-time increment algorithms. In a conceptual description, removing a disk from Voronoi diagram is the reverse of inserting a disk to a particular location in a given Voronoi diagram (which is well-described in [94]), but with a slightly more complicated tasks. Both can be done in $O(1)$ on average. However, in the increment, identifying the proper location in the Voronoi diagram and the bookkeeping after the insertion requires $O(\log n)$ time with a priority queue. In this paper, we will only provide an overview of the disk incremental algorithm of the topology-oriented algorithm (We avoid to describe the details because it belongs more to computational geometry). See [94] for details (The idea was introduced by Sugihara and Iri [88]).

Let $\mathcal{VD}_{i-1}$ be the Voronoi diagram of $i-1$ disks. 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 to save computation. The network of V-edges of $\mathcal{VD}_{i-1}$ (in fact, any Voronoi diagrams in the plane) forms a planar graph (which can be embedded in the plane without any crossing edge). The basic idea of the topology-oriented construction is to maintain the planarity of the V-edge graph of $\mathcal{VD}_i$ after incrementing d_i. Therefore, the topology-oriented incremental construction 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.

4. Computational building blocks of Voronoi diagram for disk packing problems

Given the Voronoi diagram $\mathcal{VD}$ of disks within a container, we define the concepts of clearance and void as computational building blocks for disk packing problems.

4.1. Clearances

Let $dist(x, y)$ be the Euclidean distance between two points $x, y \in \mathbb{R}^2$. Let $Dist(x, Y) = \inf_{y \in Y} dist(x, y)$ where $x \in \mathbb{R}^2$ and $Y \subset \mathbb{R}^2$. Given a V-vertex $v \in \mathbb{R}^2$ and a disk $d \subset \mathbb{R}^2$, $Dist(v, d)$ is the shortest Euclidean distance between v and d and is hereafter referred to as **distance**.

A V-vertex has an identical distance to its three generating disks by definition. Hence, a **circular probe** of this size can be placed at the V-vertex while it simultaneously contacts the generating disks.

Definition 1 (V-vertex Clearance)**.** The distance between a V-vertex v and its generating disks is the **clearance** ξ of v.

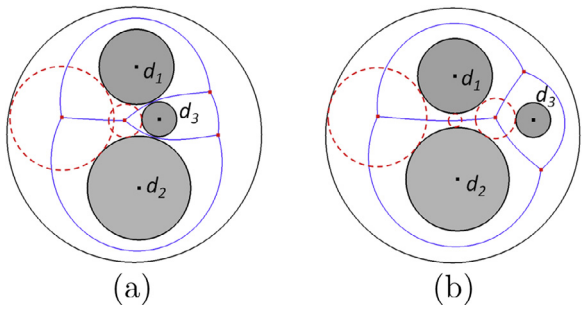

Fig. 3. Minimum and maximum clearances of a hyperbolic V-edge e defined by d_1 and d_2. (a) Both occur at the V-vertices of e. (b) While the maximum clearance occurs at a V-vertex, the minimum clearance occurs at an interior location of e. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

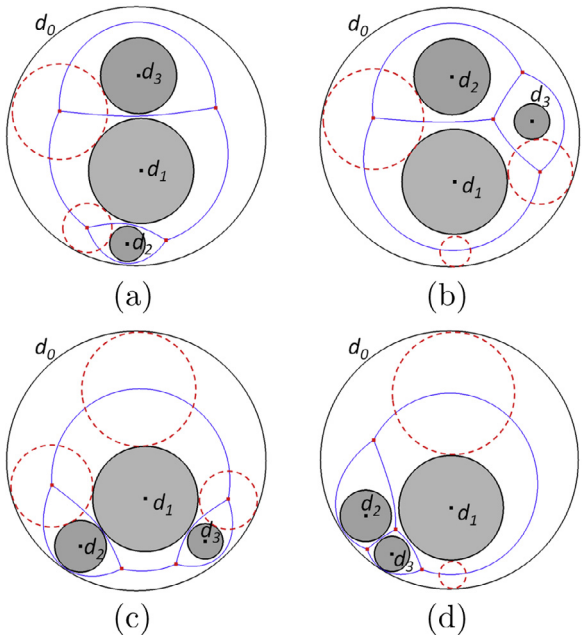

Fig. 4. Minimum and maximum clearances of an elliptic V-edge e defined by d_0 and d_1 (d_0: the container). (a) Both occurs at V-vertices. (b) The maximum occurs at a V-vertex but the minimum at an internal location of e. (c) The minimum at a V-vertex but the maximum at an internal location of e. (d) Both at internal locations of e.

A V-vertex has a clearance $\xi \in [0, r_0)$ where r_0 is the radius of the container $d_0(c_0, r_0)$. In Fig. 3(a), the two red dotted circles centered at V-vertices define the clearances of the V-vertices.

A V-edge e is a locus of the center of the circular probe that simultaneously contacts the boundaries of two generators. In $\mathcal{VD}$ for the accelerated S&S algorithm, two generators may contact but otherwise do not intersect each other.

Definition 2 (V-edge Clearance)**.** The radius of the minimum (or maximum) circular probe centered on a V-edge e which simultaneously contacts a pair of generators is the **minimum clearance ξ_{min}** (or maximum clearance ξ_{max}) of e. The V-edge e has a **clearance interval** $[\xi_{min}, \xi_{max})$.

Lemma 2. *A hyperbolic V-edge has a clearance interval* $[\xi_{min}, \xi_{max})$ *where* $0 \le \xi_{min} \le \xi_{max} < r_0$ *for the container radius* r_0.

In Fig. 3(a), the V-edge between d_1 and d_2 is hyperbolic and both ξ_{min} and ξ_{max} occur at V-vertices. On the other hand, in Fig. 3(b), the hyperbolic V-edge between d_1 and d_2 has ξ_{max} at one of its V-vertices but ξ_{min} at an interior location on the V-edge (*i.e.*, not at the other V-vertex). The following lemma obviously holds.

Lemma 3. *The maximum clearance of a hyperbolic V-edge occurs at one of its V-vertices. The minimum clearance may occur either at a V-vertex or at an interior location of the V-edge.*

Consider the elliptic V-edge defined between d_1 and the container d_0 in Fig. 4. In Fig. 4(a), both ξ_{min} and ξ_{max} occur at the V-vertices. In Fig. 4(b), ξ_{max} occurs at a V-vertex but ξ_{min} occurs at an interior location of the V-edge. In Fig. 4(c), ξ_{min}

occurs at a V-vertex but ξ_{max} occurs at an interior location of the V-edge. In Fig. 4(d), both ξ_{min} and ξ_{max} occur at interior locations. The following lemma holds.

Lemma 4. *The minimum and maximum clearances of an elliptic V-edge may occur at any locations of the V-edge.*

The location where the minimal (or maximal) clearance occurs is called the **minimal (or maximal) clearance location**.

Lemma 5. *An elliptic V-edge has a clearance interval* $[\xi_{min}, \xi_{max})$ *where* $0 \le \xi_{min} \le \xi_{max} < r_0$ *for the container radius* r_0.

Assuming $\mathcal{VD} = \mathcal{VD}(V, E, C)$ stored in the winged-edge (or half-edge) data structure [86,87], the clearance of a given V-vertex or the clearance interval of a given V-edge can be computed in $O(1)$ time because the corresponding generators can be retrieved in $O(1)$ time. The clearance information does not need to consider V-edge geometry. The following lemma holds because $|V| = |E| = O(n)$.

Lemma 6. *Given* $\mathcal{VD}$ *for a set D of n disks, the clearances of all V-vertices and the clearance intervals of all V-edges can be computed in* $O(n)$ *time.*

4.2. Voids

We want to find the interstitial region of free space among disks. Consider $d_i \in D$ as a set of locations: *i.e.*, $d_i = \{(x, y) \mid (x - x_i)^2 + (y - y_i)^2 \le r_i^2\}$ and $D = \cup_i d_i$. Let $\mathcal{V} = \{\mathcal{V}_0, \mathcal{V}_1, \ldots\}$ be a set of connected component where $\mathcal{V}_i \subset d_0$ and $d_0 \setminus D = \bigcup_i \mathcal{V}_i$. Note that we intentionally abuse notation for the sake of presentation convenience.

Each connected component $\mathcal{V}_i \in \mathcal{V}$ is called a **void**. A void which contacts the container boundary is particularly called an **infinite void**. Otherwise a void is **interstitial**. The information about all voids, both interstitial and infinity, is readily available in $\mathcal{VD}$. A void defined by three disks (when one of which can be the container) is associated with one and only one V-vertex and is called a **triangular void**. Fig. 2(c) shows an infinite void and two interstitial voids where the tiny one is triangular. Observe that the triangular void has one V-vertex whereas the bigger interstitial void has three V-vertices.

Recall that $\mathcal{VD} = \mathcal{VD}(V, E, C)$ where *V, E,* and *C* are the sets of V-vertices, V-edges, and V-cells in $\mathcal{VD}$ for *n* disks, respectively.

Lemma 7. *There are* $|V| = O(n)$ *voids in the container.*

Consider a graph $G = G(V, E)$. Suppose that we **subdivide** each V-edge e into two V-edges, say e' and e'', at the point $p \in e$ if the generators of e contact to each other at p. In such a case e also contacts both generators at p. To subdivide, we create and insert a new V-vertex of degree two at p in the winged-ede data structure. Hence, we have an **expanded data structure** $\widehat{\mathcal{VD}} = \widehat{\mathcal{VD}}(\widehat{V}, \widehat{E}, C)$ where $\widehat{V}$ and $\widehat{E}$ are the expanded sets of V-vertices and V-edges by the subdivision, respectively. $V \subseteq \widehat{V}$ and $E \subseteq \widehat{E}$. $\widehat{G} = \widehat{G}(\widehat{V}, \widehat{E})$ is called an **expanded graph** by the subdivision.

Lemma 8. $|\widehat{V}| = |\widehat{E}| = O(n)$.

Consider a void $\mathcal{V}_i$. Let $G^{\mathcal{V}_i} = G^{\mathcal{V}_i}(V^{\mathcal{V}_i}, E^{\mathcal{V}_i}) \subset \widehat{G}$ where $V^{\mathcal{V}_i} = \widehat{V} \cap \mathcal{V}_i$ and $E^{\mathcal{V}_i} = \widehat{E} \cap \mathcal{V}_i$. In other words, $G^{\mathcal{V}_i}$ is the subgraph of the expanded graph which is contained within the void $\mathcal{V}_i$. Then, $G^{\mathcal{V}_i}$ is a connected graph which can be used to get the clearance information of $\mathcal{V}_i$.

Definition 3 (Void Clearance)**.** The maximum of the clearances of both all V-vertices and V-edges in a void is called the **(maximal) clearance of a void**.

A void clearance defines the maximum possible circular disk/probe that can be placed in the void.

Lemma 9. *An interstitial void has the maximum clearance at a V-vertex.*

Proof. The V-edges of an interstitial void are hyperbolic and a hyberbolic v-edge has the maximum clearance at a V-vertex due to Lemma 3. □

The maximum clearance of the infinite void can occur on a V-vertex or a V-edge.

Definition 4. (Free space in voids; See Fig. 5) Let $\mathcal{V}$ be a void with the maximal clearance ξ_{max}. Suppose that a disk $d = d(c, r)$ is given to insert into $\mathcal{V}$ where $0 \le r \le \xi_{max}$. Let $\mathcal{P}^{\text{offset}} \subseteq \mathcal{V}$ be an offset polygon of $\mathcal{V}$ whose boundary is determined by the enlarged disk generators and by the shrunken container generators with the offset amount r. The vertex of $\mathcal{P}^{\text{offset}}$ is called a 2-contact location, a point on an edge of $\mathcal{P}^{\text{offset}}$ is called a 1-contact location, and an interior point of $\mathcal{P}^{\text{offset}}$ is called a 0-contact location. If $\mathcal{P}^{\text{offset}}$ degenerates to a point, the point is called a 3-contact location.

If we place the disk d at a k-contact location, d contacts with k generators, $k = 0, 1, 2,$ and 3. Fig. 5 shows the relationship between the new (white) disk d to insert and the existing (shaded) disks via the Voronoi diagram. $\widehat{d_0}$ corresponds to a 0-contact location; $\widehat{d_1}$ corresponds to an 1-contact location and its center is on neither V-edge nor V-vertex; $\widehat{d_2}$ and $\widehat{d_2'}$ correspond to 2-contact locations and their centers are located on the V-edges; $\widehat{d_3}$ corresponds to a 3-contact location and its center is located at a V-vertex.

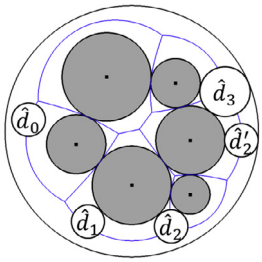

Fig. 5. Placement of a new disk at different locations. The shaded disks are existing ones and the white ones represent the locations of a new incrementing disk. $\hat{d}_0$ corresponds to a 0-contact location; $\hat{d}_1$ corresponds to an 1-contact location (Center is not on a V-edge); $\hat{d}_2$ and $\hat{d}_2'$ correspond to 2-contact locations (Centers are on a V-edge); $\hat{d}_3$ corresponds to a 3-contact location (Center coincides with a V-vertex).

Lemma 10. *If a disk d is placed at the k-contact location in a void $\mathcal{V}$, $k = 0, 1, 2, 3$, $\mathcal{V}$ changes its topological state due to the placement of d as follows:*

- *0-contact location: The genus of $\mathcal{V}$ changes from g to $g+1$, $g = 0, 1, 2, \ldots$.*
- *1-contact location: No change.*
- *2-contact location: $\mathcal{V}$ with m V-edges subdivides into two smaller voids where one has three V-edges and the other has $m+1$ V-edges.*
- *3-contact location: $\mathcal{V}$ subdivides into three smaller voids.*

Given $\mathcal{VD}$, $\mathcal{P}^{\text{offset}}$ can be computed in $O(n)$ time [99]. Hence, the following lemma holds.

Lemma 11. *Given the $\mathcal{VD}$ of a disk set D and a disk d to insert, all 2-contact locations can be computed in $O(n)$ time for n generators.*

Lemma 12. *(State transition of voids) Let p be a maximal clearance location of a void $\mathcal{V}$ with the clearance ξ_{max}. Placing a maximal disk d (with the radius $r_d = \xi_{max}$) at $p \in \mathcal{V}$ causes a state change of $\mathcal{V}$ as follows.*

(i) *If $\mathcal{V}$ is an infinite void with a genus g and if p has two generators where one of the two is the container, $\mathcal{V}$ changes its genus from g to $g-1$.*
(ii) *If $\mathcal{V}$ is an infinite void with a genus zero and if p has two generators where one of the two is the container, $\mathcal{V}$ subdivides to two voids.*
(iii) *Otherwise, $\mathcal{V}$ subdivides to three voids.*

5. Big-disk-first realtime disk packing algorithm

Here we present a heuristic realtime algorithm, the big-disk-first algorithm, for quickly finding a good solution of DPP by taking advantage of the Voronoi diagram. Suppose that $D = \{d_1, d_2, \ldots, d_n\}$ is sorted in the non-increasing order of radii: *i.e.*, $r_i \geq r_j$ if $i < j$. For notational convenience, we slightly abuse D to denote both a disk set and an arrangement of the disks in the plane (*i.e.*, the placement of the disks is determined). Let $D^+ = \{d_0\} \cup D$ be a feasible solution of DPP, *i.e.* D and d_0 satisfy the inequalities in Eqs. (2–4). Let $D_*^+ = \{d_0^*\} \cup D^*$ be a good solution found by an algorithm as an approximation of the optimal solution $D_{Opt}^+ = \{d_0^{Opt}\} \cup D^{Opt}$ of DPP.

The big-disk-first realtime disk packing algorithm is very simple as follows. We first place the center of d_1 at the origin of the coordinate system and place the center of d_2 on the positive X-axis so that d_2 has a tangential contact with d_1. We assume a sufficiently large container d_0 centered at the origin. Then, we construct the Voronoi diagram of d_1, d_2, and d_0. For the remaining disks $d_i \in D$, $i = 3, 4, \ldots, n$, we repeat the following: i) Compute the 2-contact locations (*i.e.*, corner points) of d_i at all possible V-edges; ii) Choose the one, say λ, nearest to the coordinate origin among all of the 2-contact locations; iii) Insert d_i at λ and update the Voronoi diagram. Step i) and ii) take $O(i)$ time for ith disk. This is because the Voronoi diagram with i disks is a planar graph with $O(i)$ V-vertices and V-edges and the computation of all 2-contact locations requires $O(i)$ time by scanning the V-edges once. Step iii) takes $O(1)$ time by employing the topology-oriented incremental algorithm. Hence, the big-disk-first method takes $O(n^2)$ time for n disks. After all input disks are incremented, we determine the smallest possible container by finding the minimum enclosing circle of the disks using the $O(n)$ time algorithm in [52]. Algoirthm 1 shows the pseudocode of the $O(n^2)$ time big-disk-first packing method. This proves the following lemma.

Lemma 13. *Big-disk-first packing can be done in $O(n^2)$ time for n disks.*

Fig. 6(a) shows an example of the big-disk-first packing of 100 disks. Observe the three tiny disks in the interstitial voids around the biggest disk in the center. There can be other heuristic methods: Random-sequence (Fig. 6(b)), small-disk-first methods (Fig. 6(c)), etc. are simple variations of the big-disk-first method with modified ordering of disks. All methods use the same algorithm, but with different ordering of disks. Other variations can also be easily devised using the Voronoi diagram.

Algorithm 1: The $O(n^2)$ big-disk-first disk packing algorithm.

Input : Input disks in a sorted set D
Output: Optimal disk packing$D_*^+(D^*, d_0^*)$

Step 1. [Initialization] Assume a sufficiently large container d_0; Place d_1 and d_2 in the center of d_0;
Step 2. [Main Loop: Increment disk] For each $d_i \in D$, $i = 3, 4, \ldots, n$, do the following;
  Step 2.1. Compute all 2-contact locations on Voronoi edges.;
  Step 2.2. Find the 2-contact location λ which is nearest to the origin of the coordinate system;
  Step 2.3. Place the disk d_i at λ and update the Voronoi diagram;
Step 3. [Packing completed] Compute the minimum enclosing circle as d_0^* and report the packing D_*^+.Terminate.;

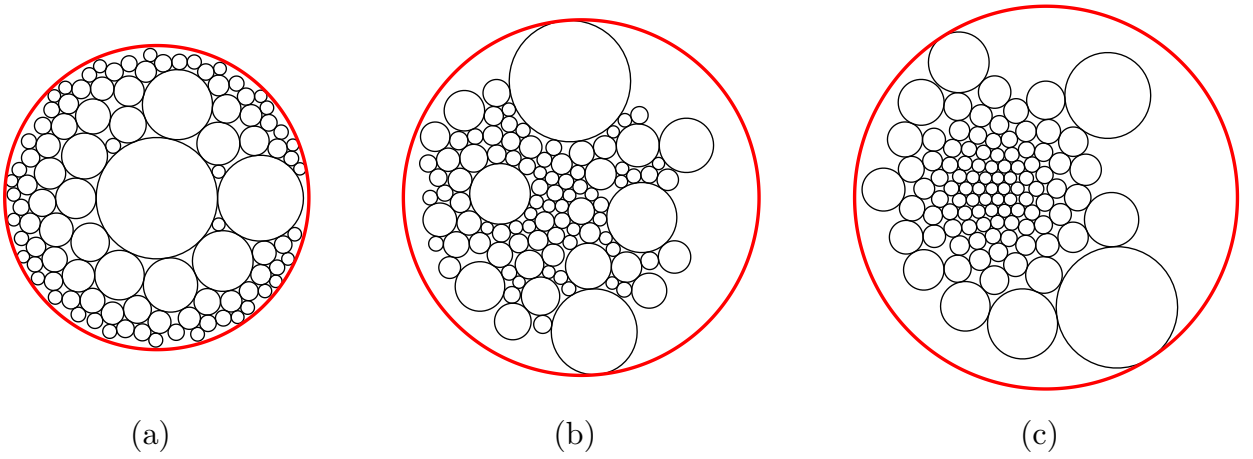

Fig. 6. The packings of a set of 100 disks. $r_i = i^{-\frac{1}{2}}$. (a) Big-disk-first packing (Container radius: 2.506). (b) Random-sequence packing (Container radius: 3.149). (c) Small-disk-first packing (Container radius: 3.217).

We improve the quadratic time complexity by using two priority queues as follows. When a disk d is to be placed in the container, d can be placed either in an interstitial void or in the infinite void (which we also call a free space). Once the placement of d is made, the information of the clearances of influenced V-vertices, V-edges, and voids must be updated. We observe that the placement of d causes changes which are local in a particular void and such changes can be best maintained by two priority queues: $Q^{\mathcal{V}}_{interstice}$ for the interstitial voids and $Q^{VE}_{infinite}$ for the V-edges in the infinite void. $Q^{\mathcal{V}}_{interstice}$ maintains the interstitial voids in the non-increasing order of clearance: *i.e.*, the void of the root node has the biggest clearance. $Q^{VE}_{infinite}$ maintains the V-edges in the infinite void in the non-decreasing order of the distance from the container center: *i.e.*, the V-edge of the root node is closest from the center.

The idea of the big-disk-first packing is to place d as near the coordinate center as possible. The algorithm first checks $Q^{\mathcal{V}}_{interstice}$ if d can be placed in an interstitial void v_{root} corresponding to the root node. If it can be placed, the algorithm finds a 2-contact location λ in v_{root}. The rationale is that the bigger an interstitial void is in the big-disk-first method, the closer it is located ot the coordinate center. The increment of d at λ divides v_{root} into two smaller voids. Then, we insert or relocate each of the subdivided voids in $Q^{\mathcal{V}}_{interstice}$ with a modified key value of the void clearance. The insertion or relocation can be done in $O(\log n)$ time because there can be at most $O(n)$ nodes in $Q^{\mathcal{V}}_{interstice}$. The search for all voids in $Q^{\mathcal{V}}_{interstice}$ with a particular range of key values can be done in $O(\log n + k)$ time for k found results. Hence, variations of the big-disk-first method can be devised depending on which one of the searched void is used for the placement of d. In this paper, we insert d in the void with the biggest clearance, thus the biggest void, which can be found in $O(1)$ time from the root of $Q^{\mathcal{V}}_{interstice}$. There can be different strategies to place d which eventually lead to different packing methods.

Suppose that d cannot be placed in the void corresponding to the root node of $Q^{\mathcal{V}}_{interstice}$. Then, d needs to be placed in the infinite void. In this case, the algorithm places d at a 2-contact location in the infinite void which is closest to the container center. This can be done by checking if the center of d can be placed at a location λ on the V-edge e which corresponds to the root node of $Q^{VE}_{infinite}$. If e has such a λ, d is placed there. Otherwise, we traverse $Q^{VE}_{infinite}$ downward to find a V-edge which provides a 2-contact location. As there are at most $O(n)$ V-edges, the search takes $O(\log n)$ time. Algoirthm 2 shows the pseudocode of the $O(n \log n)$ time big-disk-first packing method. This proves the following theorem.

Theorem 14. *The big-disk-first packing algorithm takes $O(n \log n)$ time using a priority queue of an $O(n)$ memory where n is the number of disks.*

Algorithm 2: The $O(n\log n)$ big-disk-first disk packing algorithm.

Input : A set D of ordered disks
Output: A good disk packing solution $D_*^+(D^*, d_0^*)$

Step 1. [Initialization] Assume a sufficiently large container d_0.Place d_1 and d_2 in contact around the center of d_0.Add d_1 and d_2 in D^*.Construct the Voronoi diagram $\mathcal{VD}$ of d_1 and d_2 in d_0.Create the priority queue $Q_{interstice}$ and $Q_{infinite}$.;
Step 2. [Main Loop: Increment disk] For each $d_i \in D$, $i = 3, 4, \ldots, n$, do the following.;
  Step 2.1. If d_i can be placed in the interstitial void v corresponding to the root node of $Q_{interstice}$, do the following.;
    Step 2.1.1. Compute the 2-contact location λ on the corresponding V-edge in v.;
    Step 2.1.2. Insert d_i at λ and update $\mathcal{VD}$, $Q_{infinite}$, and $Q_{interstice}$.;
  Step 2.2. Otherwise, d_i has to be placed in the infinite void at a V-edge e corresponding to the root node of $Q_{infinite}$.Do the following.;
    Step 2.2.1. Compute the 2-contact location λ on e.;
    Step 2.2.2. Insert d_i at λ and update $\mathcal{VD}$, $Q_{infinite}$, and $Q_{interstice}$.;
  Step 2.3. Add d_i in in D^*.;
Step 3. [Packing completed] Compute the minimum enclosing circle as d_0^* and report the packing solution D_*^+.Terminate.;

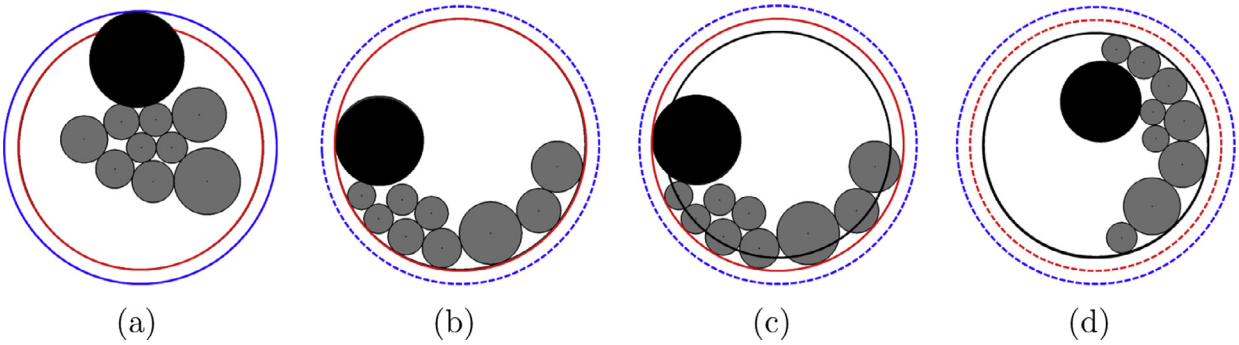

Fig. 7. Two consecutive strokes of the Shrink-and-Shake algorithm with 10 disks. (a) An initial packing D_{1st}^+ with the initial container (the blue circle). The black filled disk is protruding the red shrunken container and will play the role of pivot disk. (b) An improved packing D_{2nd}^+. The repositioned disks within the shrunken container after the first stroke of shrinking and shaking is completed. The pivot disk is also pushed in the shrunken container. (c) The second shrink operation and the protruding disks. The black circle is the shrunken container and the black filled disk is chosen as the second pivot disk. (d) A further packing improvement D_{3rd}^+ after the second stroke is completed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6. Enhancement of realtime packing using Shrink-and-Shake

The packing solution $D_*^+(D^*, d_0^*)$ produced by the big-disk-first realtime algorithm can be enhanced by using the Shrink-and-Shake (S&S) algorithm [2]. Fig. 7 shows an example of two consecutive strokes where each consists of a shrink followed by a shake. Suppose that Fig. 7(a) shows an initial packing $D_{1st}^+ = \{d_0\} \cup D$ where $D = \{d_1, d_2, \ldots, d_{10}\}$ and d_0 shown as the biggest blue circle containing the disks in D. Given D_{1st}^+, the `SHRINK` operation shrinks the container (shown as the red circle) and recognizes the disks protruding the shrunken container boundary (shown as the black filled disk in Fig. 7(a)). Given a shrunken container, the `SHAKE` operation follows: For each disk d_π protruding the shrunken container, S&S tries to push d_π into the shrunken container after repositioning all disks in the shrunken container. Depending on condition, the push of one protruding disk may resolve the protrusion problem or multiple pushes of more than one protruding disks may be necessary. If the `SHAKE` operation fails due to a failure to push a protruding disk, S&S `ENLARGE`s the container size followed by another attempt to shake the disks with the enlarged container. Hence, the S&S algorithm consists of the three operations: Shrink, shake, and enlarge.

The actions related with the push of a protruding disk by repositioning the disks is called a **pivoting**. All actions related with disks between two consecutive states of container size is called a **shake** where the container size state is either "shrink" or "enlarge." A pair of shrink and shake or sometimes a pair of enlarge and shake is called a **stroke**. A shake consists of one or more pivots. Fig. 7(b) shows an improved packing D_{2nd}^+ the disk arrangement after one stroke of shrinking and shaking is completed. Fig. 7(c) shows another shrunken container and Fig. 7(d) shows the next packing improvement D_{3rd}^+ after the second stroke is completed.

Lemma 15 (Quoted from[2]). *For each shrinkage, the 2004 S&S Algorithm takes $O(Mn^4)$ time in the worst case for n disks where $M \ll n$ represents the number of protruding disks.*

Table 1
ENTIRESET: Benchmark data set downloaded from Packomania. We used 60 (from 65) files of Group-III whose best-known results are available from Packomania. Col. A: Group identifier; Col. B: Radius definition rule; Col. C: #Data files; Col. D: Min/max #disks; Col. E: Min/max radii of disks;

Data group (A)	Radius rule (B)	#files (C)	#disks [min, max] (D)	Radius [min, max] (E)
`Group-I (CCIN)`	$r_i = i$	196	[5,200]	[1, 200]
`Group-II (CCIR)`	$r_i = i^{\frac{1}{2}}$	96	[5,100]	$[1, \sqrt{100}] = [1,10]$
`Group-III (CCIB)`	$r_i = i^{-\frac{1}{5}}$	60(65)	[5,69]	$[69^{-\frac{1}{5}}, 1] \simeq [0.429,1]$
`Group-IV (CCIS)`	$r_i = i^{-\frac{1}{2}}$	96	[5,100]	$[\frac{1}{\sqrt{100}}, 1] = [0.1,1]$
`Group-V (CCIC)`	$r_i = i^{-\frac{2}{3}}$	56	[5,60]	$[60^{-\frac{2}{3}}, 1] \simeq [0.065,1]$

Lemma 15 holds because it is sufficient to check, for each of the $M \le n$ protruding disks, if a disk $d \in D$ in the sorted list can move to a new location so that it is intersection-free from all triplets of d_i, d_j, d_k in D.

In [2], Sugihara et al. proposed an idea to accelerate S&S using the Voronoi diagram $\mathcal{VD}$ of $D \cup \{d_0\}$. As $\mathcal{VD}$ has complete information about the vacancy among the disks within the container, the size of the combinatorial search space significantly reduces from that of the 2004 S&S algorithm. In this paper, we used the same idea employed for the realtime packing algorithm, each iteration of the S&S enhancement taking an $O(Mn^2)$ time.

7. Experiments and discussion

We have implemented the $O(n^2)$ time version of the proposed realtime packing algorithm VOROPACK-D together with the accelerated S&S algorithm in C++ and tested using the Packomania data sets and other data sets of large instances up to 10,000 disks. Computing platform is as follows. CPU: Intel(R) Xeon(R) W-2133 3.60 GHz (Single core used); RAM: 32GB; OS: Ubuntu 16.04.4.

Section 7.1 explains the data set used for the experiment; Section 7.2 reports the realtime packing result using the data sets in the Packomania data; Section 7.3 reports the enhancement of the realtime packing using the S&S algorithm; Section 7.4 reports the benchmark of the VOROPACK-D result with two well-known algorithms; Section 7.5 reports the experiment result using large problem instances of up to 10,000 disks.

7.1. Data set

We used ENTIRESET = {Group-I, Group-II, ..., Group-V} downloaded from Packomania (http://www.packomania.com ; Downloaded on March 30, 2017). The data set consists of five groups of disk sets according to the way disk radii are defined. Each disk set consists of ordered disks and is stored in a file. The precision of input real number is up to 30 digits below the decimal point. See Table 1.

Group-I (`CCIN`) consists of 196 data files: Group-I = $\{G_1^I, G_2^I, \ldots, G_{196}^I\}$. G_i^I has $i+4$ disks. For example, G_1^I and G_{196}^I have 5 and 200 disks, respectively. The disk radii rule is $r_i = i$ implying that disks have integer radii corresponding to their order in each file. For example, $G_5^I = \{d_1, d_2, d_3, d_4, d_5\}$ has the corresponding radii set {1, 2, 3, 4, 5}. Group-I represents the disk sets with a large variation in the disk size.

Group-IV (`CCIS`) consists of 96 data files: Group-IV = $\{G_1^{IV}, G_2^{IV}, \ldots, G_{96}^{IV}\}$. G_i^{IV} has $i+4$ disks and the disk radii rule is $r_i = i^{-\frac{1}{2}}$. For example, $G_1^{IV} = \{d_1, d_2, d_3, d_4, d_5\}$ has the corresponding radii set $\{1^{-\frac{1}{2}}, 2^{-\frac{1}{2}}, 3^{-\frac{1}{2}}, 4^{-\frac{1}{2}}, 5^{-\frac{1}{2}}\}$. Group-IV represents the disk sets with a small variation in the disk size. Group-III, Group-IV, and Group-V are similarly defined according to the radii generation rules.

Let BIGSET = $\{G_{196}^I, G_{96}^{II}, G_{60}^{III}, G_{96}^{IV}, G_{56}^V\} \subset$ ENTIRESET be a set consisting of five files where each is the biggest data set in the group it belongs to: Table 2 shows the statistics of the five files in BIGSET.

Let EXTREMESET-I = $\{X_{x*100} \mid x = 1, 2, \ldots, 10\}$ and EXTREMESET-II = $\{X_{x*1000} \mid x = 1, 2, \ldots, 10\}$ for the test of large instances in that X_k has k disks where the radius of ith disk in the set is defined by $r_i = i^{-\frac{1}{2}}$.

7.2. Realtime packing solutions of the Packmania data set

We tested the quality of the realtime packing results generated by the three methods, namely the big-disk-first, random-sequence, and small-disk-first methods. Fig. 8 shows the disk packing solutions of all 196 data files of Group-I using the three packing methods. The horizontal axis denotes the number of disks in data files. Fig. 8(a) shows the packing quality measured in terms of the percent deviation (defined by Eq. 10) from the best-known packings. The blue curve corresponds to the big-disk-first: The average and standard deviation of the 196 percent deviations are 4.88% and 1.41%, respectively. The green and red correspond to the random-sequence and small-disk-first methods, respectively. We observe the followings: (i) The big-disk-first method produces significantly better packings (*i.e.* lower percent deviation) than the other two methods do; (ii) The solution quality of the big-disk-first method fluctuates less than the other two methods do; (ii) The larger the problem size is, the better the solution quality tends to be. Fig. 8(b) shows computation time. Two observations: (i) No significant difference is found among the three methods; (ii) Computation time seems linear to the problem

Table 2
Statistics of BIGSET = $\{G^{I}_{196}, G^{II}_{96}, G^{III}_{60}, G^{IV}_{96}, G^{V}_{56}\}$. (B): The average and standard deviation of the radii of the disks in the set, resp. (C): The container radius of the best-known packing. (D1): The container radius, percent deviation, and computation time by big-disk-first initial packing, resp. (D2): The container radius, percent deviation, and computation time by S&S enhancement of big-disk-first initial packing, resp. (E1): The container radius, percent deviation, and computation time by random-sequence initial packing, resp. (E2): The container radius, percent deviation, and computation time by S&S enhancement of random-sequence initial packing, resp. (F1): The container radius, percent deviation, and computation time by small-disk-first initial packing, resp. (F2): The container radius, percent deviation, and computation time by S&S enhancement of small-disk-first initial packing, resp.

id (A) (#disks)	radius (B) $\hat{\mu}$ $\hat{\sigma}$	best (C) known	big-disk-first (D)		random-sequence (E)		small-disk-first (F)	
			init(D1) radius %Δ time(s)	S&S(D2) radius %Δ time(s)	init(E1) radius %Δ time(s)	S&S(E2) radius %Δ time(s)	init(F1) radius %Δ time(sec)	S&S(F2) radius %Δ time(s)
G^{I}_{196} (200)	100.500	1726.220	1805.150	1802.150	1951.680	1808.180	1972.190	1808.690
	57.879		4.572	4.399	13.061	4.748	14.249	4.777
			0.058	5.522	0.048	60.292	0.060	57.473
G^{II}_{96} (100)	6.715	75.547	79.357	79.107	86.533	78.783	85.489	78.481
	2.338		5.044	4.713	14.542	4.284	13.160	3.884
			0.034	2.410	0.032	6.638	0.027	10.405
G^{III}_{60} (64)	0.528	4.755	5.114	4.997	5.336	4.985	5.681	4.971
	0.111		7.553	5.087	12.226	4.852	19.470	4.557
			0.025	3.473	0.025	3.159	0.025	4.420
G^{IV}_{96} (100)	0.186	2.426	2.506	2.480	3.149	2.490	3.217	2.479
	0.132		3.284	2.223	29.803	2.641	32.599	2.194
			0.035	9.871	0.030	2.498	0.035	6.419
G^{V}_{56} (60)	0.155	1.773	1.975	1.795	2.170	1.804	2.254	1.797
	0.153		11.383	1.263	22.409	1.778	27.136	1.364
			0.020	8.213	0.018	5.230	0.026	9.819

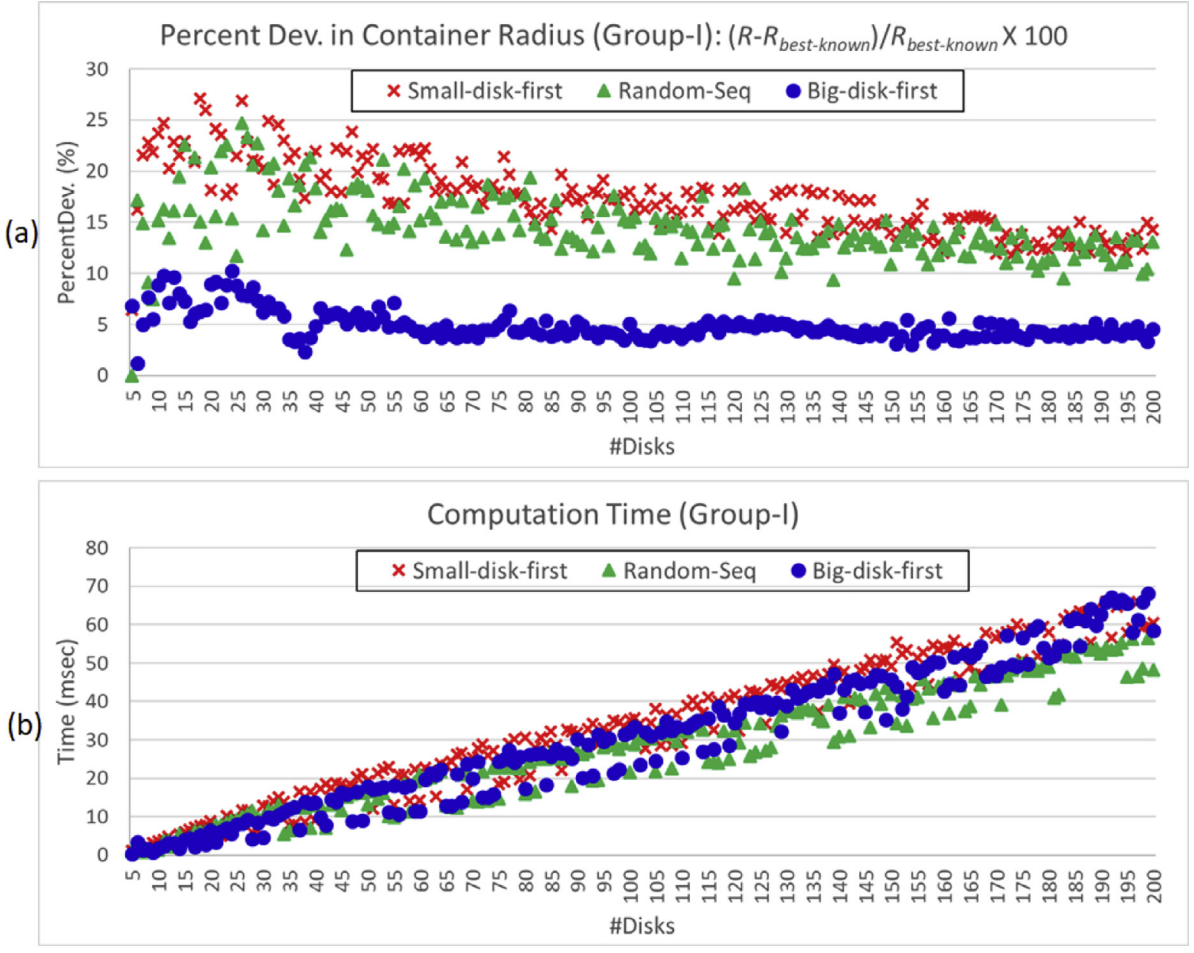

Fig. 8. Comparison of the results of the three realtime packing methods: Big-disk-first, random-sequence, and small-disk-first. Data set: Group-I. $\hat{\mu}$: average of percent deviation. $\hat{\sigma}$: standard deviation of percent deviation. (a) Solution quality: Big-disk-first ($\hat{\mu}/\hat{\sigma}$: 4.88/1.41); Random-sequence ($\hat{\mu}/\hat{\sigma}$: 14.54/3.19); Small-disk-first ($\hat{\mu}/\hat{\sigma}$: 17.31/3.55). (b) Computation time.

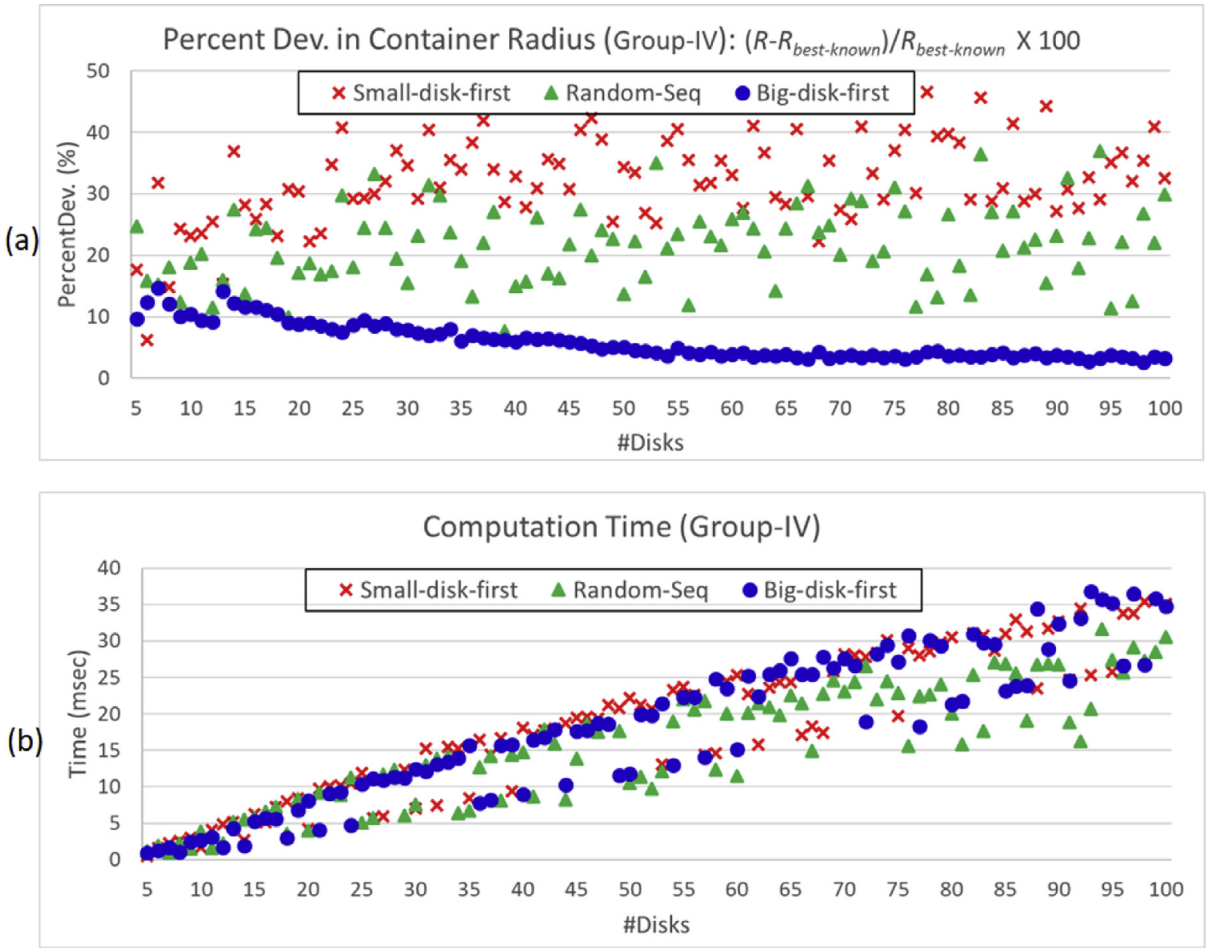

Fig. 9. Comparison of the results of the three realtime packing methods: Big-disk-first, random-sequence, and small-disk-first. Data set: Group-IV. $\hat{\mu}$: average of percent deviation. $\hat{\sigma}$: standard deviation of percent deviation. (a) Solution quality: Big-disk-first ($\hat{\mu}/\hat{\sigma}$: 5.97/2.88); Random-sequence ($\hat{\mu}/\hat{\sigma}$: 21.59/6.20); Small-disk-first ($\hat{\mu}/\hat{\sigma}$: 32.04/6.82). (b) Computation time.

size. The biggest computation time for data up to the size 200 is about 70 msec. Fig. 9 shows a similar analysis using Group-IV.

We also tested the realtime packing algorithms using the other disk sets in ENTIRESET. Fig. 10 shows the big-disk-first realtime packing results of Group-II, III, and V. Similar patterns are observed. Table 2 summarizes the experimental result of the Packomania data using the three packing algorithms.

7.3. Enhancement of the realtime packing results using the S&S algorithm

The packing results from the realtime big-disk-first algorithm might be sufficiently good for some applications. In some cases, however, an enhancement might be necessary or desirable. In such cases, we can apply the S&S enhancement to realtime packings. We used $\epsilon_{term} = r_{min} * 10^{-3}$ for the termination tolerance of shrinkage (See Appendix A). We used the S&S enhancement of realtime packing solutions of ENTIRESET and compared the solution quality with the best-known results posted in Packomania.

Fig. 11 shows the enhanced packing results of Group-I data using the S&S algorithm with the initial packings produced by the three realtime packing methods. The horizontal axis denotes the data size; The left and the right vertical axes denote percent deviation and computation time, respectively. Fig. 11(a) shows the big-disk-first case: The solid and broken red curves denote the S&S enhancement result and the initial packing from the big-disk-first method, respectively. The packing qualities of both the initial packing and the enhanced packing are measured by the percent deviation from the best-known packings. The blue solid curve denotes the computation time taken by the S&S enhancement. Fig. 11(b) and (c) show the random sequence and small-disk-first cases, respectively. Fig. 12 shows a similar analysis using Group-IV data set. We also performed the same experiment using the other diks sets in ENTIRESET. Fig. 13 shows the S&S enhancement of the big-disk-first realtime packing for Groups II, III, and V.

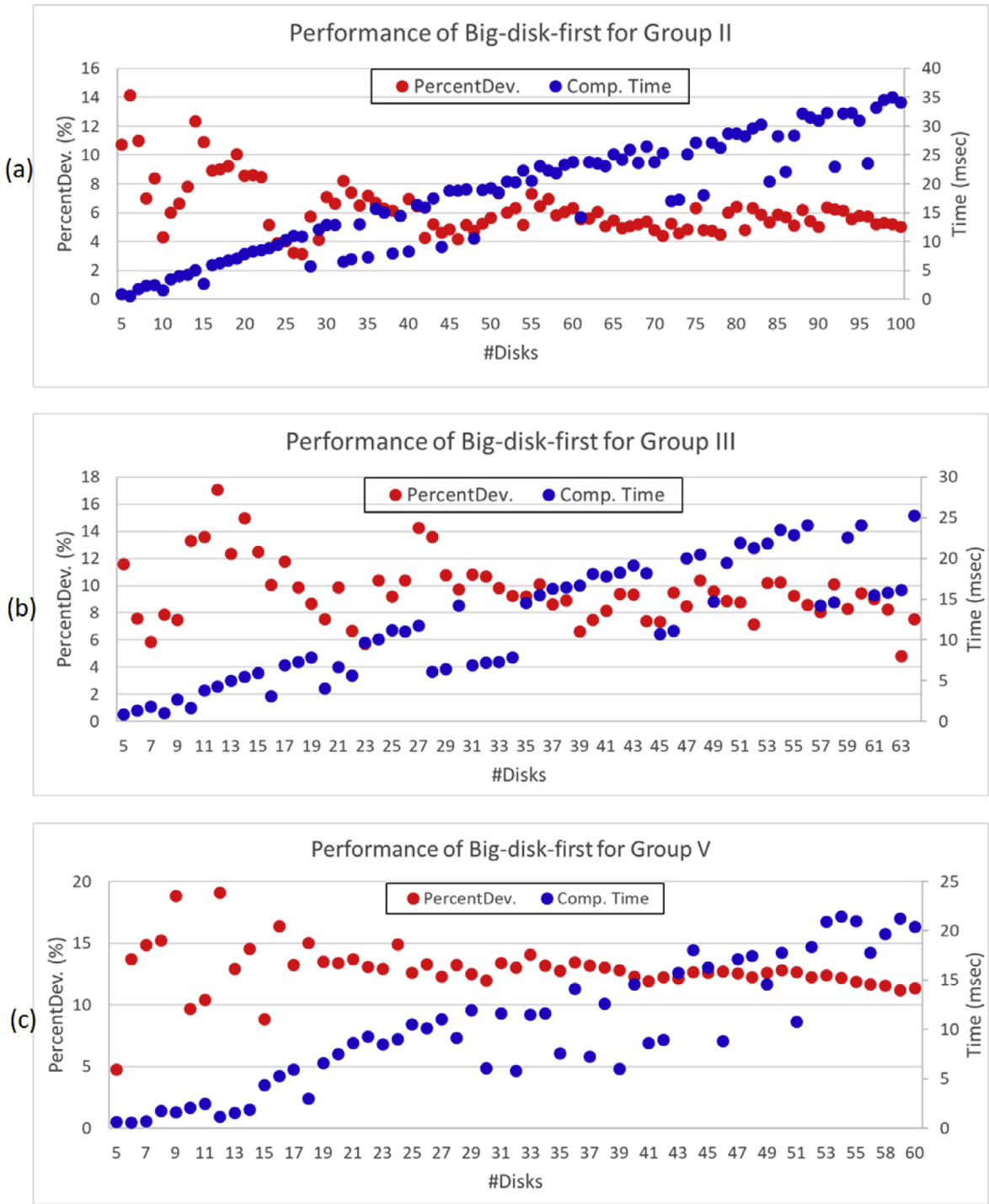

Fig. 10. The results of the big-disk-first realtime packing method for the other three data sets in Packomania. Red: Solution quality (Percentage deviation; Left vertical axis). Blue: Computation time (Right vertical axis). $\hat{\mu}$: average of percent deviation. $\hat{\sigma}$: standard deviation of percent deviation. (a) Group-II ($\hat{\mu}/\hat{\sigma}$: 6.25/1.86). (b) Group-III ($\hat{\mu}/\hat{\sigma}$: 9.55/2.27). (c) Group-V ($\hat{\mu}/\hat{\sigma}$: 12.88/2.02). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7.4. Benchmark test

We compared the performance of the realtime packing followed by the S&S enhancement with two well-known algorithms. Al-Mudahka, Hifi, and M'Hallah developed an algorithm in 2011, abbreviated here as AMHiMH11, which consists of a taboo search (to explore the combinatorial nature of DPP) and nested partitioning and nonlinear optimization (to explore

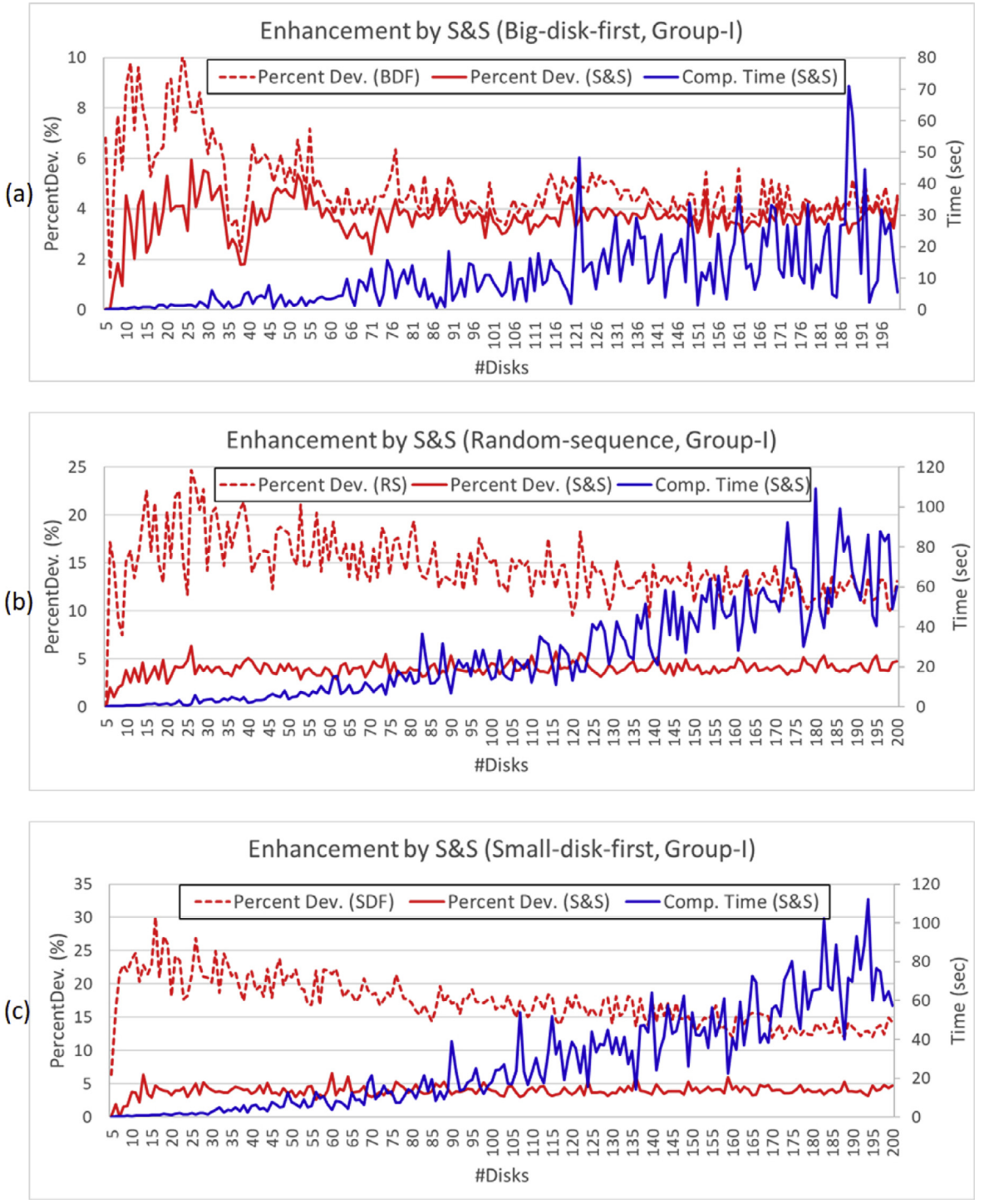

Fig. 11. Comparison of the results of the S&S algorithm using the initial packings from the three realtime packing methods (Group-I): Big-disk-first, random-sequence, and small-disk-first. Data set: Group-I. Red curve: Percent deviation (solid: S&S enhanced; Broken: Initial packing). Blue curve: Computation time. $\hat{\mu}$: average of percent deviation. $\hat{\sigma}$: standard deviation of percent deviation. (a) Big-disk-first method ($\hat{\mu}/\hat{\sigma}$: 3.65/0.78). (b) Random-sequence method ($\hat{\mu}/\hat{\sigma}$: 3.95/0.74). (c) Small-disk-first method ($\hat{\mu}/\hat{\sigma}$: 3.96/0.82). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

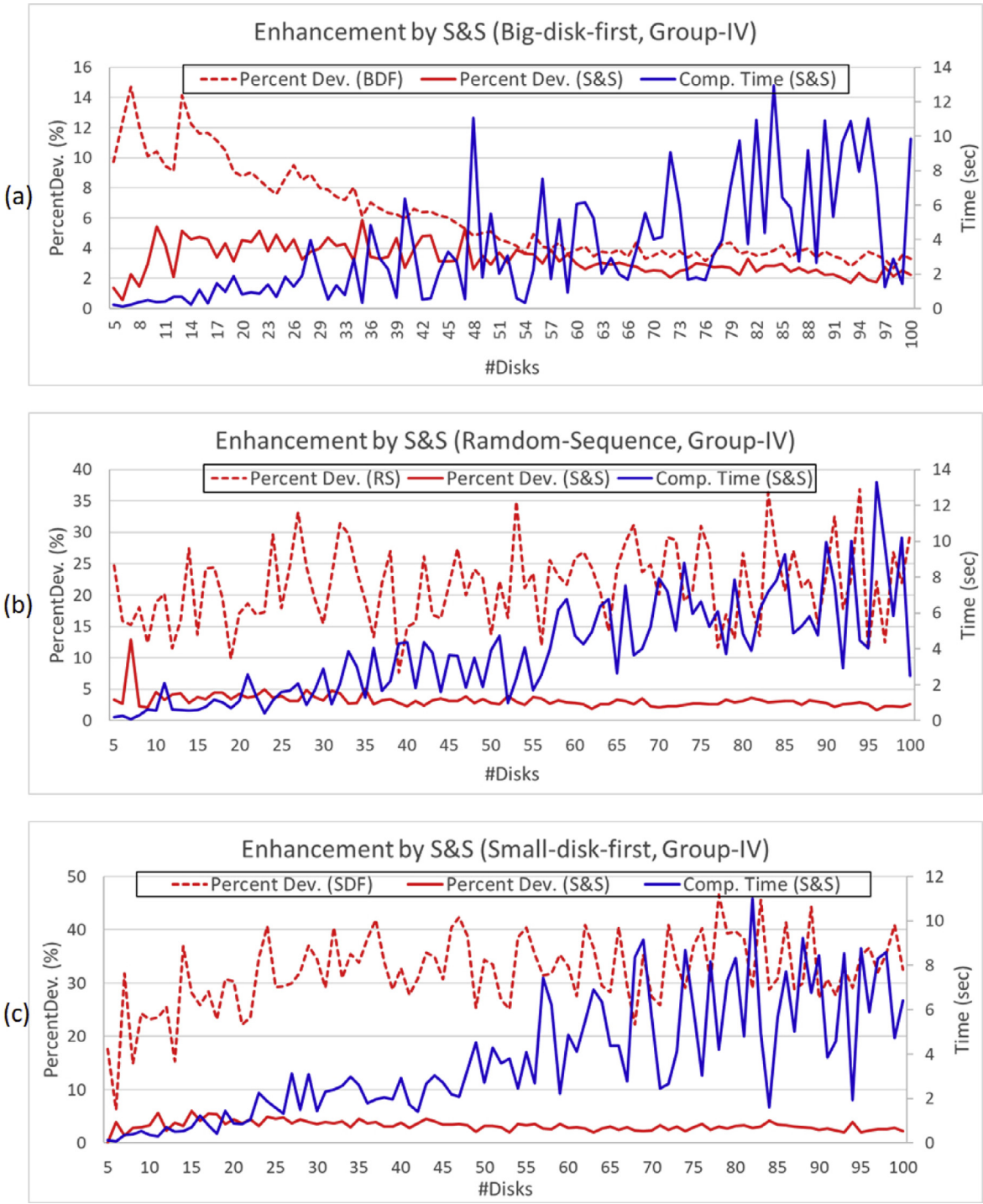

Fig. 12. Comparison of the results of the S&S algorithm using the initial packings from the three realtime packing methods: Big-disk-first, random-sequence, and small-disk-first. Data set: Group-IV. Red curve: Percent deviation (solid: S&S enhanced; Broken: Initial packing). Blue curve: Computation time. $\hat{\mu}$: average of percent deviation. $\hat{\sigma}$: standard deviation of percent deviation. (a) Big-disk-first method ($\hat{\mu}/\hat{\sigma}$: 3.22/1.00). (b) Random-sequence method ($\hat{\mu}/\hat{\sigma}$: 3.24/1.22). (c) Small-disk-first method ($\hat{\mu}/\hat{\sigma}$: 3.26/0.91). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

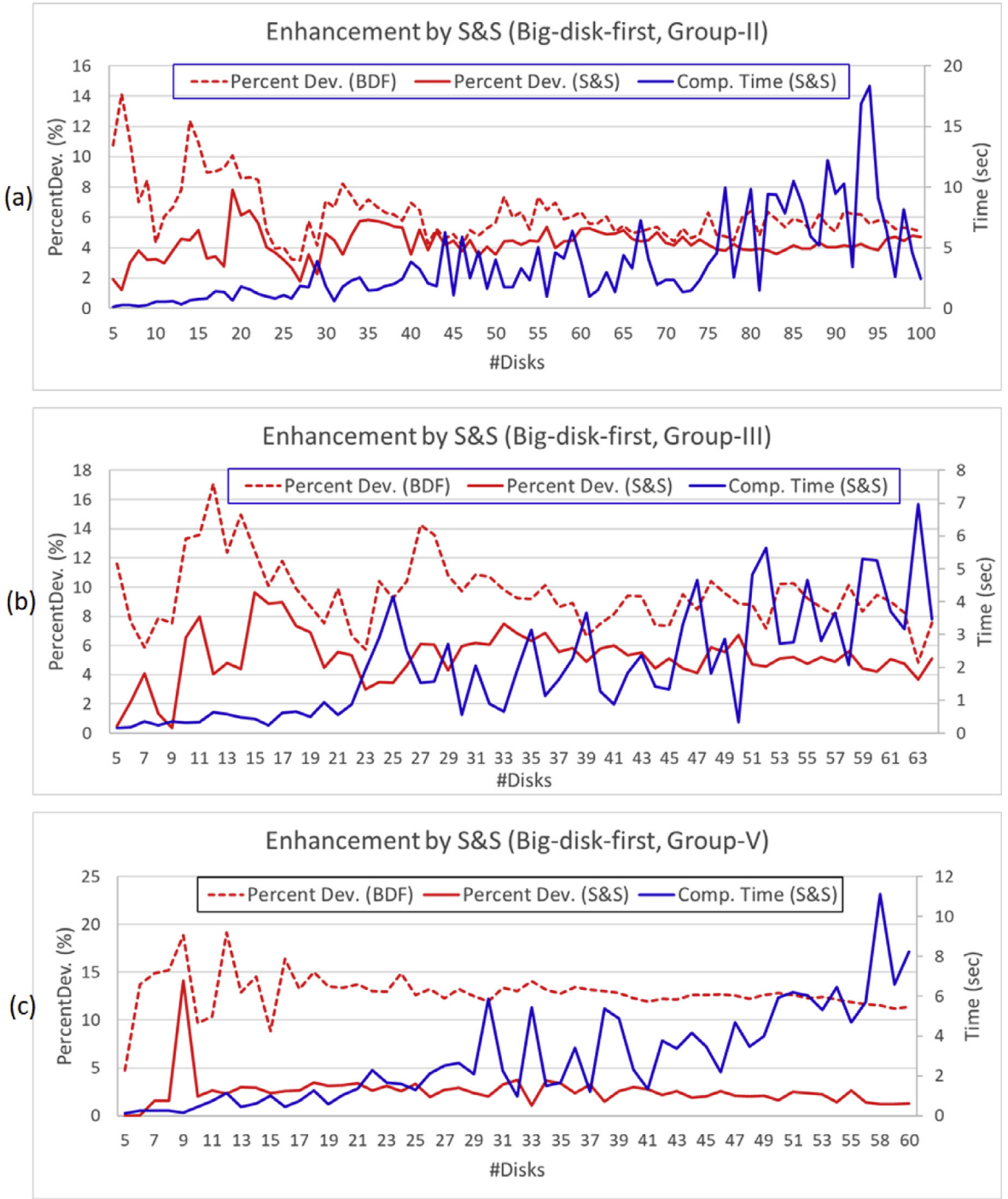

Fig. 13. S&S enhancements of the big-disk-first realtime packing for the data sets in Group-II, Group-III, Group-V. Red curve: Percent deviation (solid: S&S enhanced; Broken: Initial packing). Blue curve: Computation time. $\hat{\mu}$: average of percent deviation. $\hat{\sigma}$: standard deviation of percent deviation. (a) Group-II. Big-disk-first: ($\hat{\mu}/\hat{\sigma}$: 6.25/1.86); S&S enhanced: ($\hat{\mu}/\hat{\sigma}$: 4.27/0.96). (b) Group-III Big-disk-first: ($\hat{\mu}/\hat{\sigma}$: 9.55/2.27); S&S enhanced: ($\hat{\mu}/\hat{\sigma}$: 5.21/1.72). (c) Group-V Big-disk-first: ($\hat{\mu}/\hat{\sigma}$: 12.88/2.02); S&S enhanced: ($\hat{\mu}/\hat{\sigma}$: 2.56/1.77). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

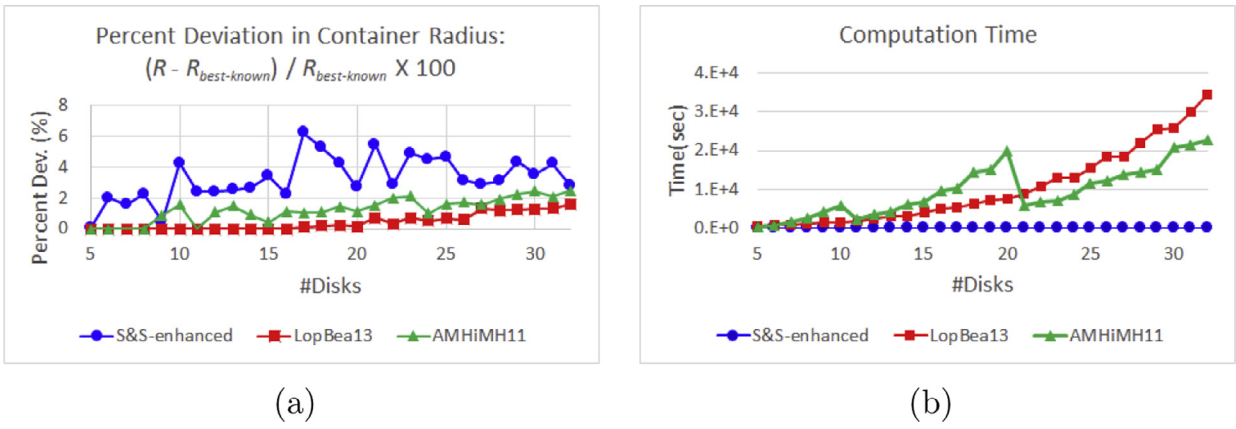

Fig. 14. Comparison of the S&S-enhanced realtime packing with LopBea13 [58] and AMHiMH11 [100]. Data set: BENCHMARKSET-A = $\{G_1^I, G_2^I, \ldots, G_{28}^I\}$. $|G_{28}^I|$=32. Radius generation rule: $r_i = i$. Computing environments used are slightly different. $\epsilon_{term} = r_{min} * 10^{-3}$). Blue circle: S&S-enhanced realtime packing. Red square: LopBea13 [58]. Green triangle: AMHiMH11 [100]. Both reported computation results up to G_{28}^I. Percentage deviation: average 3.93% and standard deviation 0.84%. (a) Packing quality (Percentage deviation from the best-known packings). (b) Computation time. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the continuous optimization nature of DPP) [100]. They reported computation results of BENCHMARKSET ⊂ ENTIRESET using the following platform: Pentium IV, 2.66 GHz CPU with 512MB RAM. Program coded in FORTRAN that invokes GAMS which in turn calls CONOPT3 NLP solver. Lopez and Beasley developed an algorithm in 2013, abbreviated here as LopBea13, which consists of an optimization phase (based on formulation space search) and an improvement phase (using a perturbation of a local solution by swapping two disks) [58]. They reported the packing results of BENCHMARKSET using the following computing platform: Intel(R) Core(TM) i5-2500 3.30 GHz CPU with 4.0GB RAM; Program coded in MatLab 7.9.0 using SNOPT nonlinear solver.

Fig. 14(a) and (b) shows the solution quality and the computation time of the three algorithms with BENCHMARKSET-A = $\{G_1^I, G_2^I, \ldots, G_{28}^I\}$ ⊂ Group-I consisting of 28 data files, respectively. AMHiMH11 and LopBea13 reported computation results only up to G_{28}^{IV}. The blue curve corresponds to the S&S enhancement of the big-disk-first method; the green AMHiMH11; the red LopBea13. With the tested small models in BENCHMARKSET-A (G_{28}^I with 32 disks is the biggest data tested by both algorithms), LopBea13 is best from solution quality point of view and AMHiMH11 is better than S&S. However, observe in Fig. 14(a) that the red and green curves have clearly increasing patterns w.r.t. model size (whereas the blue curve for the S&S enhancement is beginning to decrease as is clearly shown in the figure). All three algorithms have similar percentage deviation at the biggest model G_{28}^I. More importantly, compare the computation time (Be aware that the computing environments are slightly different). The S&S enhancement is several orders of magnitude faster than the others! For example, LopBea13 took 34,444.8 s for G_{28}^I but the proposed method took only 4.9 s: The proposed method was more than 7000 times faster than LopBea13. We note an interesting pattern of AMHiMH11: The computation time curve has sharp decreases around 11 and 21 disks but the corresponding packing quality does not deteriorate at all.

Fig. 15 shows a similar benchmark using a subset of Group-IV data. Fig. 15(a) and (b) shows the comparison of the percentage deviation and computation time among the three algorithms using BENCHMARKSET-B = $\{G_1^{IV}, G_2^{IV}, \ldots, G_6^{IV}, G_8^{IV}, G_{10}^{IV}, G_{12}^{IV}, G_{14}^{IV}, G_{16}^{IV}, G_{21}^{IV}, G_{26}^{IV}, G_{31}^{IV}\}$ ⊂ Group-IV consisting of 14 data files. AMHiMH11 and LopBea13 reported computation results only up to G_{31}^{IV}. Fig. 15(a) shows that LopBea13 produces best packings for small models but deteriorates quickly as model size increases. AMHiMH11 and the proposed method have very similar behavior of packing quality for small models. All three algorithms produced a similar packing quality for G_{31}^{IV} (with 35 disks). Fig. 15(b) shows that the computation time taken by S&S is tiny compared to the other two algorithms. For G_{31}^{IV}, S&S took 2.4,s whereas LopBea13 took 125,523 s: S&S was more than 50,000 times faster than LopBea13.

7.5. Expriments with large problem instances

In order to make a proper assessment of the big-disk-first realtime packing algorithm for large problem instances, we performed experiments using EXTREMESET-I and EXTREMESET-II where each has ten moderate-sized and large-sized data files, up to 10,000 disks, respectively. The radius of ith disk is defined by $r_i = i^{-\frac{1}{2}}$. Fig. 16 shows the experiment result of the big-disk-first realtime packing applied to EXTREMESET-I: The red and black curves denote the packing density (the left vertical axis) and computation time (the right vertical axis), respectively. Fig. 18 shows the result of the big-disk-first packing of the data set EXTREMESET-I. Figs. 17 and 19 show the same experiment with EXTREMESET-II. Fig. 20 shows the close-ups

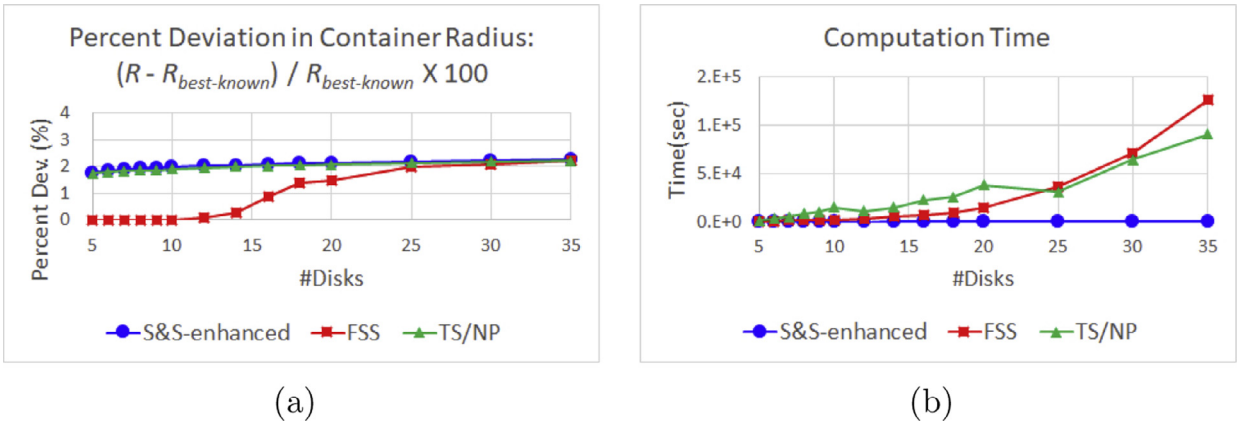

Fig. 15. Comparison of the S&S-enhanced realtime packing with LopBea13 [58] and AMHiMH11 [100]. Data set: BENCHMARKSET-B = $\{G_1^{IV}, G_2^{IV}, \ldots, G_6^{IV}, G_8^{IV}, G_{10}^{IV}, G_{12}^{IV}, G_{14}^{IV}, G_{16}^{IV}, G_{21}^{IV}, G_{26}^{IV}, G_{31}^{IV}\}$. $|G_{31}^{IV}|$=35. Radius generation rule: $r_i = i^{-\frac{1}{2}}$. Computing environments used are slightly different. $\epsilon_{term} = r_{min} * 10^{-3}$). Blue circle: S&S-enhanced realtime packing. Red square: LopBea13 [58]. Green triangle: AMHiMH11 [100]. Both reported computation results up to G_{28}^{I}. Percentage deviation: average 3.37% and standard deviation 0.85%. (a) Packing quality (Percentage deviation from the best-known packings). (b) Computation time. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

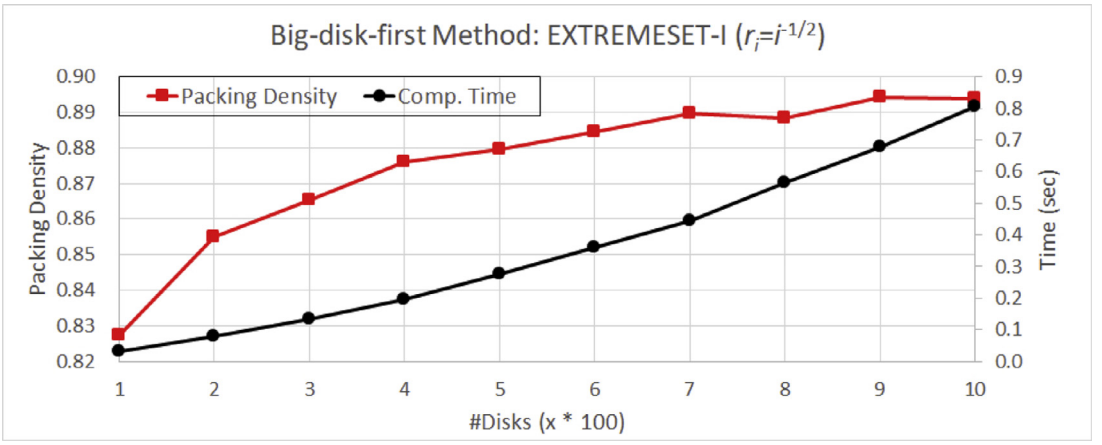

Fig. 16. Big-disk-first realtime packing method applied to EXTREMESET-I = $\{X_{x*100} \mid x = 1, 2, \ldots, 10\}$. X_{x^*100} has x^*100 disks. The radius of ith disk is $r_i = i^{-\frac{1}{2}}$. Red curve: Packing density (Left vertical axis). Black curve: Computation time (Right vertical axis). Horizontal axis: Problem size (# disks). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

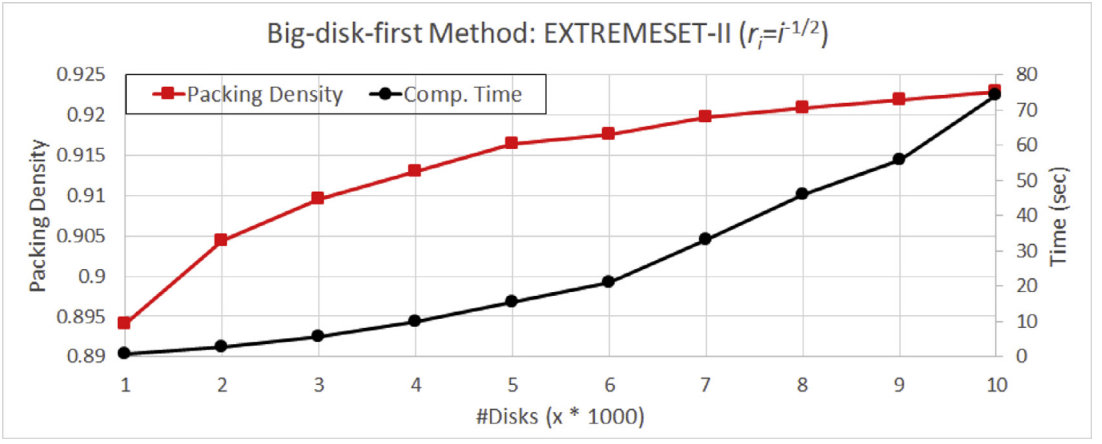

Fig. 17. Big-disk-first realtime packing method applied to EXTREMESET = $\{X_{x*1000} \mid x = 1, 2, \ldots, 10\}$. X_{x^*1000} has x^*1000 disks where the radius of ith disk is defined by $r_i = i^{-\frac{1}{2}}$. Red curve: Packing density (Left vertical axis). Black curve: Computation time (Right vertical axis). Horizontal axis: Problem size (# disks). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

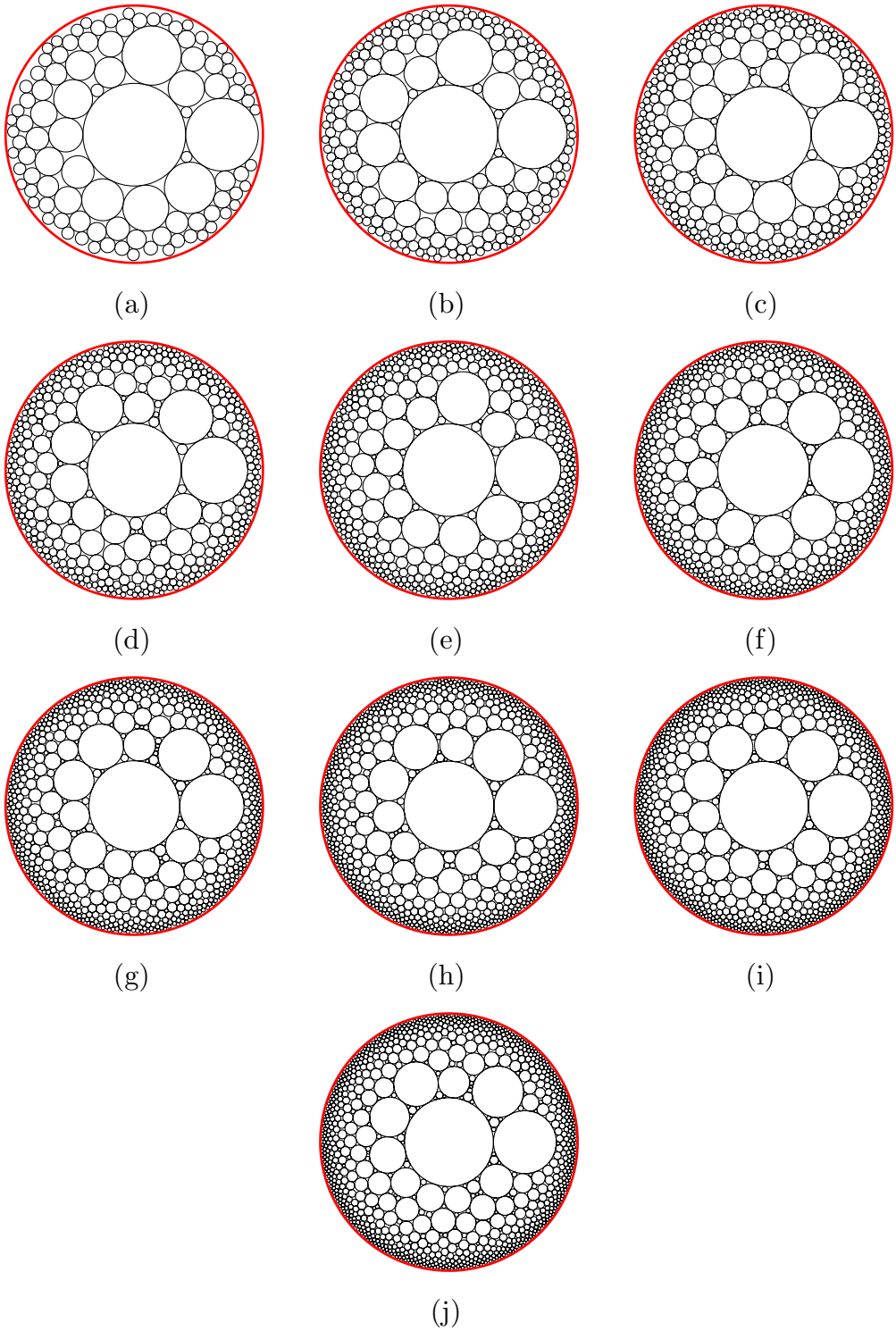

Fig. 18. Big-disk-first packing results of EXTREMESET-I. All containers are normalized. ρ: density. t: computation time. (a) 100 disks (ρ: 0.8272; t: 0.03 s), (b) 200 disks (ρ: 0.8548; t: 0.08 s), (c) 300 disks (ρ: 0.8652; t: 0.13 s), (d) 400 disks (ρ: 0.8760; t: 0.20 s), (e) 500 disks (ρ: 0.8795; t: 0.28 s), (f) 600 disks (ρ: 0.8845; t: 0.36 s), (g) 700 disks (ρ: 0.8897; t: 0.44 s), (h) 800 disks (ρ: 0.8883; t: 0.57 s), (i) 900 disks (ρ: 0.8941; t: 0.68 s), (j) 1000 disks (ρ:0.8940; t: 0.81 s).

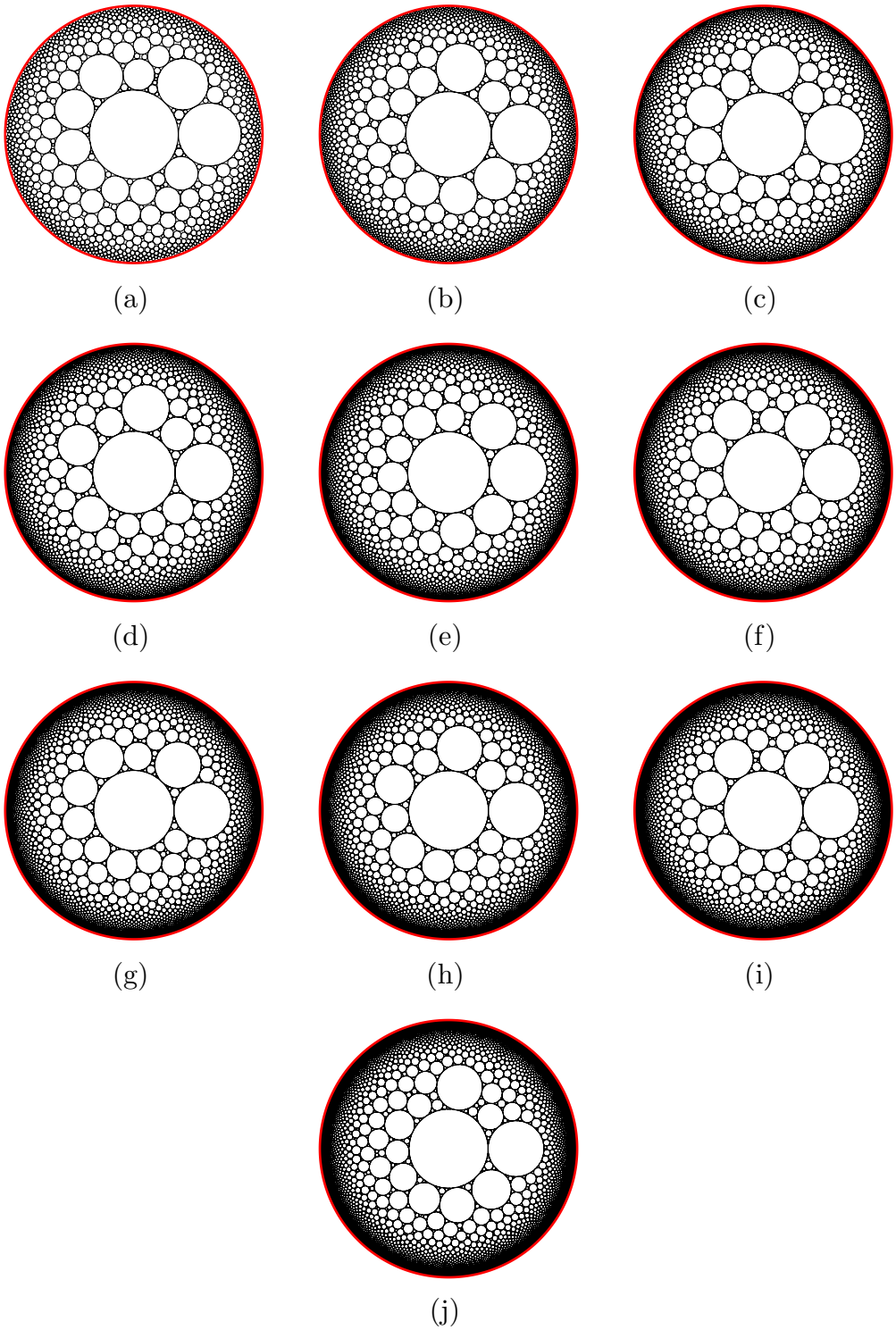

Fig. 19. Big-disk-first packing results of EXTREMESET-II. All containers are normalized. ρ: density. t: computation time. (a) 1000 disks (ρ: 0.8940; t: 0.81 s), (b) 2,000 disks (ρ: 0.9044; t: 2.63 s), (c) 3000 disks (ρ: 0.9096; t: 5.79 s), (d) 4000 disks (ρ: 0.9130; t: 9.98 s), (e) 5000 disks (ρ: 0.9164; t: 15.52 s), (f) 6000 disks (ρ: 0.9175; t: 21.12 s), (g) 7000 disks (ρ: 0.9196; t: 33.14 s), (h) 8000 disks (ρ: 0.9208; t: 46.00 s), (i) 9000 disks (ρ: 0.9218; t: 55.82 s), (j) 10,000 disks (ρ: 0.9228; t: 74.08 s).

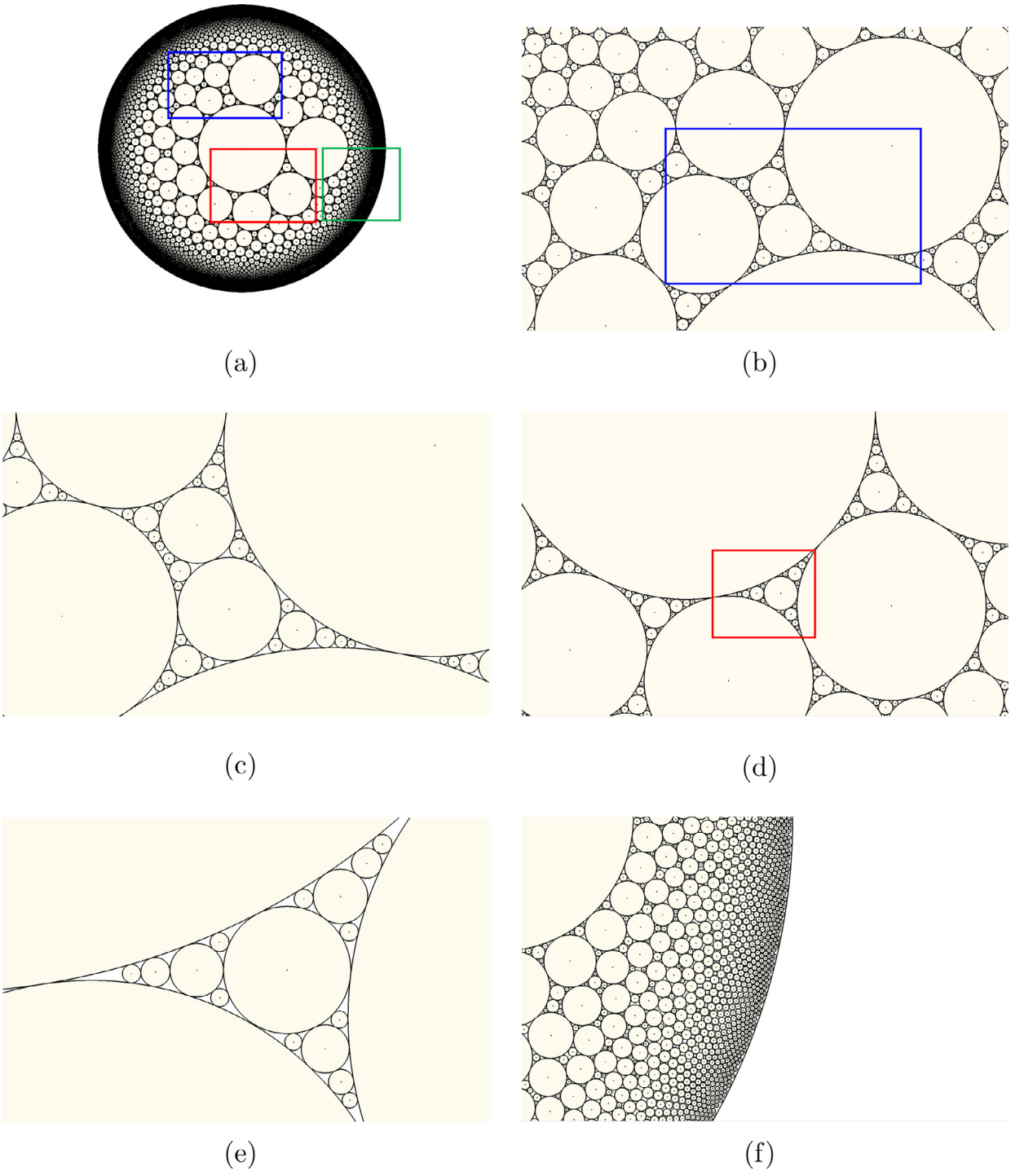

Fig. 20. Close-ups of the big-disk-first realtime packing of the 10,000 disks. (a) The big-disk-first realtime packing of the 10,000 disks in Fig. 19(j). The rectangles are the locations for zoom-up. (b–c) Consecutive zoom-ups of the blue rectangle in (a). (d–e) Consecutive zoom-ups of the red rectangle in (a). (f) Zoom-up of the green box in (a). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

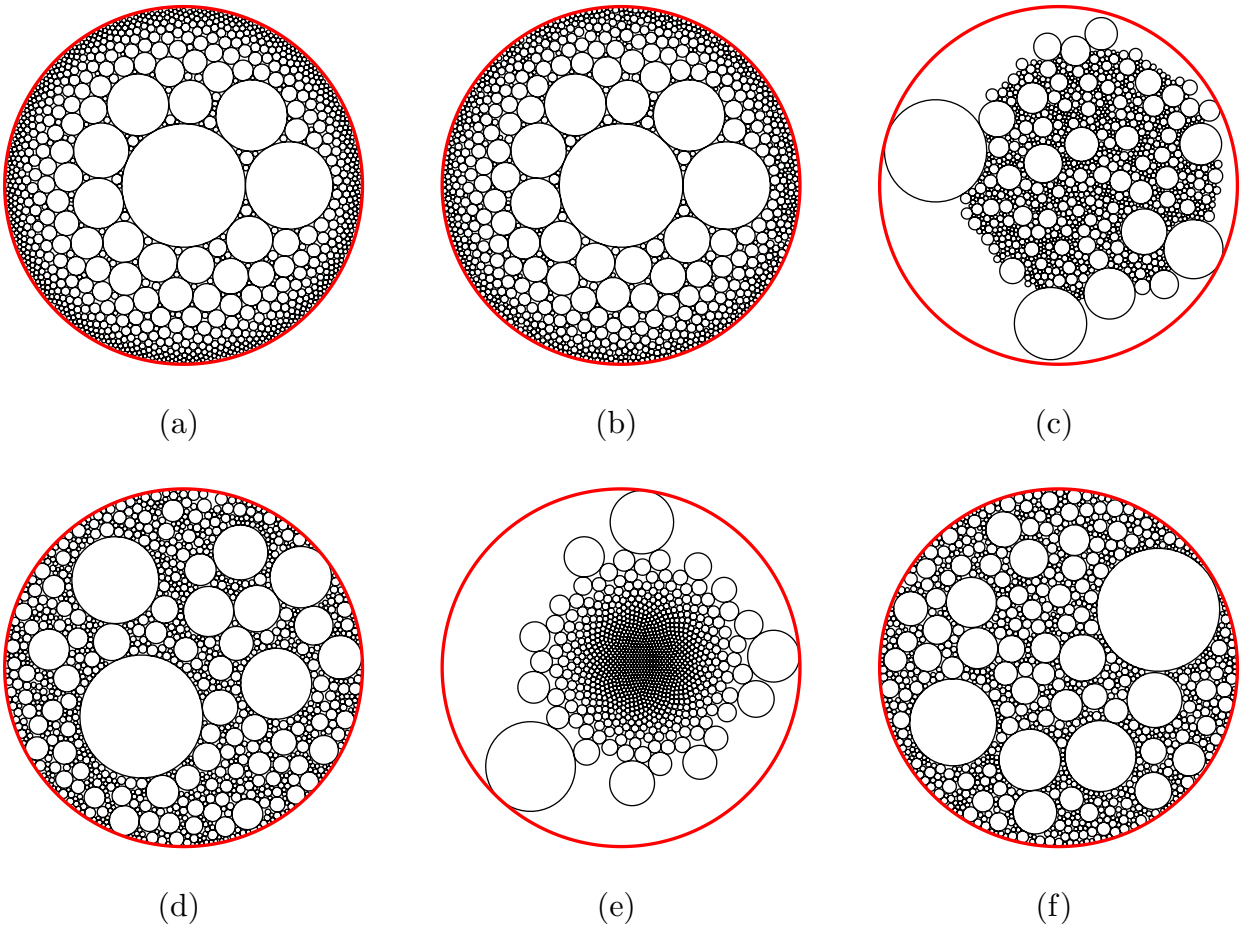

Fig. 21. The three realtime packings of X_{1000} disk set with 1,000 disks and its enhancement using S&S. All containers are normalized. Radius rule: $r_i = i^{-\frac{1}{2}}\epsilon_{term} = r_{min} * 10^{-3}$. (a -b) Big-disk-first. (a) Initial packing: Container radius: 2.894; Packing density: 0.894; Computation time: 0.674 s. (b) S&S-enhancement: Container radius: 2.893; Packing density: 0.895; Computation time: 78.669 s. (c–d) Random-sequence. (c) Initial packing: Container radius: 3.496; Packing density: 0.612; Computation time: 0.399 s. (d) S&S-enhancement: Container radius: 2.909; Packing density: 0.885; Computation time: 224.887 s. (e–f) Small-disk-first. (e) Initial packing: Container radius: 3.985; Packing density: 0.471; Computation time: 0.564 s. (f) S&S-enhancement: Container radius: 2.907; Packing density: 0.886; Computation time: 274.073 s.

of the big-disk-first realtime packing of the 10,000 disks in Fig. 19(j). Note that the interstitial areas among big disks are packed by several smaller disks as much as possible.

We performed another experiment as follows. We selected two problem instances from EXTREMESET-II: The smallest X_{1000} with 1000 disks and the largest $X_{10,000}$ with 10,000 disks. Fig. 21(a) shows the big-disk-first packing of the 1000 disks in X_{1000} in the container radius 2.894 computed in 0.674 s. Fig. 21(b) shows the S&S-enhanced packing with the container radius 2.893: The 0.001 radius enhancement is obtained by about 79 s. Fig. 21(c) and (d) corresponds to the random initial packing and its S&S-enhancement, respectively. Fig. 21(e) and (f) corresponds to the small-disk-first initial packing and its S&S-enhancement, respectively. Observe that the big-disk-first realtime packing without an S&S-enhancement significantly outperforms the S&S-enhancements of both random and small-disk-first initial packings.

Fig. 22 shows the same experiment with the 10,000 disks in $X_{10,000}$. Fig. 22(a) shows the big-disk-first realtime packing of the 10,000 disks in the container radius 3.257 computed in approximately 61 s. Fig. 22(b) shows the result of the S&S-enhancement: No enhancement was made even if the S&S was attempted more than 1.5 h. Fig. 22(c) and (d) corresponds to the same experiment using the random realtime packing followed by the S&S-enhancement (which took more than 4.5 h). Observe the big void at Southeash of the container. This void is formed because we make a placement of disk at a 2-contact point. Fig. 22(e) and (f) corresponds to the same experiment using the small-disk-first realtime packing followed by the S&S-enhancement. We again observe that the big-disk-first realtime packing without an S&S-enhancement significantly outperforms the S&S-enhancements of both random and small-disk-first initial packings. For large problem instances, the big-disks-first initial packing is already good enough for most applications.

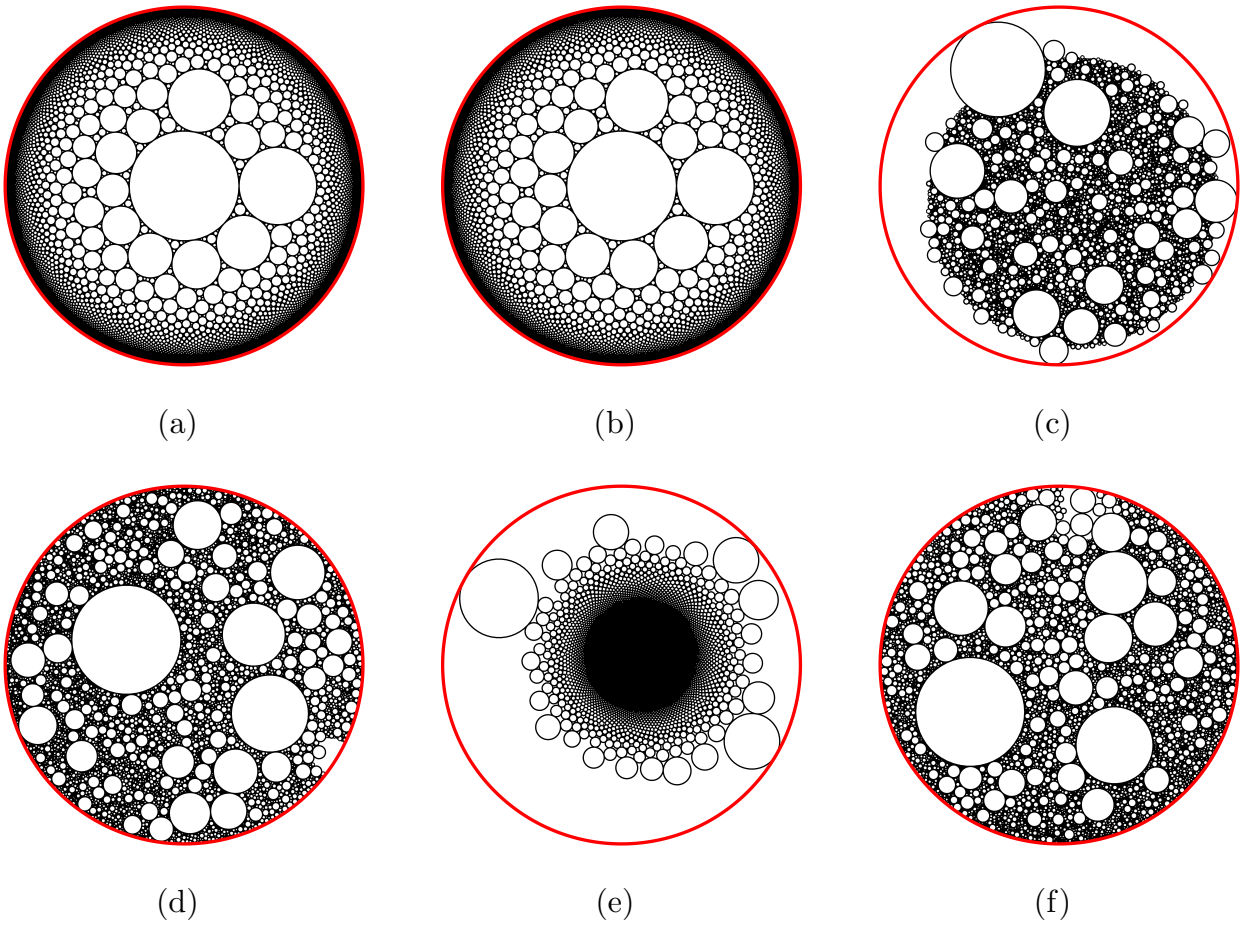

Fig. 22. The three realtime packings of X_{10000} disk set with 10,000 disks and its enhancement using S&S. Radius rule: $r_i = i^{-\frac{1}{2}}\epsilon_{term} = r_{min} * 10^{-3}$. (a–b) Big-disk-first. Observe that no enhancement is made up to the fifth digit after the decimal point. (a) Initial packing: Container radius: 3.25674; Packing density: 0.92281; Computation time: 60.903 s. (b) S&S-enhancement: Container radius: 3.25674; Packing density: 0.92281; Computation time: 5599.120 s ($\approx$ 1.56 h). (c–d) Random-sequence. (c) Initial packing: Container radius: 3.776; Packing density: 0.687; Computation time: 49.371 s. (d) S&S-enhancement: Container radius: 3.276; Packing density: 0.912; Computation time: 16327.800 s ($\approx$ 4.54 h). (e–f) Small-disk-first. (e) Initial packing: Container radius: 4.549; Packing density: 0.473; Computation time: 51.296 s. (f) S&S-enhancement: Container radius: 3.279; Packing density: 0.910; Computation time: 40122.600 s ($\approx$ 11.15 h).

8. Conclusion

In this paper, we present a realtime algorithm VOROPACK-D for packing disks in a circular container: The algorithm finds good packing solutions sufficient enough for many applications in realtime, specifically speaking, in less than one second for moderate to large problem instances. For small to moderate sized problems, the realtime solution can be further enhanced using the Shrink-and-Shake algorithm in the cost of more computation, For extremely large problem instances, the big-disk-first method produces sufficiently good packing and the S&S enhancement may not be necessary.

In both the realtime algorithm and the S&S algorithm, we use the vacancy information available from the Voronoi diagram of disks in a circular container. The most critical technical issues are two-fold (We avoid to explain their details in this paper): The construction of the Voronoi diagram and its maintenance through the decrement of a disk from and the increment of a disk to the Voronoi diagram, both robustly and efficiently. The topology-oriented incremental algorithm turned out to be sufficiently good for this purpose [88,94].

VOROPACK-D is fully implemented (using the $O(n^2)$ time algorithm) and tested with a set of benchmark data available in Packomania web site. The program is freely available from Voronoi Diagram Research Center (http://voronoi.hanyang.ac.kr).

Outlook: In this paper, we have not attempted to find the global optimum from the local optimum found by either the realtime algorithm or the S&S algorithm. As the current packing results have a deviation of approximately 2–4% from the best-known literature results, it might be necessary for some applications to escape from a local minimum. The vacancy information in the Voronoi diagram and the computational efficiency and robustness of constructing the Voronoi diagram

of disks using the topology-oriented incremental algorithm [94] would be critical for this purpose. We expect that an effort to optimize the code of VOROPACK-D will further improve its performance. We anticipate a significant amount of current computation time can be saved from code optimization including an efficient priority queue implementation.

We have introduced the Voronoi diagram to the disk packing problem in 2004 [2] and present here its improvement with a full implementation. We expect the idea of using Voronoi diagram can be adapted to other hard optimization problems such as cutting stock, nesting, ellipse packing, bin packing, minimal convex hulls, knapsack problems, etc. in relation with a variety of container shapes. Extending the approach to three-dimensional objects (spheres, ellipsoids, etc.) is another challenge. We invite suggestions and collaborations for possible applications of Voronoi diagrams.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP, MSIT) (Nos. 2017R1A3B1023591, 2016K1A4A3914691).

Appendix A. Termination tolerance

Fig. 23 shows the influence of termination tolerance ϵ_{term} on the performance of S&S using BIGSET. Let r_{min} be the radius of the smallest disk in each file. The horizontal axis denotes the six different values of $\epsilon_{term} = r_{min} * 10^{-k},\ k = 1, 2, \ldots, 6$. Fig. 23(a) shows the packing quality (*i.e.*, the percentage deviation of the container radius) by S&S to the best-known container radius according to

$$(R^* - R_{best-known})/R_{best-known} * 100 \tag{10}$$

where R^* and $R_{best-known}$ are the radii of the container found by the S&S algorithm and that of the best-known packing, respectively. Fig. 23(b) and (c) shows the computation time and the number of pivotings, respectively. We observe the following: The smaller ϵ_{term} is, the better the packing quality is, the smaller the computed container is, and therefore the smaller the percent deviation is. However, the computational requirement increases as ϵ_{term} decreases whereas the packing quality improves marginally beyond $k > 3$. Hence, we conclude from this experiment that $\epsilon_{term} = r_{min} * 10^{-3}$ is an optimal choice considering the trade-off between the packing quality and computation time. Note that this value of ϵ_{term} was used by Sugihara and Kim in 2004 [2].

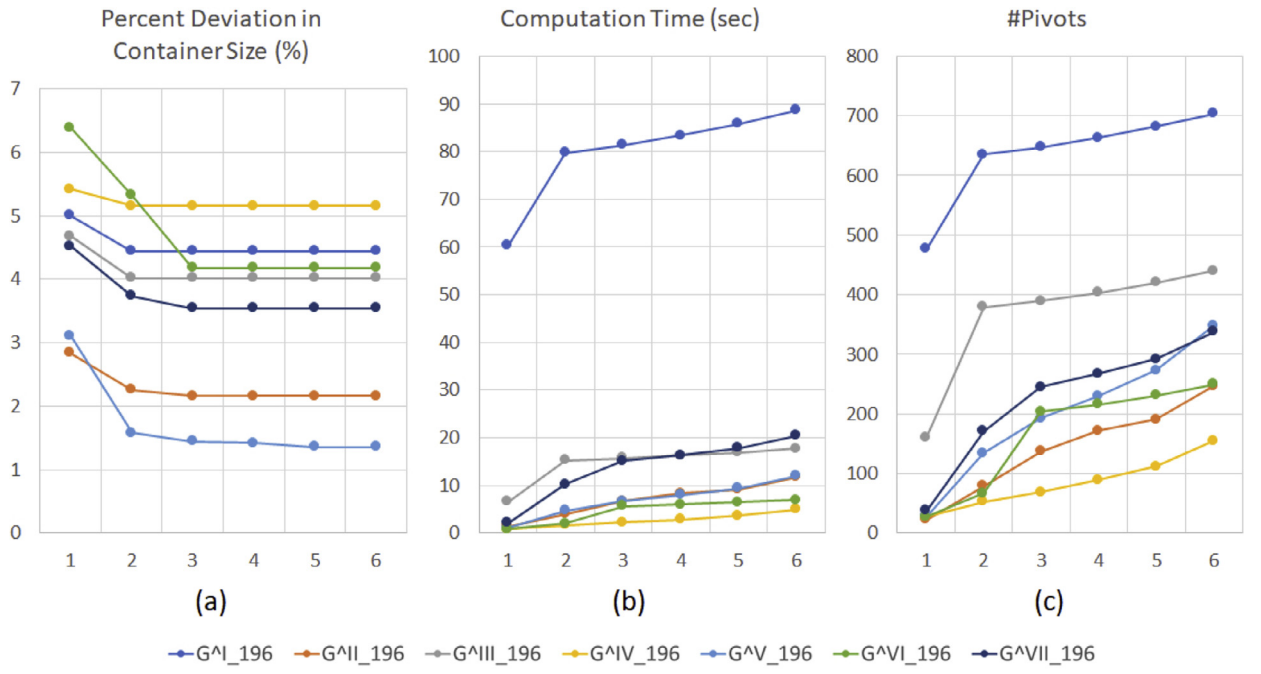

Fig. 23. The influence of the terminating tolerance on the packing quality and computation time. $k = 1, 2, \ldots, 6$ in the horizontal axis denotes $\epsilon_{term} = r_{min} * 10^{-k}$ (r_{min} is the radius of the smallest disk in each disk set). (a) Packing quality (measured in the percentage deviation from the best-known packing). (b) Computation time (s). (c) Total number of pivots (corresponding to protruding disks). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

References

[1] R.J. Fowler, M.S. Paterson, S.L. Tanimoto, Optimal packing and covering in the plane are np-complete, Inf. Process. Lett. 12 (3) (1981) 133–137.
[2] K. Sugihara, M. Sawai, H. Sano, D.-S. Kim, D. Kim, Disk packing for the estimation of the size of a wire bundle, Jpn. J. Ind. Appl. Math. 21 (3) (2004) 259–278.
[3] C.R. Collins, K. Stephenson, A circle packing algorithm, Comput. Geom. 25 (3) (2003) 233–256.
[4] B. Mohar, A polynomial time circle packing, Discr. Math. 117 (1993) 257–263.
[5] G.L. Orick, K. Stephenson, C. Collins, A linearized circle packing algorithm, Comput. Geom. 64 (2017) 13–29.
[6] R.L. Graham, J.C. Lagarias, C.L. Mallows, A.R. Wilks, C.H. Yan, Apollonian circle packings: number theory, J. Number Theory 100 (1) (2003) 1–45.
[7] A. Kontorovich, H. Oh, Apollonian circle packings and closed horospheres on hyperbolic 3-manifolds, J. Am. Math. Soc. 24 (3) (2011) 603–648.
[8] H. Li, T. Li, Recursive sequences in the ford sphere packing, Chaos, Solitons Fract. 106 (2018) 94–106.
[9] P.G. Szabo, M.C. Makrot, T. Csendes, E. Specht, L.G. Casado, I. Garcia, New Approaches to Circle Packing in a Square: With program codes, Springer, 2007.
[10] M. Hifi, R. M'Hallah, A literature review on circle and sphere packing problems: models and methodologies, Adv. Oper. Res. 2009 (2009) 1–22.
[11] B. Segre, K. Mahler, On the densest packing of circles, Am. Math. Mon. 51 (5) (1944) 261–270.
[12] A. Thue, Über die dichteste Zusammenstellung von kongruenten Kreisen in einer Ebene, Christiania : J. Dybwad, 1910.
[13] F. Pfender, G.M. Ziegler, Kissing numbers, sphere packings, and some unexpected proofs, Notices-Am. Math. Soc. 51 (8) (2004) 873–883.
[14] L. Fejes Tóth, Über einen geometrischen Satz, Math. Z. 46 (1) (1940) 83–85.
[15] J. Schaer, The densest packing of 9 circles in a square, Can. Math. Bull. 8 (3) (1965) 273–277.
[16] J. Schaer, On the densest packing of spheres in a cube, Can. Math. Bull. 9 (3) (1966) 265–270.
[17] S. Kravitz, Packing cylinders into cylindrical containters, Math. Mag. 40 (2) (1967) 65–71.
[18] H.S.M. Coxeter, M.G. Greening, R. Graham, J. Ratz, Sets of points with given minimum separation (solution to problem e1921), Am. Math. Mon. 75 (2) (1968) 192–193.
[19] M. Goldberg, Packing of 14, 16, 17 and 20 circles in a circle, Math. Mag. 44 (3) (1971) 134–139.
[20] G.E. Reis, Dense packing of equal circles within a circle, Math. Mag. 48 (1) (1975) 33–37.
[21] H. Melissen, Densest packings of eleven congruent circles in a circle, Geom. Dedic. 50 (1) (1994) 15–25.
[22] K.A. Dowsland, Optimising the palletisation of cylinders in cases, OR Spektrum 13 (4) (1991) 204–212.
[23] H.J. Fraser, J.A. George, Integrated container loading software for pulp and paper industry, Eur. J. Oper. Res. 77 (3) (1994) 466–474.
[24] K.A. Dowsland, W.B. Dowsland, Packing problems, Eur. J. Oper. Res. 56 (1) (1992) 2–14.
[25] H. Isermann, Heuristiken zur lösung des zweidimensionalen packproblems für rundgefäße, Oper.-Res.-Spektrum 13 (4) (1991) 213–223.
[26] B.D. Lubachevsky, How to simulate billiards and similar systems, J. Comput. Phys. 94 (2) (1991) 255–283.
[27] B.D. Lubachevsky, F.H. Stillinger, Geometric properties of random disk packing, J. Stat. Phys. 60 (5) (1990) 561–583.
[28] B.D. Lubachevsky, F.H. Stillinger, E.N. Pinson, Disks vs. spheres: contrastion properties of random packings, J. Stat. Phys. 64 (3) (1991) 501–524.
[29] R.L. Graham, B.D. Lubachevsky, Dense packings of equal disks in an equilateral triangle, Electron. J. Comb. 2 (1995) 1–39.
[30] R.L. Graham, B.D. Lubachevsky, Dense packings of equal disks in an equilateral triangle: from 22 to 34 and beyond, Electron. J. Comb. 2 (1995) 1–39.
[31] R.L. Graham, B.D. Lubachevsky, Repeated patterns of dense packings of equal disks in a square, Electron. J. Comb. 3 (1) (1996) 1–17.
[32] B.D. Lubachevsky, R. Graham, Dense packings of congruent circles in rectangles with a variable aspect ratio, In: Aronov B., Basu S., Pach J., Sharir M. (eds). Discrete and Computational Geometry. Algorithms and Combinatorics, vol 25, Springer: Berlin, Heidelberg, 2003.
[33] B.D. Lubachevsky, R.L. Graham, Curved hexagonal packings of equal disks in a circle, Discr. Comput. Geom. 18 (1997) 179–194.
[34] R. Graham, B. Lubachevski, K. Nurmela, P. Ostergard, Dense packings of congruent circles in a circle, Discr. Math. 181 (1-3) (1998) 139–154.
[35] B.D. Lubachevsky, R.L. Graham, Minimum perimeter rectangles that enclose congruent non-overlapping circles, Discr. Math. 309 (8) (2009) 1947–1962.
[36] M. Gavrilova, J. Rokne, D. Gavrilov, Dynamic collision detection algorithms in computational geometry, in: 12th European Workshop on CG,(1996), 1996, pp. 103–106.
[37] C.D. Maranas, C.A. Floudas, P.M. Pardalos, New results in the packing of equal circles in a square, Discr. Math. 142 (1-3) (1995) 287–293.
[38] Z. Drezner, E. Erkut, Solving the continuous p-dispersion problem using non-linear programming, J. Oper. Res. Soc. 46 (4) (1995) 516–520.
[39] J.A. George, J.M. George, B.W. Lamar, Packing different-sized circles into a rectangular container, Eur. J. Oper. Res. 84 (3) (1995) 693–712.
[40] W. Huang, R. Xu, Two personification strategies for solving circles packing problem, Sci. China (Series E) 42 (6) (1999) 595–602.
[41] H.-X. Huang, H.-A. Liang, P.M. Pardalos, Some properties for the euclidean distance matrix and positive semidefinite matrix completion problems, J. Global Opt. 25 (1) (2003) 3–21.
[42] W. Huang, K. Yan, A short note on a simple search heuristic for the diskspacking problem, Ann. Oper. Res. 131 (1-4) (2004) 101–108.
[43] W. Huang, M. Chen, Note on: An improved algorithm for the packing of unequal circles within a larger containing circle, Comput. Ind. Eng. 50 (3) (2006) 338–344.
[44] Z. Lu, W. Huang, Perm for solving circle packing problem, Comput. Oper. Res. 35 (5) (2008) 1742–1755.
[45] T. Ye, W. Huang, Z. Lü, Iterated tabu search algorithm for packing unequal circles in a circle, 2013. arXiv preprint arXiv: 1306.0694.
[46] Z. Zeng, X. Yu, K. He, W. Huang, Z. Fu, Iterated tabu search and variable neighborhood descent for packing unequal circles into a circular container, Eur. J. Oper. Res. 250 (2) (2016) 615–627.
[47] D.-F. Zhang, A.-S. Deng, An effective hybrid algorithm for the problem of packing circles into a larger containing circle, Comput. Oper. Res. 32 (8) (2005) 1941–1951.
[48] N. Mladenović, F. Plastria, D. Urošević, Reformulation descent applied to circle packing problems, Comput. Oper. Res. 32 (9) (2005) 2419–2434.
[49] J. Kallrath, Cutting circles and polygons from area-minimizing rectangles, J. Global Opt. 43 (2-3) (2009) 299–328.
[50] M. Hifi, R. M'Hallah, A dynamic adaptive local search algorithm for the circular packing problem, Eur. J. Oper. Res. 183 (3) (2007) 1280–1294.
[51] N. Megiddo, Linear-time algorithms for linear programming in r^3 and related problems, SIAM J. Comput. 12 (4) (1983) 759–776.
[52] E. Welzl, Smallest enclosing disks (balls and ellipsoids), in: Proceedings of the New Results and New Trends in Computer Science, in: Lecture Notes in Computer Science, 555, 1991, pp. 359–370.
[53] M. Hifi, R. M'Hallah, Adaptive and restarting techniques-based algorithms for circular packing problems, Comput. Opt. Appl. 39 (2008) 17–35.
[54] H. Akeb, M. Hifi, R. M'Hallah, A beam search algorithm for the circular packing problem, Comput. Oper. Res. 36 (5) (2009) 1513–1528.
[55] C. Wormser, Generalized Voronoi Diagrams and Applications, University of Nice-Sophia Antipolis, 2008 Ph.D. thesis.
[56] L. Lu, Y.-K. Choi, F. Sun, W. Wang, Variational Circle Packing based on Power Diagram, 2011. Technical Report
[57] E. Specht, A precise algorithm to detect voids in polydisperse circle packings, Proc. R. Soc. 471 (2182) (2016) 1–19.
[58] C. López, J. Beasley, Packing unequal circles using formulation space search, Comput. Oper. Res. 40 (5) (2013) 1276–1288.
[59] G. Valette, A better packing of ten equal circles in a square, Discr. Math. 76 (1) (1989) 57–59.
[60] K.J. Nurmela, P.R.J. Östergård, Packing up to 50 equal circles in a square, Discr. Comput. Geom. 18 (1) (1997) 111–120.
[61] M. Goldberg, The packing of equal circles in a square, Math. Mag. 43 (1) (1970) 24–30.
[62] M. Mollard, C. Payan, Some progress in the packing of equal circles in a square, Discr. Math. 84 (3) (1990) 303–307.
[63] E. Specht, High density packings of equal circles in rectangles with variable aspect ratio, Comput. Oper. Res. 40 (1) (2013) 58–69.
[64] J. Kallrath, M.M. Frey, Packing circles into perimeter-minimizing convex hulls, J. Global Opt. (2018) 1–37. Doi: 10.1007/s10898-018-0724-0
[65] J. Machchhar, G. Elber, Dense packing of congruent circles in free-form non-convex, Comput. Aided Geom. Des. 52-53 (2017) 13–27.
[66] J. Kallrath, S. Rebennack, Cutting ellipses from area-minimizing rectangles, J. Global Opt. 59 (2-3) (2014) 405–437.
[67] Y. Stoyan, A. Pankratov, T. Romanova, Quasi-phi-functions and optimal packing of ellipses, J. Global Opt. 65 (2) (2016) 283–307.

[68] A. Pankratov, T. Romanova, I. Litvinchev, Packing ellipses in an optimized rectangular container, Wirel. Netw. (2018) 1–18, doi:10.1007/s11276-018-1890-1.
[69] F.J. Kampas, J.D. Pinter, I. Castillo, Optimal Packing of General Ellipses in a Circle, Springer, 2016.
[70] S.I. Galiev, M.S. Lisafina, Linear models for the approximate solution of the problem of packing equal circles into a given domain, Eur. J. Oper. Res. 230 (3) (2013) 505–514.
[71] E.G. Birgin, R.D. Lobato, A nonlinear programming model with implicit variables for packing ellipsoids, J. Global Opt. 68 (3) (2017) 467–499.
[72] J. Kallrath, Packing ellipsoids into volume-minimizing rectangular boxes, J. Global Opt. 67 (1-2) (2017) 151–185.
[73] F.M. Schaller, S.C. Kapfer, J.E. Hilton, P.W. Cleary, K. Mecke, C.D. Michele, T. Schilling, M. Saadatfar, M. Schroter, G.W. Delaney, Non-universal voronoi cell shapes in amorphous ellipsoid packs, Lett. J. Explor. Front. Phys. 111 (2) (2015) 1–6.
[74] D.N. Ilin, M. Bernacki, Advancing Layer Algorithm of Dense Ellipse Packing for Generating Statistically Equivalent Polygonal Structures, Springer, 2016.
[75] J.Q. Gan, Z.Y. Zhou, A.B. Yu, Interparticle force analysis on the packing of fine ellipsoids, Powder Technol. 320 (2017) 610–624.
[76] K. Kildashti, K. Dong, B. Samali, Explicit force model for discrete modelling of elliptical particles, Comput. Geotech. Vol. Comput. Geotech. 110 (2019) 122–131.
[77] M. Lee, Q. Fang, Y. Cho, J. Ryu, L. Liu, D.-S. Kim, Support-free hollowing for 3d printing via Voronoi diagram of ellipses, Comput.-Aided Des. 101 (2018) 23–36.
[78] Y. You, Y. Zhao, Discrete element modelling of ellipsoidal particles using super-ellipsoids and multi-spheres: a comparative study, Powder Technol. 331 (15) (2018) 179–191.
[79] S. Zhao, T.M. Evans, X. Zhou, Three-dimensional Voronoi analysis of monodisperse ellipsoids during triaxial shear, Powder Technol. 323 (1) (2018) 323–336.
[80] S. Zhao, N. Zhang, X. Zhou, L. Zhang, Particle shape effects on fabric of granular random packing, Powder Technol. 310 (2017) 175–186.
[81] L. Riley, L. Schirmer, T. Segura, Granular hydrogels: emergent properties of jammed hydrogel microparticles and their applications in tissue repair and regeneration, Curr. Opin. Biotechnol. 60 (2019) 1–8.
[82] T. Hales, M. Adams, G. Bauer, D.T. Dang, J. Harrison, T.L. Hoang, C. Kaliszyk, V. Magron, S. McLaughlin, T.T. Nguyen, T.Q. Nguyen, T. Nipkow, S. Obua, J. Pleso, J. Rute, A. Solovyev, A.H.T. Ta, T.N. Tran, D.T. Trieu, J. Urban, K.K. Vu, R. Zumkeller, A formal proof of the kepler conjecture, Forum of Mathematics, Pi. 5 (2) (2017) 1–21.
[83] Y. Stoyan, G. Yaskov, Packing equal circles into a circle with circular prohibited areas, Int. J. Comput. Math. 89 (10) (2012) 1355–1369.
[84] A. Okabe, B. Boots, K. Sugihara, S.N. Chiu, Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd, John Wiley & Sons, Chichester, 1999.
[85] F. Aurenhammer, R. Klein, D.-T. Lee, Voronoi Diagrams and Delaunay Triangulations, World Scientific, 2013.
[86] F.P. Preparata, M.I. Shamos, Computational Geometry: An Introduction, Springer-Verlag, 1985.
[87] M. Mäntylä, An Introduction to Solid Modeling, W.H. Freeman & Company, New York, 1988.
[88] K. Sugihara, M. Iri, A Solid Modelling System Free from Topological Inconsistency, J. Inf. Process. 12 (4) (1989) 380–393.
[89] D.-S. Kim, I.-K. Hwang, B.-J. Park, Representing the Voronoi diagram of a simple polygon using rational quadratic Bézier curves, Comput.-Aided Des. 27 (8) (1995) 605–614.
[90] C.-K. Yap, An $O(n\log n)$ algorithm for the Voronoi diagram of a set of simple curve segments, Discr. Comput. Geom. 2 (1987) 365–393.
[91] L. Jin, D. Kim, L. Mu, D.-S. Kim, S.-M. Hu, A sweepline algorithm for Euclidean Voronoi diagram of circles, Comput.-Aided Des. 38 (3) (2006) 260–272.
[92] D.T. Lee, R.L. Drysdale, Generalization of Voronoi diagrams in the plane, SIAM J. Comput. 10 (1) (1981) 73–87.
[93] M. Sharir, Intersection and closest-pair problems for a set of planar discs, SIAM J. Comput. 14 (2) (1985) 448–468.
[94] M. Lee, K. Sugihara, D.-S. Kim, Topology-oriented incremental algorithm for the robust construction of the Voronoi diagrams of disks, ACM Trans. Math. Softw. 43 (2) (2016) 14:1–14:23.
[95] D.-S. Kim, D. Kim, K. Sugihara, Voronoi diagram of a circle set from Voronoi diagram of a point set: I. topology, Comput. Aided Geom. Des. 18 (2001) 541–562.
[96] D.-S. Kim, D. Kim, K. Sugihara, Voronoi diagram of a circle set from Voronoi diagram of a point set: II. geometry, Comput. Aided Geom. Des. 18 (2001) 563–585.
[97] K. Lee, Principles of CAD/CAM/CAE Systems, Addison-Wesley, Boston, 1999.
[98] D. Kim, D.-S. Kim, K. Sugihara, Euclidean Voronoi diagram for circles in a circle, Int. J. Comput. Geom. Appl. 15 (2) (2005) 209–228.
[99] D.-S. Kim, Polygon offsetting using a Voronoi diagram and two stacks, Comput.-Aided Des. 30 (14) (1998) 1069–1076.
[100] Al-Mudahka, M. Hifi, R. M'Hallah, Packing circles in the smallest circle: an adaptive hybrid algorithm, J. Oper. Res. Soc. 62 (2011) 1917–1930.