A Novel Approximation Algorithm for Max-Covering Circle Problem

Kaiqi Zhang[1(✉)], Siyuan Zhang[1], Jirun Gao[1], Hongzhi Wang[1], Hong Gao[2], and Jianzhong Li[1,3]

[1] School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
{zhangkaiqi,wangzh,lijzh}@hit.edu.cn

[2] School of Computer Science and Technology, Zhejiang Normal University, Jinhua, China
honggao@zjnu.edu.cn

[3] Faculty of Computer Science and Control Engineering, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China

Abstract. We study the efficient approximation algorithm for max-covering circle problem. Given a set of weighted points in the plane and a circle with specified size, max-covering circle problem is to find the proper place where the center of the circle is located so that the total weight of the points covered by the circle is maximized. Our core approach is to approximate the circle with a symmetrical rectilinear polygon (SRP). We first present a method to construct the circumscribed SRP of a given circle and disclose their area relationship. Then, we convert max-covering SRP problem to SRP intersection problem, which can be efficiently solved with simple partition and modification based on the existing method. Finally, the optimal solution returned from max-covering SRP problem can be used to produce an approximate answer to max-covering circle problem. We prove that for most of the inputs, our algorithm can give a $(1-\varepsilon)$ approximation to the optimal solution, which only needs $O\left(n\varepsilon^{-1}\log n + n\varepsilon^{-1}\log\left(\frac{1}{\varepsilon}\right)\right)$ time for unit points and $o\left(n\varepsilon^{-2}\log n\right)$ time for weighted points.

Keywords: Max-covering problem · Symmetrical rectilinear polygon · $(1-\varepsilon)$ approximation

1 Introduction

Max-covering problem is a fundamental operation in computational geometry and database community. Given a set of weighted points Q in the plane $\mathbb{R}^2$ and a planar geometry G with specified size, max-covering problem is to find the proper place where the center of G is located so that the total weight of the points covered by G is maximized. In a word, max-covering problem aims to cover the points with maximum weight in a fixed-size region. There are many

W. Wu and J. Guo (Eds.): COCOA 2023, LNCS 14461, pp. 226–238, 2024.
https://doi.org/10.1007/978-3-031-49611-0_16

studies on this problem that we only present some important results on rectangle and circle.

For max-covering rectangle problem, two efficient exact algorithms are presented with $O(n \log n)$ time complexity where n is the number of points. Imai *et al.* [14] solve max-covering rectangle problem by finding a maximum clique of an intersection graph of rectangles in the plane. An alternative algorithm is proposed by Nandy and Bhattacharya in [19]. They transform max-covering rectangle problem into rectangle intersection problem and employ plane-sweep technique and interval tree data structure [20] to find the intersection region with maximum weight. Up to now, the time complexity has not been improved. Hemmer *et al.* [12] solve this algorithm regardless of whether the boundaries of the rectangles are open or closed. Tao *et al.* [21] design a grid-sampling-based approximation algorithm, which obtains a $(1-\varepsilon)$-approximate answer with extremely high probability in $O\left(n \log \frac{1}{\varepsilon} + n \log \log n\right)$ time. Due to the weak scalability of the above in-memory algorithms, Choi *et al.* [5] propose an I/O-optimal external-memory algorithm. Later on, this problem is extended to trajectories and exact and approximation algorithms are proposed in [22].

For max-covering circle problem, Chazelle and Lee [4] substantially give an exact algorithm that runs in $O(n^2)$ time. It is widely acknowledged that the time complexity of any exact algorithm for this problem could not be improved any further, because it has been proved to be a 3SUM-HARD problem [2], which means a lower bound of $\Omega\left(n^2\right)$ running time must be held by any exact algorithms. Mark De Berg *et al.* [8] present a method that gives a $(1-\varepsilon)$-approximate solution in a deterministic $O\left(n \log n + n\varepsilon^{-3}\right)$ time. Although Choi *et al.* [5] give a $\frac{1}{4}$-approximate solution in $O(n \log n)$ time, the approximation accuracy cannot be specified by users. A special case of this problem argues that points have unit weight. Thus this problem changes to compute the maximum number of covered points by a circle. Aronov and Har-Peled [2] propose a random sampling method that returns a $(1-\varepsilon)$- approximate solution in $O\left(n\varepsilon^{-2} \log n\right)$ time with high probability.

Besides max-covering problem, another type of geometric covering problem is k-covering problem. It aims to find a smallest geometry G covering at least k $(k \leq n)$ points of Q. For k-covering circle problem, Efrat *et al.* [9] give an exact algorithm for $O(nk\log^2 n)$ time. Subsequently, Matouvsek *et al.* [17] present a stochastic algorithm that runs in $O(n \log n + nk)$ expected time. All the above exact algorithms degenerate to $O(n^2)$ when $k = O(n)$. Har-Peled *et al.* [11] improve the exact result to $O(nk)$ and give a stochastic approximation algorithm that runs in $O\left(n + n \cdot \min\left(\frac{1}{k\varepsilon^3} \log^2 \frac{1}{\varepsilon}, k\right)\right)$ expected time. The case where G is an axis-aligned rectangle or square has also been studied somewhat for k-covering problem. When G is an axis-aligned square, the best result belongs to Mahapatra [16]. He gives the algorithm with time complexity $O\left(n + (n-k)\log^2(n-k)\right)$. The idea is to transform k-covering problem into iteratively solving max-covering problem by using the characteristics of the square. When G is an axis-aligned rectangle, the objective of solving k-covering problem usually falls into two categories: minimizing the perimeter of the rectangle and minimizing the area of

the rectangle. For the former, the first two algorithms with $O\left(n^3\right)$ time complexity and $O\left(k^2 n \log n\right)$ time complexity are given by Aggarwal *et al.* [1]. The second result is subsequently improved by Eppstein & Erickson [10] and Datta *et al.* [6] to $O\left(n \log n + k^2 n\right)$ running time and again by Kaplan *et al.* [15] to $O\left(n \log n + n k^{3/2} \log^2 k\right)$ time complexity. For the latter, there is an $O\left(n^{5/2} \log^2 n\right)$ algorithm given by Kaplan *et al.* [15]. Besides, de Berg *et al.* [7] gives a k-sensitive algorithm with $O\left(n \log^2 n + n k^2 \log n\right)$ time complexity. The optimal result for k-covering problem of rectangular perimeter or area minimization belongs to Timothy M. Chan and Sariel Har-Peled [3]. They propose a general algorithm with $O\left(n^2 \log n\right)$ running time (for both perimeter and area), and, at the same time, a k-sensitive algorithm with $O(n \log n + nk \log k)$ time (perimeter) and a k-sensitive algorithm with $O\left(nk \log \frac{n}{k} \log k\right)$ time (area).

Our Contributions. In this paper, we concentrate on studying the efficient approximation algorithm for max-covering circle problem. From the previous work, we have the following observations. (1) For existing algorithms, solving max-covering rectangle problem has a better time complexity than solving max-covering circle problem. (2) A symmetrical rectilinear polygon (SRP) can approximate the circle in terms of area. Thus, the solution of max-covering SRP problem can be used to approximatively solve max-covering circle problem. These give birth to our core idea. We first present a method to construct the circumscribed SRP of a given circle and disclose their area relationship. The area difference is determined by the number of edges in the circumscribed SRP. Then, we convert max-covering SRP problem to SRP intersection problem, which can be efficiently solved with simple partition and modification based on the existing method. Finally, the optimal solution returned from max-covering SRP problem can be used to produce an approximate answer to max-covering circle problem. We prove that for most of the inputs, our algorithm can give a $(1-\varepsilon)$ approximation to the optimal solution, which only needs $O\left(n\varepsilon^{-1} \log n + n\varepsilon^{-1} \log\left(\frac{1}{\varepsilon}\right)\right)$ time for unit points and $o\left(n\varepsilon^{-2} \log n\right)$ time for weighted points.

2 Preliminaries

Let us consider a set of points Q in 2-dimensional space $\mathbb{R}^2$. Each point $q \in Q$ has a non-negative weight $w(q)$.

Definition 1. *(Covering Weight). Given a set of points Q and a geometry G, the covering weight of G is:*

$$\textit{covering-weight}(G, Q) = \sum_{q \in Q \cap G} w(q)$$

Definition 2. *(Max-Covering Problem). Given a set of weighted points Q and a geometry G with specified size, max-covering problem is to find the proper place of G to maximize the covering weight of G.*

The geometry can be arbitrary shape, including circle, rectangle, rectilinear polygon, etc. Similarly, let G be a circle with a given diameter, this problem can be renamed as max-covering circle (MaxC-C) problem. In the following, we formally define $(1-\varepsilon)$-approximate max-covering circle problem.

Definition 3. *($(1-\varepsilon)$-Approximate Max-Covering Circle Problem). Given a set of points Q and a circle C with specified diameter, for any ε $(0<\varepsilon<1)$, $(1-\varepsilon)$-approximate max-covering circle problem finds a place in $\mathbb{R}^2$ to place C that satisfies*

$$\textit{covering-weight}(C) \geq (1-\varepsilon) \times \textit{covering-weight}(C^*)$$

where C^* is an optimal circle of the original problem.

In this paper, we propose an approximation algorithm for solving MaxC-C problem with a $(1-\varepsilon)$ accuracy to the optimal solution. In other words, our algorithm can solve $(1-\varepsilon)$-approximate MaxC-C problem. We achieve this result by approximatively converting MaxC-C problem to Max-covering symmetrical rectilinear polygon problem. Here, symmetrical rectilinear polygon is a special polygon which will be introduced below.

3 Symmetrical Rectilinear Polygon Construction

We first formulate several definitions of rectilinear polygon and symmetrical rectilinear polygon.

Definition 4. *(Rectilinear Polygon). A polygon is said to be a rectilinear polygon if the following conditions hold in a two-dimensional rectangular coordinate system: (1) For each of the given x-axis and y-axis, each side of this polygon is either perpendicular to the given coordinate axis or parallel to the given coordinate axis. (2) Any two sides of this polygon do not intersect except at the endpoints.*

Definition 5. *(Symmetrical Rectilinear Polygon). A rectilinear polygon RP, assuming that its center is at the origin, is symmetrical if $\forall q(x,y) \in RP$ such that $q_1(-x,y) \in RP$, $q_2(x,-y) \in RP$ and $q_3(-x,-y) \in RP$.*

In a word, symmetrical rectilinear polygon is both centrosymmetric and axisymmetric. To simplify the presentation, symmetrical rectilinear polygon can be abbreviated to SRP. Max-covering symmetrical rectilinear polygon problem can thus be called as MaxC-SRP problem.

Definition 6. *(Circumscribing and Inscribing). Given a circle C and a positive number k, take the diameter (Y-axis-parallel) of the circle and divide it into k segments equally. Then, make $k-1$ vertical lines of this diameter through these k-bisected points respectively and intersect the circle at $2k-2$ points axisymmetrically. Moreover, connect adjacent points on the circumference with line segments perpendicular to and parallel to this diameter. The resulting polygon is said to be the circumscribed SRP of the circle. Similarly, circle C is said to be the inscribed circle of this circumscribed SRP.*

For a given circle, the circumscribed SRP depends only on k. Once k is determined, let S_k be the circumscribed SRP of the circle.

Given a circle, the construction methods of every circumscribed SRP have been presented above. As far as our work goes, that is not the exact point. What we need to do is, for a given parameter ε, choose an appropriate k such that for most of the inputs, the optimal solution of MaxC-SRP problem with respect to S_k can be used to produce a $(1-\varepsilon)$ approximation answer to MaxC-C problem. We denote the part that belongs to S_k but not C as $S_k - C$. It is a simple fact that, the area of $S_k - C$ can be a useful measure of how well the circumscribed SRP S_k approximates the circle C.

Let A_G be the area of any closed geometry G. We can infer the following relationship between A_{S_k-C} and A_{S_k}.

Theorem 1. *e is the base of natural logarithm and ε is in range $(0,1)$, there is an even number $k = O\left(\varepsilon^{-1}\right)$ such that $\frac{A_{S_k-C}}{A_{S_k}} < \frac{\varepsilon}{e}$.*

Proof. We consider the length from the center of the circle to each corner of S_k. Since k is even, we can only consider the part of S_k which is above the horizontal line that goes through the center of the circle. Let r be the radius. For the i-th corner $\left(1 \le i \le \frac{k}{2}\right)$, from bottom to top of this part, the length from the center of the circle to it is $r\sqrt{\left(1+\frac{8i-4}{k^2}\right)}$, with simple geometric derivations.

Obviously, the following inequality holds

$$\pi r^2 < A_{S_k} < \pi r^2 (1 + \frac{4k-4}{k^2})$$

Then, we have

$$\frac{A_{S_k-C}}{A_{S_k}} < \frac{\frac{4k-4}{k^2}}{1+\frac{4k-4}{k^2}} < \frac{4k-4}{k^2}$$

We just choose

$$k = 2\lfloor \frac{1 + \lceil 4e\varepsilon^{-1} \rceil}{2} \rfloor$$

This guarantees $4e\varepsilon^{-1} \le k \le 4e\varepsilon^{-1} + 1$ and $2|k$. Therefore, there is an even number k, which satisfies

$$k = O\left(\varepsilon^{-1}\right)$$
$$\frac{A_{S_k-C}}{A_{S_k}} < \frac{4k-4}{k^2} < \frac{4}{k} < \frac{\varepsilon}{e}$$

4 Algorithm for MaxC-SRP Problem

In this section, we discuss how to solve MaxC-SRP problem. In fact, this problem is not difficult and we can solve it completely with existing techniques.

Given a set of geometries $\mathcal{G}$ and a point q in $\mathbb{R}^2$, let $\mathcal{G}_q$ be the set of geometries covering q, denoted by $\mathcal{G}_q = \{G \mid \forall G \in \mathcal{G}, q \in G\}$. Each geometry $G \in \mathcal{G}$ has a non-negative weight $w(G)$.

Definition 7. *(Geometry Intersection Problem). Given a set of weighted geometries $\mathcal{G}$, the intersection problem of $\mathcal{G}$ is to find a point $q \in \mathbb{R}^2$ that maximizes*

$$\textit{covered-weight}(q, \mathcal{G}) = \sum_{G \in \mathcal{G}_q} w(G)$$

Given a point q and a circumscribed SRP SRP, let SRP_q be the circumscribed SRP whose center point is at q and size is the same with SRP. For a set of points Q, we define $SRP_Q = \{SRP_q \mid \forall q \in Q\}$.

Lemma 1. *Given a set of weighted points Q, $\forall q \in \mathbb{R}^2$, covering-weight$(SRP_q, Q)$ = covered-weight(q, SRP_Q).*

Proof. We first prove the following two statements:

(1)$\forall q' \in Q$, if q' is covered by SRP_q, q is also covered by $SRP_{q'}$.

We let the coordinates of q be (x_q, y_q) and the coordinates of q' be $(x_{q'}, y_{q'})$. By the centrosymmetry of SRP in the definition, the fact that SRP_q contains $q'(x_{q'}, y_{q'})$ implies that it also contains $q''(2x_q - x_{q'}, 2y_q - y_{q'})$. Considering translating SRP_q so that its center becomes q', we obtain $SRP_{q'}$, which contains q, since q is the position of q'' after translation. Now, statement (1) is proved.

(2)$\forall q' \in Q$, if q' is not covered by SRP_q, q is not covered by $SRP_{q'}$ either.

We assume that $\exists q \in \mathbb{R}^2$, q' is not contained by SRP_q, but q is contained by $SRP_{q'}$. If so, then by centrosymmetry it is equally possible to obtain that $SRP_{q'}$ contains the point $q''(2x_{q'} - x_q, 2y_{q'} - y_q)$. Thus, after translating the center of $SRP_{q'}$ to q, the point corresponding to q'' is q'. This implies that SRP_q contains the point q', contradicting the assumption, and thus statement (2) holds.

Based on the above statements, we have

covering-weight$(SRP_q, Q) = \sum_{q' \in Q_{covered}} w(q') = \sum_{SRP \in SRP_Q \, covering \; q} w$ (SRP) = covered-weight(q, SRP_Q).

Theorem 2. *Given a set of weighted points Q, let q^* is the answer to intersection problem of SRP_Q, then SRP_{q^*} is an optimal SRP to MaxC-SRP problem in Q.*

Proof. We assume that SRP_{q^*} is not an optimal SRP to MaxC-SRP problem in Q. Then, $\exists q \in \mathbb{R}^2$, covering-weight$(SRP_q, Q)$ > covering-weight(SRP_{q^*}, Q). With Lemma 1, we have both covering-weight(SRP_q, Q) = covered-weight$(q,$ $SRP_Q)$ and covering-weight(SRP_{q^*}, Q) = covered-weight(q^*, SRP_Q). This shows that covered-weight(q, SRP_Q) > covered-weight(q^*, SRP_Q), which implies that q^* is not the answer to intersection problem of SRP_Q, contradicting our premise. Therefore, the theorem is proved.

Now we just need to give the method to solve the intersection problem of n SRPs. We use the division method to divide each SRP into several rectangles with parallel axes. We specify that for each SRP, the tangent along the line where its respective horizontal edges is located. Due to symmetry, let the number of horizontal edges to the left of its axis be l. Then each SRP is divided into

$l-1$ rectangles. With a known input SRP, the time required to compute these rectangles is $O(l)$: we scan the SRP horizontally from top to bottom and group its horizontal boundaries two by two. For n SRPs, it takes $O(nl)$ time to compute all divided $n(l-1)$ rectangles. Meanwhile, we want these $n(l-1)$ rectangles to satisfy that the individual rectangles divided by the same SRP are disjoint, which requires determining the attribution of common edges between these rectangles at the time of division. We simply specify that the common edge of two adjacent rectangles divided by the same SRP belongs to the rectangle that is entirely above this edge.

Obviously, computing the intersection problem of these n SRPs is equivalent to settle the intersection problem of corresponding $n(l-1)$ axis-parallel rectangles. This has been proved in [22]. The existing sweepline algorithm [19] solves this problem well by using an interval tree for updating and counting events, returning the region with the greatest total weight and the corresponding weight. In the original MaxC-SRP problem, the former of the returned results corresponds to the positions where the SRP can be placed and the latter corresponds to the total weight of points it covers at most. Therefore, our algorithm is to divide the SRP into rectangles and then use the sweepline algorithm for these $n(l-1)$ rectangles. Considering that these rectangles may have open or closed boundaries (open boundaries may exist only in the horizontal direction), we use the improved algorithm [12], which is able to handle this case with no increase in time complexity and space complexity compared to the classical sweepline algorithm. When this algorithm is applied to solve our problem, it has a time complexity of $O(nl(\log n + \log l))$ and a space complexity of $O(nl)$. The total time complexity of this algorithm is the same as it, for the time of dividing is $O(nl)$.

5 Approximation Algorithm for MaxC-C Problem

Now, we know that the MaxC-SRP algorithm, based on binary tree search, returns the correct result in deterministic $O(nl(\log n + \log l))$ time and $O(nl)$ space when each SRP has $O(l)$ edges. Actually, this algorithm can be used to approximatively compute the MaxC-C problem. Given a set of points Q and a specified circle C, we first construct the circumscribed SRP S_k of C based on a given approximate accuracy ε. Then, the optimal SRP SRP^* can be returned by MaxC-SRP algorithm. Finally, the inscribed circle of SRP^* can be a desirable approximation to the optimal solution of MaxC-C problem for most of the inputs of Q. For a few specific inputs of Q, our result is not a satisfying approximation. Obviously, the correctness of the approximation degree depends on the distribution of points (*i.e.*, the inputs of Q). We would like to analyze whether the algorithm can give correct results for various different inputs and discuss its performance.

Assuming a random distribution of n points in the plane is clearly not a good choice. To discuss the various inputs without bias, we assume a random distribution of m points inside the SRP (the remaining $n-m$ points are outside

the SRP). For practical applications with a large number of different inputs, we believe that this assumption is reasonable. For a given ε, we use our MaxC-SRP algorithm on S_k, where $k = O\left(\varepsilon^{-1}\right)$ and $\frac{A_{S_k - C}}{A_{S_k}} < \frac{\varepsilon}{e}$ are both satisfied. Then, for each point within S_k, let the probability that it falls within C be $1 - p$ and the probability that it falls within $S_k - C$ be p. By Theorem 1 and the assumptions, we thus easily know that $p < \frac{\varepsilon}{e}$. Let the total weight of these m points falling within $S_k - C$ be the random variable X.

5.1 Points with Unit Weight

If each point has unit weight, max-covering problem changes to compute the maximum number of covered points and X represents the number of these m points falling within $S_k - C$. Hence, X follows a binomial distribution, that is $X \sim B(m, p)$. The mathematical expectation $E(X)$ of X is mp. Based on Chernoff inequality [18], we have the following bound.

Theorem 3. *Let $Y_1, Y_2, ..., Y_m$ be m independent Bernoulli trials, where $Pr\{Y_i = 1\} = p_i$ and $Pr\{Y_i = 0\} = 1 - p_i$ for $i = 1, 2, ..., m$. Define $Y = \sum_{i=1}^{m} Y_i$. Then, for any $\varepsilon > 0$,*

$$Pr\left\{Y \geq (1+\varepsilon)E(Y)\right\} \leq \left(\frac{e^{\varepsilon}}{(1+\varepsilon)^{1+\varepsilon}}\right)^{E(Y)}$$

X coincides with Theorem 3, we thus have

$$Pr\left\{X \geq m\varepsilon\right\} < Pr\left\{X \geq emp\right\} = Pr\left\{X \geq (1+e-1)E(X)\right\} \leq e^{-E(X)}$$

For any $0 < p < \frac{\varepsilon}{e}$, if $p \geq \frac{\varepsilon}{3e}$, then

$$Pr\left\{X \geq m\varepsilon\right\} < e^{-E(X)} = e^{-mp} \leq e^{-\frac{m\varepsilon}{3e}}$$

For $0 < p < \frac{\varepsilon}{3e}$, let $\frac{\varepsilon}{3e} \leq p^* < \frac{\varepsilon}{e}$ and its corresponding random variable to be X^*, $X^* \sim B(m, p^*)$. It is obvious that

$$Pr\left\{X \geq m\varepsilon\right\} < Pr\left\{X^* \geq m\varepsilon\right\} < e^{-\frac{m\varepsilon}{3e}}$$

Therefore, for given m and ε, $Pr\left\{X \geq m\varepsilon\right\} < e^{-\frac{m\varepsilon}{3e}}$ always holds. Now we discuss the value of this probability. We consider two cases separately. When $m = \Omega\left(\varepsilon^{-1}\log n\right)$, we can easily say that our algorithm gives $(1-\varepsilon)$ approximations for most inputs. This is because

$$Pr\left\{X \geq m\varepsilon\right\} < e^{-\frac{m\varepsilon}{3e}} < n^{-\frac{c}{3e}}$$

where c is a suitable constant. Then we have

$$Pr\left\{(m - X) \geq m(1-\varepsilon)\right\} > 1 - n^{-\frac{c}{3e}}$$

We know that $m-X$ represents the number of points covered by the inscribed circle of SRP. Therefore, we know that under our assumption

$$Pr\left\{\frac{(m-X)}{m} \geq (1-\varepsilon)\right\} > 1 - n^{-\frac{c}{3e}}$$

We note the fact that the maximum number of points that can be covered by the circumscribed SRP of a circle must be not less than the maximum number of points that can be covered by this circle. Let C^* be the optimal circle of the MaxC-C problem and $m^* = \textit{covering-weight}(C^*)$. We have

$$\begin{aligned} &m^* \leq m \\ &Pr\left\{\frac{(m-X)}{m^*} \geq (1-\varepsilon)\right\} \geq Pr\left\{\frac{(m-X)}{m} \geq (1-\varepsilon)\right\} \quad > 1 - n^{-\frac{c}{3e}} \end{aligned}$$

The above equation tells us that, when $m = \Omega\left(\varepsilon^{-1}\log n\right)$, for most of the inputs (at least $1 - n^{-c_1}$ times of the total input ($c_1 = \frac{c}{3e}$)), the inscribed circle of the SRP returned by our algorithm can be used as a $(1-\varepsilon)$ approximation to the optimal solution. From Sect. 4, we know that the MaxC-SRP algorithm takes $O\left(n\varepsilon^{-1}\log n + n\varepsilon^{-1}\log\left(\frac{1}{\varepsilon}\right)\right)$ running time when $l = O(k) = O\left(\varepsilon^{-1}\right)$.

For the other cases (*i.e.*, $m = o\left(\varepsilon^{-1}\log n\right)$), the order of magnitude of m is not sufficient for our algorithm to return satisfactory solutions for most inputs. However, due to $m^* \leq m$, we know that the maximum number of points that can be covered by a given circle does not exceed m. This is equivalent to the fact that the depth of the deepest point in the arrangement of a set of circles does not exceed m. Another fact is that m is already computed when we run the algorithm. Now we simply use the conclusion [2] that, given any set X of n psedocircles, we can compute the deepest point in $\mathcal{A}(X)$ in $O(nd + n\log n)$, where $d = \textit{depth}(X) = o\left(\varepsilon^{-1}\log n\right)$. Therefore, we just use the exact algorithm to compute the solution when m is not big enough. The time cost is $o\left(n\varepsilon^{-1}\log n\right)$.

Given n points Q, circle radius r and approximation accuracy ε, the basic procedure of the algorithm for $(1-\varepsilon)$-approximate MaxC-C problem is as follows.

Step 1. Construct S_k based on given r and ε.

Step 2. Compute the optimal SRP SRP^* of MaxC-SRP problem based on Q and S_k.

Step 3. Count the number of points inside SRP^*, denoted by m.

Step 4. Make a judgment based on the value of m: (1) If $m \geq c\varepsilon^{-1}\log n$ where c is an arbitrarily chosen positive real number, the center of the corresponding SRP^* is used as the center of the circle, and r is used as the radius to construct the circle. The circle is returned as the approximate solution of the circle position, and the number of points covered by this circle (which requires additional $O(n)$ time complexity to calculate it) is returned as the approximate solution of the maximum number of points covered by the circle. (2) Otherwise, we directly calculate max-covering circle problem.

In summary, our algorithm finishes running in $O\left(n\varepsilon^{-1}\log n + n\varepsilon^{-1}\log\left(\frac{1}{\varepsilon}\right)\right)$ time and returns a $(1-\varepsilon)$ approximation for at least $1 - n^{-c_1}$ events for all

possible values of m. Compared to almost all previous methods, our algorithm consumes surprisingly less time. It can be used either alone as an approximation algorithm (running when there is some tolerance for errors) or as a preprocessor for other approximation algorithms to reduce their time consumption.

5.2 Weighted Points

If points have different weights, X cannot follow a binomial distribution. The mathematical expectation $E(X)$ of X is Mp where M is the total weight of point inside the SRP. From Hoeffding inequality [13], we have the following bound.

Theorem 4. *Let $Y_1, Y_2, ..., Y_m$ be independent and identically distributed random variables where $Y_i \in [a, b]$ for $i = 1, 2, .., m$. Define $Y = \sum_{i=1}^{m} Y_i$. Then*

$$Pr(Y - E[Y] \geq \alpha) \leq e^{-\frac{2\alpha^2}{m(b-a)^2}}$$

X coincides with Theorem 4, we thus have

$$Pr\{X \geq M\varepsilon\} < Pr\{X \geq eMp\} = Pr\{X - E(X) \geq (e-1)E(X)\} \leq e^{-\frac{2(e-1)^2E(X)^2}{mb^2}}$$

For any $0 < p < \frac{\varepsilon}{e}$, if $p \geq \frac{\varepsilon}{3e}$, then

$$Pr\{X \geq M\varepsilon\} \leq e^{-\frac{2(e-1)^2E(X)^2}{mb^2}} = e^{-\frac{2(e-1)^2M^2p^2}{mb^2}} \leq e^{-\frac{2(e-1)^2M^2\varepsilon^2}{9e^2mb^2}}$$

For $0 < p < \frac{\varepsilon}{3e}$, let $\frac{\varepsilon}{3e} \leq p^* < \frac{\varepsilon}{e}$ and its corresponding random variable to be X^*, X^* coincides with Theorem 4. It is obvious that

$$Pr\{X \geq M\varepsilon\} < Pr\{X^* \geq M\varepsilon\} < e^{-\frac{2(e-1)^2M^2\varepsilon^2}{9e^2mb^2}}$$

Therefore, for given M and ε, $Pr\{X \geq M\varepsilon\} < e^{-\frac{2(e-1)^2M^2\varepsilon^2}{9e^2mb^2}}$ always holds. In fact, $M = \sum_{i=1}^{m} w(q_i) = m\overline{w}$ where $\overline{w} = \frac{\sum_{i=1}^{m} w(q_i)}{m}$. Then

$$Pr\{X \geq M\varepsilon\} < e^{-\frac{2(e-1)^2m\overline{w}^2\varepsilon^2}{9e^2b^2}}$$

We argue that $\overline{w}$ and b are stable and can be regarded as constants. When $m = \Omega(\varepsilon^{-2}\log n)$, we can easily say that our algorithm gives $(1-\varepsilon)$ approximations for most inputs. Because

$$Pr\{X \geq M\varepsilon\} < n^{-c}$$

where c is a suitable constant. Then we have

$$Pr\{(M - X) \geq M(1-\varepsilon)\} > 1 - n^{-c}$$

Similar to points with unit weight, $M - X$ is the total weight of points covered by the inscribed circle of SRP. Thus

$$Pr\left\{\frac{(M-X)}{M} \geq (1-\varepsilon)\right\} > 1 - n^{-c}$$

Let C^* be the optimal circle of the MaxC-C problem and $M^* = \textit{covering-weight}(C^*)$. We have

$$M^* \leq M$$
$$Pr\left\{\frac{(M-X)}{M^*} \geq (1-\varepsilon)\right\} \geq Pr\left\{\frac{(M-X)}{M} \geq (1-\varepsilon)\right\} \quad > 1 - n^{-c}$$

The above equation tells us that, when $m = \Omega\left(\varepsilon^{-2}\log n\right)$, for most of the inputs (at least $1 - n^{-c}$ times of the total input), the inscribed circle of the SRP returned by our algorithm can be used as a $(1-\varepsilon)$ approximation to the optimal solution. Recall Sect. 4 again, MaxC-SRP algorithm takes $O\left(n\varepsilon^{-1}\log n + n\varepsilon^{-1}\log\left(\frac{1}{\varepsilon}\right)\right)$ time when $l = O(k) = O\left(\varepsilon^{-1}\right)$.

For the other cases (*i.e.*, $m = o\left(\varepsilon^{-2}\log n\right)$), the order of magnitude of m is not sufficient for our algorithm to return satisfactory solutions for most inputs. However, due to $M^* \leq M$, that is the total weight of points that can be covered by a given circle does not exceed M. We know that $M^* \leq M \leq bm$ where b is the maximum weight of all the points. In fact, M can be computed when we run MaxC-SRP algorithm. Then, in the case of $m = o\left(\varepsilon^{-2}\log n\right)$, the griding method of Mark De Berg *et al.* [8] can achieve a time complexity of $o\left(n\varepsilon^{-2}\log n\right)$. It suffices to note that $M^* \leq M \leq bm$ holds, such that each circular region equal in size to the input circle has at most $\frac{b}{a}m$ points. And each grid in [8] can be covered by a constant number of circles, so the number of points in each grid is $o\left(\varepsilon^{-2}\log n\right)$ and the running time of their method is $o\left(n\varepsilon^{-2}\log n\right)$.

In summary, our algorithm finishes running in $o\left(n\varepsilon^{-2}\log n\right)$ time when $l = O(k) = O\left(\varepsilon^{-1}\right)$ time and returns a $(1-\varepsilon)$ approximation for at least $1 - n^{-c}$ events for all possible values of m.

6 Conclusion

We propose a novel approximation algorithm for max-covering circle problem. We first construct the circumscribed SRP of a given circle and disclose their area relationship. The area difference is determined by the number of edges in the circumscribed SRP. Then, we convert max-covering SRP problem to SRP intersection problem, which can be efficiently solved with simple partition and modification based on the existing method. Finally, the optimal solution returned from max-covering SRP problem can be used to produce an approximate answer to max-covering circle problem. We prove that for most of the inputs, our algorithm can give a $(1-\varepsilon)$ approximation to the optimal solution, which only needs $O\left(n\varepsilon^{-1}\log n + n\varepsilon^{-1}\log\left(\frac{1}{\varepsilon}\right)\right)$ time for unit points and $o\left(n\varepsilon^{-2}\log n\right)$ time for weighted points.

Acknowledgements. This work was supported by NSFC grant (Nos. 62102119, U19A2059, U22A2025 and 62232005).

References

1. Aggarwal, A., Imai, H., Katoh, N., Suri, S.: Finding k points with minimum diameter and related problems. J. Algorithms **12**(1), 38–56 (1991)
2. Aronov, B., Har-Peled, S.: On approximating the depth and related problems. SIAM J. Comput. **38**(3), 899–921 (2008)
3. Chan, T.M., Har-Peled, S.: Smallest k-enclosing rectangle revisited. Discrete Comput. Geomet. **66**(2), 769–791 (2021)
4. Chazelle, B.M., Lee, D.T.: On a circle placement problem. Computing **36**(1), 1–16 (1986)
5. Choi, D., Chung, C., Tao, Y.: A scalable algorithm for maximizing range sum in spatial databases. Proc. VLDB Endow **5**(11), 1088–1099 (2012)
6. Datta, A., Lenhof, H.P., Schwarz, C., Smid, M.: Static and dynamic algorithms for k-point clustering problems. J. Algorithms **19**(3), 474–503 (1995)
7. De Berg, M., Cabello, S., Cheong, O., Eppstein, D., Knauer, C.: Covering many points with a small-area box. arXiv preprint arXiv:1612.02149 (2016)
8. De Berg, M., Cabello, S., Har-Peled, S.: Covering many or few points with unit disks. Theory Comput. Syst. **45**(3) (2009)
9. Efrat, A., Sharir, M., Ziv, A.: Computing the smallest k-enclosing circle and related problems. Comput. Geom. **4**(3), 119–136 (1994)
10. Eppstein, D., Erickson, J.: Iterated nearest neighbors and finding minimal polytopes. Discrete Comput. Geomet. **11**(3), 321–350 (1994)
11. Har-Peled, S., Mazumdar, S.: Fast algorithms for computing the smallest k-enclosing circle. Algorithmica **41**(3), 147–157 (2005)
12. Hemmer, M., Kleinbort, M., Halperin, D.: Improved implementation of point location in general two-dimensional subdivisions. In: Epstein, L., Ferragina, P. (eds.) ESA 2012. LNCS, vol. 7501, pp. 611–623. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33090-2_53
13. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**, 13–30 (1963)
14. Imai, H., Asano, T.: Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. J. Algorithms **4**(4), 310–323 (1983)
15. Kaplan, H., Roy, S., Sharir, M.: Finding axis-parallel rectangles of fixed perimeter or area containing the largest number of points. Comput. Geom. **81**, 1–11 (2019)
16. Mahapatra, P.R.S., Karmakar, A., Das, S., Goswami, P.P.: k-enclosing axis-parallel square. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2011. LNCS, vol. 6784, pp. 84–93. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21931-3_7
17. Matoušek, J.: On enclosing k points by a circle. Inf. Process. Lett. **53**(4), 217–221 (1995)
18. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press (1995)
19. Nandy, S.C., Bhattacharya, B.B.: A unified algorithm for finding maximum and minimum object enclosing rectangles and cuboids. Comput. Math. Appl. **29**(8), 45–61 (1995)

20. Preparata, F.P., Shamos, M.I.: Computational Geometry. Springer, New York (1985). https://doi.org/10.1007/978-1-4612-1098-6
21. Tao, Y., Hu, X., Choi, D., Chung, C.: Approximate Maxrs in spatial databases. Proc. VLDB Endow **6**(13), 1546–1557 (2013)
22. Zhang, K., Gao, H., Han, X., Chen, J., Li, J.: Maximizing range sum in trajectory data. In: IEEE International Conference on Data Engineering, pp. 755–766 (2022)