Dispersing Facilities on Planar Segment and Circle Amidst Repulsion

Vishwanath R. Singireddy  Manjanna Basappa*

Department of Computer Science & Information Systems,
BITS Pilani, Hyderabad Campus, Telangana 500078, India
{p20190420,manjanna}@hyderabad.bits-pilani.ac.in
May 10, 2022

Abstract

In this paper we consider the problem of locating $k$ obnoxious facilities (congruent disks of maximum radius) amidst $n$ demand points (existing repulsive facility sites) ordered from left to right in the plane so that none of the existing facility sites are affected (no demand point falls in the interior of the disks). We study this problem in two restricted settings: (i) the obnoxious facilities are constrained to be centered on along a predetermined horizontal line segment $pq$, and (ii) the obnoxious facilities are constrained to lie on the boundary arc of a predetermined disk $C$. An $(1 - \epsilon)$-approximation algorithm was given recently to solve the constrained problem in (i) in time $O((n + k) \log \frac{||pq||}{\epsilon k})$, where $\epsilon > 0$ [6]. Here, for the problem in (i), we first propose an exact polynomial-time algorithm based on a binary search on all candidate radii computed explicitly. This algorithm runs in $O((nk)^2 \log (nk) + (n + k) \log (nk))$ time. We then show that using the parametric search technique of Megiddo [1]; we can solve the problem exactly in $O((n + k)^2)$ time, which is faster than the latter. Continuing further, using the improved parametric technique we give an $O(n \log^2 n)$-time algorithm for $k = 2$. We finally show that the above $(1 - \epsilon)$-approximation algorithm of [6] can be easily adapted to solve the circular constrained problem of (ii) with an extra multiplicative factor of $n$ in the running time.

*Corresponding author
1 Introduction

The obnoxious facility location is a well-known topic in the operations research community. This paper addresses a variant of the problem, namely, the continuous obnoxious facility location on a line segment (COFL) problem, motivated by the following application. We wish to find locations for establishing \( k \) obnoxious or undesirable facilities (such as garbage dump yards, industries generating pollution, etc.) along a straight highway road such that the pairwise distance between these new facilities and the distance between each of the new facilities and other existing non-obnoxious facilities (such as hospitals, schools, etc.) are maximized. The formal definition of the problem is below:

The COFL problem: Given a horizontal line segment \( \overline{pq} \) and an ordered set \( P \) of \( n \) points lying above \( \overline{pq} \) in the Euclidean plane, we aim to locate points on \( \overline{pq} \) for centering \( k \) non-overlapping congruent disks of maximum radius \( r_{\text{max}} \) such that none of the points in \( P \) lie interior to any of these disks.

Recently, it has been shown in [6] that we can solve the decision version of this problem (in which we are also given a radius \( L \) as input along with \( \overline{pq} \) and the set \( P \) of \( n \) points) in time \( O(n + k) \) if the points in \( P \) are given in order. Then, using this decision algorithm as a subroutine, an \((1 - \epsilon)\)-approximation algorithm (FPTAS) has been given to solve the COFL problem in time \( O((n + k) \log \frac{||\overline{pq}||}{\epsilon}) \), where \( \epsilon > 0 \) and \( ||\overline{pq}|| \) is the length of the segment \( \overline{pq} \) [6].

In this work, we show that we can, in fact, solve the COFL problem exactly in polynomial time. Using the linear-time decision algorithm of [6] again as a subroutine, we present two polynomial-time exact algorithms based on two different approaches: (i) the algorithm is based on doing a binary search on all candidate radii \( L \) computed explicitly and runs in \( O((nk)^2 \log (nk) + (n + k) \log (nk)) \) time, and (ii) the algorithm is based on Megiddo’s parametric search [1], and runs in \( O((n + k)^2) \) time. We then discuss an \( O(n \log^2 n) \)-time algorithm to solve the COFL problem for \( k = 2 \), which is faster than the previous two algorithms. Finally, we show how to solve the circular COFL problem using the algorithm of [6].

2 Related Work

Several variants of obnoxious facility location problems have been investigated by the operations research and computational geometry communities. A recent review of many variants of obnoxious facility location problems and specialized heuristics to solve them may be found in [11]. Kartz et al. [3] studied two variants of \( k \)-obnoxious facility location problems (i) max-min \( k \)-facility location, in which the distance between any two facilities is at least some fixed value and the minimum distance between a facility and a demand point is to be maximized, and (ii) min-sum one-facility location problem in which the sum of weights of demand points lying within a given distance to the facility is to be minimized. They solved the first problem in \( O(n \log^2 n) \) time for \( k = 2 \) or 3 and for any \( k \geq 4 \) their algorithm runs in \( O(n(k-2) \log^2 n) \) time for rectilinear case. In [3], they also solved the Euclidean case where their algorithm runs in \( O(n \log n) \) time for \( k = 2 \). For the second problem, their algorithm runs in \( O(n \log n) \) time (rectilinear case) and \( O(n^2) \) time (Euclidean case). Qin et al. [12] studied a variant of \( k \)-obnoxious facility location problem in which the facilities are restricted to lie within a given convex polygonal domain. They proposed a 2-factor approximation algorithm based on a Voronoi diagram for this problem, and its running time is \( O(kN \log N) \), where \( N = n + m + k \), \( n \) denotes the number of demand points, \( m \) denotes the number of vertices of the polygonal domain and \( k \) is the number of obnoxious facilities to be placed. Díaz-Báñez et al. [9] modelled obnoxious facility as empty circular annulus whose width is maximum. This refers to a max-min facility location problem such that the
facility is a circular ring of maximum width, where the width is the absolute difference between the radii of the two concentric circles. They solved the problem in $O(n^3 \log n)$ time, and if the inner circle contains a fixed number of points then the problem can be solved in $O(n \log n)$ time. Maximum width empty annulus problems for axis parallel square and rectangles can be solved in $O(n^3)$ and $O(n^2 \log n)$ time respectively [10]. Abravaya and Segal [2] studied the problem of locating maximum cardinality set of obnoxious facilities within a bounded rectangle in the Euclidean plane such that their pairwise distance is at least a given threshold. They proposed a 2-approximation algorithm and also a PTAS based on shifting strategy [5]. Agarwal et al. [8] showed the application of Megiddo’s parametric search [1] method to solve several geometric optimization problems in the Euclidean plane such as the big stick problem, the minimum width annulus problem, and the problem of finding largest mutual visible spheres.

## 3 Polynomial time exact algorithms

Given a horizontal segment $pq$, without loss of generality, we can assume that all the points in $P$ lie in the upper half-plane defined by a line through $pq$. After fixing the segment $pq$, we define the following decision version of the COFL problem as follows:

\[ \text{DCOFL}(P, k, L): \text{Given an ordered set } P \text{ of } n \text{ points lying above a line through } pq, \text{ a real number } L \text{ and an integer } k, \text{ can we pack } k \text{ non-overlapping disks of radius } L \text{ centered on } pq \text{ such that none of these disks contains a point of } P \text{ in their interior?} \]

Let $I = \{[x_{i,1}, x_{i,2}] | p_i \in P\}$ be the set of intervals on $pq$ ordered from $p$ to $q$, where every point in each interval $[x_{i,1}, x_{i,2}]$ is at distance at most $L$ from $p_i \in P$ (see Fig. 1). The decision algorithm returns YES if we can place $k$ pairwise interior-disjoint disks of radius $L$, centered on $pq$ but outside the intervals in $I$. Otherwise the algorithm returns NO. Let $I^c = \{[x_{i,2}, x_{j,1}] | p_i, p_j \in P, i < j\} \cup \{[x_{0,2}, x_{i,1}], [x_{n,2}, x_{n+1,1}]\}$ be the set of complemented intervals of $I$, where every point in each interval $[x_{i,2}, x_{j,1}] \in I^c$ is at distance at least $L$ from every point in $P$ and $x_{0,2} = x(p)$, $x_{n+1,1} = x(q)$. We use the algorithm of [6] for solving the decision version of the problem. For completeness and for the purpose of describing exact algorithms here, we have reproduced that decision algorithm again here (see Algorithm 1). Let $R$ be the the Minkowski sum of $pq$ and a disk of radius $L$, i.e., the union of $pq$ and the region swept by a disk of radius $L$ when its center moves along $pq$.

![Figure 1: A point $p_i$ is having two center points of $pq$ which are at unknown distance $L$](image)

### 3.1 Algorithm based on binary search on all candidate radii values

The approach that we follow here is that we first compute the set $L_{\ell_2}$ (resp. $L_{\ell_\infty}$) of all candidate radii $L$, based on some observations on optimal packing, where $L_{\ell_\infty}$ and $L_{\ell_2}$ are for rectilinear space
Algorithm 1: Greedy_LPacking\( (P, k, L) \)

1. Compute \( I \)
2. Compute \( I^c \) where the elements are ordered from \( p \) to \( q \), and let \( m = |I^c| \)
3. \( j \leftarrow 0 \)

4. for each \( i \leftarrow 1 \) to \( m \) do
   5. \( \gamma \leftarrow \frac{\text{length of } i\text{th interval of } I^c}{2L} \)
   6. if \( (j + \gamma + 1) \leq k \) then
      7. On the \( i\)th interval of \( I^c \), pack \( \gamma + 1 \) disks with radius \( L \)
      8. update \( j \leftarrow (j + \gamma + 1) \)
   9. else
      10. On the \( i\)th interval of \( I^c \), pack \( k - j \) disks with radius \( L \)
      11. update \( j \leftarrow k \)
   12. end

13. end
14. if the number of disks packed on intervals of \( I^c \) is \( k \) then
15. return YES and the \( k \) disks packed \( \{d_1, d_2, \ldots, d_k\} \)
16. end
17. else
18. return NO
19. end

and Euclidean space respectively. We sort \( \mathcal{L} \) (resp. \( \mathcal{L}_{\ell_\infty} \)), then invoke Algorithm 1 by setting \( L \) to some element in \( \mathcal{L}_{\ell_2} \) (resp. \( \mathcal{L}_{\ell_\infty} \)). Ultimately, we find \( r_{\text{max}} \) by doing binary search on the ordered elements in \( \mathcal{L}_{\ell_2} \) (resp. \( \mathcal{L}_{\ell_\infty} \)).

To get familiarity with this approach, we first discuss this for the rectilinear version of the COFL problem (in \( \ell_1 \) or \( \ell_\infty \) metric). We then apply the approach to solve the (Euclidean) COFL problem (i.e., in \( \ell_2 \) metric). We now define the rectilinear version of the problem as follows:

Problem 1 Given a horizontal line segment \( \overline{pq} \), an ordered set \( P \) of \( n \) demand points, and an integer \( k \), pack \( k \) maximum-size axis-aligned squares centered on \( \overline{pq} \) such that no point of \( P \) lies in the interior of any of these squares, where the size of a square is defined to be half of its side length.

The optimal size of the square is determined by searching all possible candidate sizes as discussed below:

Consider an instance of the COFL problem, i.e., a line segment \( \overline{pq} \), an ordered set \( P \) of \( n \) points, and an integer \( k \). As in the previous section, without loss of generality, let all the points of \( P \) be lying above the line through the segment \( \overline{pq} \). Now, we need to place \( k \) non-overlapping squares of maximum size with centers lying on \( \overline{pq} \) without violating the constraint that no point of \( P \) lies inside any of these squares. The maximum size of the square depends on some ”influencing points” in \( P \) for the given position of the fixed horizontal line segment \( \overline{pq} \).

Definition 3.1 The influencing points \( P^{\text{inf}} \subseteq P \) are those points which satisfy the following criteria: given any point \( p_i \in P^{\text{inf}} \) (possibly along with another point \( p_j \in P^{\text{inf}} \)) we can place \( k \) pairwise disjoint congruent squares centered on \( \overline{pq} \) such that (i) \( p_i \) (and \( p_j \)) lies on the boundary of some of these squares, and (ii) if \( p_i \) (or \( p_j \) or both) is removed from \( P \), then the size of these squares may be increased while their centers are perturbed on \( \overline{pq} \) to keep their pairwise disjointness intact, and (iii) no other points of \( P \) lie in the interior of the squares all the time.
Lemma 3.1 *If the position of \( pq \) is fixed and an ordered set \( P \) of \( n \) points are not lying strictly below the line through \( pq \), then for any optimal solution to the COFL problem in \( \ell_\infty \) metric, there are at most two points in \( P \) that will determine the optimal radius \( r_{\text{max}} \) for the disks (axis-aligned squares with size \( r_{\text{max}} \)) in the packing, and also \( |L_{\ell_\infty}| = O(n^2k^2) \).*

**Proof:** The proof follows from the following cases:

**Case 0:** If \( P^{inf} \) is empty, then no points of \( P \) are at a distance of \( \frac{||pq||}{2(k-1)} \) from the line segment \( pq \).

This means that none of the points of \( P \) lie on the boundary of the \( k \) squares in the optimal packing. Therefore, we have that the maximum size of the square is \( \frac{||pq||}{2(k-1)} \).

If \( P^{inf} \) is non-empty, then there will be at least one point on the boundary of any of the \( k \) squares in the optimal packing.

First, we will discuss how to find the set \( P^{inf} \subseteq P \) of all the influencing points. Let \( s_1, s_2, \ldots, s_k \) be the squares in the optimal solution. Let the point \( p' \in P \) be the leftmost point among all the points of \( P \) that lie on the boundaries of the squares \( s_j \) (where \( j = 1, 2, \ldots, k \)) in the optimal packing (in case of ties, we may consider the one with largest \( y \)-coordinate). We will see that this point \( p' \in P^{inf} \) in many cases, and sometimes \( p' \notin P^{inf} \) (this means that \( p' \) is not involved in determining \( r_{\text{max}} \) and it may be that some other two points \( p_i, p_q \in P^{inf} \) on the right of \( p' \) determines the radius \( r_{\text{max}} \)). Now, assume that \( p' \) participates in determining \( r_{\text{max}} \), let \( p_i = p' \). Then, \( p_i \) may lie on the right edge, or the left edge or the top edge boundaries of some square \( s_j \). Based on these, we have the following cases.

**Case 1:** Assume that \( p_i \) lies on the right edge of \( s_j \) such that its position determines the size of the square (see Fig. 2). For each \( j = 1, 2, \ldots, k \) in the optimal packing, \( p_i \) can lie on the right edge of \( s_j \). For each such point \( p_i \in P^{inf} \) the candidate radius \( r_{\text{can}} = \frac{x(p)-x(p_i)}{2j-1} \) can be calculated and stored in \( L_{\ell_\infty} \). Hence, the number of candidate radii in this case is \( O(nk) \).

![Figure 2: Illustration of case 1](image)

**Case 2:** If \( p_i \) lies on the top edge of \( s_j \), then there are three subcases:

1. There is another point \( p_j' \in P \) that lies on the right edge of \( s_j \) (see Fig. 3) or on the right edge of \( s_j' \), where \( j' \geq j + 1 \) (see Fig. 4). Then if this pair \( p_i, p_j' \) determines the maximum size of the square then the candidate radius \( r_{\text{can}} = y(p_i) - y(q) \in L_{\ell_\infty} \) for every pair \( p_i, p_j' \in P^{inf} \). Note that this can be done exhaustively for every pair of points in \( P \) and for every possible value of \( j \) by checking that no other point of \( P \) lies inside any of these \( j \) squares.

2. If there is no point lying on the right edge of \( s_j' \) for any \( j' = j + 1, j + 2, \ldots, k \) (see Fig. 5) then this case will be same as Case 0 if \( \bigcup_{j=1}^{k} s_j \cap [p,q] = [p,q] \). Otherwise, \( r_{\text{can}} = y(p_i) - y(q) \), where \( i \in \{1, 2, \ldots, n\} \) is the index of the point in \( P \) with the smallest \( y \)-coordinate such that \( r_{\text{can}}/2 + x(p) \leq x(p_i) \leq x(q) + r_{\text{can}}/2 \).
Case 3: Now assume that the point $p_i$ lies on the left edge of $s_j$ for $j = 1, 2, \ldots, k$ (see Fig. 6). Observe that if $p_i$ lies on the left edge of $s_j$ where $j \geq 2$, then without loss of generality we can assume that all the first $j - 1$ squares are compactly packed together and the right edge of $s_{j-1}$ coincides with the left edge of $s_j$. Thus, this is same as one of the above cases as the point $p_i$ lies on the right edge of $s_{j-1}$. Hence, in the remaining cases we can assume that $p_i$ always lies on the left edge of $s_1$.

(i) There is no point $p_{i'}$ lying on either the top edge or right edge of any of the squares $s_{j'}$ for $j' = 2, 3, \ldots, k$. This is essentially the same case as Case 0 except that the segment $\overline{pq}$
is shrunken to be of length \(|pq| - (x(p_i) - x(p) + r_{\text{CAN}})\) and \(s_1\) is centered at the left end point of this shrunken segment. Then, \(r_{\text{CAN}} = \frac{|pq| - x(p_i) + x(p)}{2k - 1} \in \mathcal{L}_{\ell_\infty}^\ast\).

(ii) If \(p_i\) lies on the left edge of the square \(s_1\) then based on the position of the point \(p_{i'}\) \((i' > i)\) that lies on the square \(s_1\) or the square \(s_{j'}\) \((j' > j = 1)\) on the right side of \(p_i\) we have the following situations.

(a) When \(p_{i'}\) lies on the right edge of \(s_1\). Then, \(r_{\text{CAN}} = \frac{x(p_{i'}) - x(p_i)}{2} \in \mathcal{L}_{\ell_\infty}^\ast\).

(b) When \(p_{i'}\) lies on the right edge of the square \(s_{j'}\) (see Fig. 7). Then, \(r_{\text{CAN}} = \frac{x(p_{i'}) - x(p_i)}{2j - 1} \in \mathcal{L}_{\ell_\infty}^\ast, j' = 2, 3, \ldots, k\).

(c) When \(p_{i'}\) lies on the top edge of the square \(s_j\) (see Fig. 8). Then, \(r_{\text{CAN}} = y(p_{i'}) - y(p) \in \mathcal{L}_{\ell_\infty}^\ast\). It is similar to Case 2(i).

Case 4: Now assume any two distinct points \(p_i, p_{i'} \in P\) being located such that they lie on the adjacent sides of the same square \(s_j\) \((j = j')\) (see Fig. 9). Then, if we consider the size of such a square for each pair of points in \(P\), there are at most \(O(n^2)\) candidate radii \(r_{\text{CAN}}\) in \(\mathcal{L}_{\ell_\infty}^\ast\).

Observe that all the remaining cases can be reduced to one of the above cases. Since there are only a constant number \(c\) of possible positions of the two squares \(s_j, s_{j'}\) in the optimal packing for a fixed pair \(p_i, p_{i'} \in P^{\text{inf}}\), the number of candidate radii \(r_{\text{CAN}}\), i.e., \(|\mathcal{L}_{\ell_\infty}^\ast| = \binom{n}{2} \sum_{j=1}^{k} \sum_{j'=j}^{k} c = O(n^2k^2)\).

Hence the cardinality of \(\mathcal{L}_{\ell_\infty}^\ast\) is \(O((nk)^2)\), we can compute \(\mathcal{L}_{\ell_\infty}^\ast\) in \(O((nk)^2)\) worst case times.

\(\square\)

Now, sort the elements in \(\mathcal{L}_{\ell_\infty}^\ast\). By doing binary search on the sorted \(\mathcal{L}_{\ell_\infty}^\ast\), solve the \(\text{DCOFL}(P, k, L)\) problem repeatedly (invoke Algorithm 1) by setting \(L\) to the median \(r_{\text{CAN}} \in \mathcal{L}_{\ell_\infty}^\ast\) each time. Since the set of candidate radii computed is exhaustive, the above search guarantees to find \(r_{\text{CAN}} = r_{\text{MAX}}\). Hence we have the following theorem.
Theorem 3.2 For a given line segment $\overline{pq}$ and an ordered set $P$ of $n$ points in the rectilinear plane, we can solve the COFL problem optimally in $O((nk)^2 \log (nk) + (n + k) \log (nk))$ time.

Remark 3.1 The COFL problem for $k = 2$, under the $\ell_\infty$ norm, can be solved in $O(n \log n)$ by employing the optimization technique of Frederickson and Johnson [4] instead of doubling search and bisection methods, because the problem will be a special case of the one in [3].

The Euclidean COFL problem:

The approach here is that we first compute the set $\mathcal{L}$ of all candidate radii $L$, based on some observations of possible positions of all $k$ disks in an optimal packing. We sort all the elements in $\mathcal{L}$. Then, by doing a binary search on the sorted $\mathcal{L}$, we invoke the decision algorithm (Algorithm 1) each time we peek an element at the middle index by setting $\mathcal{L}$ to this element as the candidate radius. We continue this search until we find the radius $L^*$ (maximum radius) such that for all $L \in \mathcal{L}$ with $L > L^*$ the decision algorithms returns NO. Hence, $L^* = r_{\text{max}}$.

Consider an instance of the COFL problem, i.e., a line segment $\overline{pq}$, an ordered set $P$ of $n$ points, and an integer $k$. As already assumed, without loss of generality, let all the points of $P$ be lying above the line through the segment $\overline{pq}$. Now, we need to place $k$ non-overlapping congruent disks of maximum radius with centers lying on $\overline{pq}$ without violating the constraint that no point of $P$ lies inside any of these disks. The maximum radius of the disks depends on some ”influencing points” in $P$ for the given position of the fixed horizontal line segment $\overline{pq}$.

Definition 3.2 The influencing points $P^{\text{inf}} \subseteq P$ are those points which satisfy the following criteria: given any point $p_i \in P^{\text{inf}}$ (possibly along with another point $p_i' \in P^{\text{inf}}$) we can place $k$ pairwise disjoint congruent disks centered on $\overline{pq}$ such that (i) $p_i$ (and $p_i'$) lies on the boundary of some of these disks, and (ii) if $p_i$ (or $p_i'$ or both) is removed from $P$, then the radius of these disks may be increased while their centers are perturbed on $\overline{pq}$ to keep their pairwise disjointness intact, and (iii) none of the other points of $P$ lie in the interior of any of these disks before and after the radius increases.

Lemma 3.3 Given that the position of $\overline{pq}$ is fixed, for any optimal solution to the COFL problem in $\ell_2$ metric, there are at most two points in $P^{\text{inf}} \subseteq P$ that will determine the optimal radius $r_{\text{max}}$ of the disks in the corresponding packing, and also $|\mathcal{L}| = O(n^2k^2)$.

Proof: The proof is based on case analysis. First, let $r_{\text{CAN}} \in \mathcal{L}$ be a candidate radius.

Case 1: The set $P^{\text{inf}}$ is empty and then $r_{\text{CAN}} = \frac{|\overline{pq}|}{k}$. If this case is not satisfied, then there will be at least one point lying on the boundary of some disk in the optimal packing which will influence on the value of $r_{\text{max}}$. Among these points in $P$ lying on the
boundaries of the optimal disks, let \( p_i \) be the left most, i.e., the point with the smallest \( x \)-coordinate. Let \( d_1, d_2, \ldots, d_k \) be the disks ordered from left to right in the optimal packing. Now we will examine all possible positions of these disks for every subset of at most two influencing points in \( P^{\text{inf}} \). We will also show how to compute the corresponding candidate radii \( r_{\text{CAN}} \in L \). To this end, consider a disk \( d \) centered on \( pq \). Divide the part of the boundary arc \( \partial d \) of \( d \) lying above \( pq \) into left and right arc segments by a vertical line through the center of \( d \) (see Figure 10). We then show that at most two points \( p_i, p'_v \in P^{\text{inf}} \subseteq P \) determine the radius \( r_{\text{max}} \) of the disks in an optimal packing.

![Figure 10: Left arc segment and right arc segment of disks \( d_j \) and \( d_{j'} \)](image)

Let \( d_j \) and \( d_{j'} \) (possibly \( j = j' \)) be the two disks, on whose boundary arc segments \( L^j_{\text{arc}} \) or \( R^j_{\text{arc}} \), and \( L^j_{\text{arc}} \) or \( R^j_{\text{arc}} \) the two points \( p_i \) and \( p'_v \) lie. Based on all possible positions of \( d_j \) and \( d_{j'} \) for any pair \( p_i \) and \( p'_v \), we have the following cases.

**Case 2:** Assume that \( p_i \) lies on \( R^i_{\text{arc}} \) of \( d_j \) such that its position determines the radius of the disk (there is no point \( p'_v \) that influences the radius) (see Figure 11). For any \( j = 1, 2, \ldots, k \) in the optimal packing, \( p_i \) can lie on \( R^j_{\text{arc}} \) of \( d_j \). For each such point \( p_i \in P^{\text{inf}} \) the candidate radius \( r_{\text{CAN}} \) can be calculated and stored in \( L \). Hence the number of candidate radii in this case is \( O(nk) \). The candidate radius \( r_{\text{CAN}} \) with respect to every point can be calculated using the below equation:

\[
(j - 1)2r_{\text{CAN}} - r_{\text{CAN}} + 2r_{\text{CAN}} = x(p_i) + \left( r_{\text{CAN}} - \sqrt{r_{\text{CAN}}^2 - (y(p_i) - y(q))^2} \right) - x(p)
\]

\[
((j - 1)2 - 1 + 2)r_{\text{CAN}} - r_{\text{CAN}} = x(p_i) - \sqrt{r_{\text{CAN}}^2 - (y(p_i) - y(q))^2} - x(p)
\]

\[
2(j - 1)r_{\text{CAN}} = x(p_i) - \sqrt{r_{\text{CAN}}^2 - (y(p_i) - y(q))^2} - x(p)
\]

The candidate radius \( r_{\text{CAN}} \in L_{d_j} \) is calculated from the above equation as we know the value of every term of the equation except \( r_{\text{CAN}} \). The mirror case of this where the point \( p'_v \) is the right most point and lies on \( L^j_{\text{arc}} \) of \( d_{j'} \) can be handled similarly. In this case, \( r_{\text{CAN}} \) can be calculated using the below equation.

\[
(2(k - j') + 1)r_{\text{CAN}} = ||pq|| - x(p'_v) + x(p) + r_{\text{CAN}} - \sqrt{r_{\text{CAN}}^2 - (y(p'_v) - y(q))^2}
\]

**Case 3:** Let both the points \( p_i, p'_v \in P \) be determining the radius of the disks in the optimal packing. Then the point \( p_i \) can lie on \( L^j_{\text{arc}} \) or \( R^j_{\text{arc}} \) of \( d_j \), where \( j = 1, 2, \ldots, k \) (see Fig 12). Similarly, the point \( p'_v \) can also lie on \( L^j_{\text{arc}} \) or \( R^j_{\text{arc}} \) of \( d_{j'} \), where \( j' = j, j + 1, j + 2, \ldots, k \). Here, the disks centered between \( p_i \) and \( p'_v \) are compactly packed and determining the optimal radius along with the positions of \( p_i \) and \( p'_v \). It is easy to observe that there are a constant number \( c \) of possible positions for the two disks \( d_j \) and \( d_{j'} \) in an optimal solution such that the two points \( p_i, p'_v \) lie on their boundaries and does not let their radius \( r_{\text{CAN}} \) to increase by repositioning all the \( k \) disks. Therefore, the number of all candidate radii \( r_{\text{CAN}} \) is \( \left( \begin{array}{c} n \end{array} \right) \sum_{j=1}^{k} \sum_{j'=j}^{k} c = k^2c \left( \begin{array}{c} n \end{array} \right) = O(cn^2k^2) \). The candidate radii values \( r_{\text{CAN}} \in L \) can be computed
from the following equation:

\[ x(p_i') - x(p_i) = 2(j' - j)r_{\text{can}} \pm (1) \sqrt{r_{\text{can}}^2 - (y(p_i) - y(q))^2} \pm (2) \sqrt{r_{\text{can}}^2 - (y(p_i') - y(q))^2} \]

where in \pm (1), + indicate \( p_i \) lies on \( L_{\text{arc}}^{j} \) of \( d_j \) and \( - \) indicate \( p_i \) lies on \( R_{\text{arc}}^{j} \) of \( d_j \), similarly, in \pm (2), + indicate \( p_{i'} \) lies on \( R_{\text{arc}}^{j_1} \) of \( d_{j'} \) and \( - \) indicate \( p_{i'} \) lies on \( L_{\text{arc}}^{j_2} \) of \( d_{j'} \).

**Case 4:** Assume the point \( p_i \in P \) lies on \( R_{\text{arc}}^{j} \) of some disk \( d_j \) and there are also other (optional) points \( p_{i'}, p_{i''} \in P \) lying on boundary arcs of the disks on the right of \( d_j \). Based on all possible positions of \( p_i, p_{i'}, p_{i''} \) we can observe that at most two of these points will determine \( r_{\text{can}} \) and each of these cases corresponds to one of the above cases.

(i) When \( p_{i'} \in P \) lies on \( L_{\text{arc}}^{j+1} \) of the disk \( d_{j+1} \) and there is empty space between \( d_j \) and \( d_{j+1} \) (see Figure 13). This corresponds to case 2 above as either \( j-1 \) disks on the left of \( p_i \) or \( k-j-1 \) disks on the right of \( p_{i'} \) are compactly packed with their radius increased to the maximum possible value.

(ii) When \( p_{i'} \in P \) lies on \( L_{\text{arc}}^{j''} \) of the disk \( d_{j''} \) and \( p_{i''} \in P \) lies on the disk \( d_{j'''} \), where \( j' - j > 1 \) and \( j'' - j' > 1 \) (see Fig. 14). Then, the disks in the optimal packing can be partitioned into four subsequences of consecutive disks \( d_1, d_2, \ldots, d_j, d_{j+1}, \ldots, d_{j'}, d_{j''}, \ldots, d_{j'''} \). In at least one of these, the disks must be compactly packed with their radius increased to the maximum possible value, otherwise it would contradict that we have the optimal packing. Hence, one of the above cases applies and at most two points among \( p_i, p_{i'}, p_{i''} \) determine the value of \( r_{\text{can}} \).

Thus the lemma follows.

**Theorem 3.4** For a given line segment \( \overline{pq} \) and an ordered set \( P \) of \( n \) points in the Euclidean plane, we can solve the COFL problem optimally in \( O((nk)^2 \log (nk) + (n + k) \log (nk)) \) time.
Proof: The constant $c = 10$ because of the following: (i) no point lies on the boundary of any disk in optimal packing, (ii) $p_i$ lies on $L^i_{arc}$ or $R^i_{arc},$ (iii) both $p_i, p'_i$ lie on $L_i^j_{arc}$ or $R_i^j_{arc},$ (iv) $p_i$ lies on $L_i^j_{arc}$ and $p'_i$ lies on $R_i^j_{arc},$ (v) $p_i$ lies on $L_i^j_{arc}$ and $p'_i$ lies on $R_i^j_{arc}$ or $p_i$ lies on $R_i^j_{arc}$ and $p'_i$ lies on $L_i^j_{arc}$ or $p_i$ lies on $L_i^j_{arc}$ and $p'_i$ lies on $R_i^j_{arc}$ or $p_i$ lies on $L_i^j_{arc}$ and $p'_i$ lies on $R_i^j_{arc}.$ Observe that (i) corresponds to case-1 and contributes 1, (ii) corresponds to case-2 and contributes 2, (iii) contributes 1, (iv) contributes 4, hence $c = 1 + 2 + 2 + 1 + 4 = 10.$ Therefore, $|\mathcal{L}| = O((nk)^2)$ and we find $r_{max} \in \mathcal{L}$ by a binary search on sorted $\mathcal{L}.$ Sorting $\mathcal{L}$ takes $O((nk)^2 \log (nk))$ and then invoking the decision algorithm $O(\log(nk))$ times results in overall time of $O((nk)^2 \log(nk) + (n + k) \log(nk)).$

\[\square\]

3.2 Algorithm based on parametric search

The FPTAS given in [6] for the COFL problem is weakly polynomial because the number of calls to the decision algorithm is proportional to the number of bits of precision of accuracy for the desired approximation. Here, we discuss an algorithm based on (slower version of) Megiddo’s parametric search [1] so that its running time is independent of any numerical precision parameter ($\frac{1}{2}$). As we know, in a solution based on parametric search, there is a test algorithm, and a decision algorithm wherein the test algorithm is typically a step-by-step simulation of the decision algorithm. We now will describe how to simulate the steps of Algorithm 1 at the unknown maximum $L^* (= r_{max}).$

Consider a point $p_i \in \mathcal{P}$ which is having two center points $c_{i,1}$ and $c_{i,2}$ on the segment $pq$ (see Fig. 1). Let the coordinates of these points be $p_i = (x(p_i), y(p_i)),$ $c_{i,1} = (x_{i,1}, y_{i,1})$ and $c_{i,2} = (x_{i,2}, y_{i,2}).$ Clearly, these points and $L$ satisfy the equations:

\begin{align*}
L^2 &= (x(p_i) - x_{i,1})^2 + (y(p_i) - y_{i,1})^2 \implies L^2 = (x(p_i) - x_{i,1})^2 + (y(p_i) - y(q))^2 \quad (1) \\
L^2 &= (x(p_i) - x_{i,2})^2 + (y(p_i) - y_{i,2})^2 \implies L^2 = (x(p_i) - x_{i,2})^2 + (y(p_i) - y(q))^2 \quad (2)
\end{align*}

where $y_{i,1} = y_{i,2} = y(p) = y(q)$ as both the points $c_{i,1}$ and $c_{i,2}$ are located on $pq.$
Since we know the coordinate values \( y(q), \ x(p_i) \) and \( y(p_i) \), the values of \( x_{i,1}, \ x_{i,2} \) are given as follows:

\[
(x(p_i) - x_{i,1})^2 = L^2 - (y(p_i) - y(q))^2 \implies x_{i,1} = x(p_i) - (L^2 - (y(p_i) - y(q))^2)^{\frac{1}{2}},
\]

\[
(x(p_i) - x_{i,2})^2 = L^2 - (y(p_i) - y(q))^2 \implies x_{i,2} = x(p_i) - (L^2 - (y(p_i) - y(q))^2)^{\frac{1}{2}}
\]

respectively, if \( y(p_i) \leq y(q) + L \)

Now, consider the end points of the \( i \)th complemented interval as \([r_{i-1}l_i]\) (at line 5 of Algorithm 1). Let the coordinates \( r_{i-1} = (x_{i-1,2}, y_{i-1,2}) \), and \( l_i = (x_{i,1}, y_{i,1}) \), then

\[
\gamma = \frac{|r_{i-1}l_i|}{2L} = \frac{(x_{i,1} - x_{i-1,2})}{2L},
\]

\[
\gamma = \frac{x(p_i) - (L^2 - (y(p_i) - y(q))^2)^{\frac{1}{2}} - (x(p_{i-1}) - (L^2 - (y(p_{i-1}) - y(q))^2)^{\frac{1}{2}})}{2L}.
\]

In the for-loop, at line 4 of Algorithm 1, we know the values of \( j \) and \( k \). With these values known we need to perform the comparison \( j + \gamma + 1 - k \leq 0 \). Note that this is a branching point depending on a comparison which involves a polynomial in \( L, \ poly(L) \).

\[
j + \frac{x(p_i) - (L^2 - (y(p_i) - y(q))^2)^{\frac{1}{2}} - (x(p_{i-1}) - (L^2 - (y(p_{i-1}) - y(q))^2)^{\frac{1}{2}})}{2L} + 1 - k \leq 0,
\]

\[
2(j + 1 - k)L + (x(p_i) - x(p_{i-1})) - (L^2 - (y(p_i) - y(q))^2)^{\frac{1}{2}} + (L^2 - (y(p_{i-1}) - y(q))^2)^{\frac{1}{2}} \leq 0,
\]

\[
2(j + 1 - k)L + (x(p_i) - x(p_{i-1})) \leq (L^2 - (y(p_i) - y(q))^2)^{\frac{1}{2}} - (L^2 - (y(p_{i-1}) - y(q))^2)^{\frac{1}{2}}.
\]

On simplification the above inequality becomes

\[
2((j + 1 - k)^2 - 1)L^2 + 4(j + 1 - k)(x(p_i) - x(p_{i-1}))L + (x(p_i) - x(p_{i-1}))^2
\]

\[
+ 2\sqrt{(L^2 - (y(p_i) - y(q))^2)(L^2 - (y(p_{i-1}) - y(q))^2)} \leq 0 \tag{4}
\]

that involves a degree-two polynomial, \( poly(L) \): \( A L^2 + B L + C + 2((L^2 - D)(L^2 - E))^\frac{1}{2} = 0 \), where the coefficients \( A, \ B, \ C, \ D \) and \( E \) depend on the values known at the point of execution of the corresponding comparison step. To simulate comparison steps in the for loop at line 6, we compute all the roots of the associated polynomial \( poly(L) \) and invoke Algorithm 1 with the value of \( L \) equal to each of these roots. This yields an interval between two roots or a root and 0 or \( |pq|/(2(k - 1)) \) that contains \( L^* \). This enables us to determine the sign of \( poly(L^*) \) and proceed with the generic execution of the next step of the algorithm. Essentially, each time Algorithm 1 returns yes for the guessed value of \( L \), the region \( R \) (which depends on the right end of interval containing \( L^* \)) is shrinking, and hence the number of complemented intervals in \( I^C \) on which we have to run Algorithm 1 is reducing, until the shrunk \( R \) corresponds to \( L^* \). Finally, after completing the simulation of the for loop, if \( j = k \), then we return \( L^* \). To construct the set \( D = \{d_1, d_2, \ldots, d_k\} \), we run Algorithm 1 with the computed \( L^* \) one more time. Therefore, we have the following theorem.

**Theorem 3.5** We have an algorithm to solve the COFL problem in \( O((n + k)^2) \) using parametric search technique.

**Proof:** From [6] we know that Algorithm 1 runs in \( O(n + k) \) time. In the worst case, for each step of Algorithm 1 we obtain two different values of \( L \) and invoke Algorithm 1 on each of them as the candidate radius. Let \( L_1, L_2, \ldots, L_t \) be those different values of \( L \) across the entire simulation, where \( t = O(n + k) \). Then, initially \( r_{max} \in [0, |pq|/k] \). After the entire simulation is completed, clearly \( r_{max} = L^* = \max \{L_u \mid DCOFL(P, k, L_u) = \text{yes}, u = 1, 2, \ldots, t\} \). Since degree of the polynomial in inequality (4) is at most 2 and the decision algorithm \( DCOFL(P, k, L_u) \) is monotone for any \( L_u \in \mathbb{R}^+ \cup \{0\} \), the entire setup fits in the framework of parametric search. Hence, the correctness of the algorithm follows and the overall time for the simulation is \( O((n + k)^2) \). \( \square \)
3.3 Improved algorithm for $k = 2$

Here, we show that the decision problem $\text{DCOFL}(P, 2, L)$ can be solved in $O(\log n)$ parallel time using $n$ processors. Let $Q(L, P)$ be the complement of the union of $n$ open disk of radius $L$ centered at the demand points in $P$. Let $S(\overline{pq}, L)$ be the intersection of $Q(L, P)$ and $\overline{pq}$, which is the collection $I^c$ of $O(n)$ disjoint feasible intervals $[r_{i-1}, l_i] \subset \overline{pq}$, $i = 1, 2, \ldots, m = O(n)$, where the coordinates $r_{i-1} = (x_{i-1,2}, y_{i-1,2})$, and $l_i = (x_{i,1}, y_{i,1})$. Let an infeasible interval $[x_{i,1}, x_{i,2}]$ be an interval on $\overline{pq}$, which is not feasible for centering a facility in that (see Figure 15). When we have the infeasible intervals computed, implicitly we also have computed the feasible intervals in $I^c$. A parallel algorithm for computing these intervals is as follows: (1) We assign one demand point from $P$ to each of the $n$ processors. (2) Let each compute the corresponding infeasible interval $[x_{i,1}, x_{i,2}] = d_i \cap \overline{pq}$, where $d_i$ is the open disk of radius $L$ centered at $p_i \in P$ for $i = 1, 2, \ldots, n$ (see Figure 15). (3) Then, the merging of consecutive overlapping infeasible intervals into one bigger infeasible interval is performed by processors as follows: If an interval is not overlapping with its adjacent intervals then this interval is maintained on the same processor and for each pair of consecutive overlapping intervals, the processor of the first interval merges them into one and the second processor sits idle. In this way, for a sequence of consecutive overlapping intervals, alternating processors will perform the merging. This process will be repeated until there are only isolated intervals. Since in every step we merge a pair of consecutive overlapping intervals there will be at most $\log n + 1$ steps in total and each step will take $O(1)$ parallel time. Also, the serial time to construct $S(\overline{pq}, L)$ is only $O(n)$.

Now choosing two farthest points on the intervals in $S(\overline{pq}, L)$ is easy, just pick the left end point of the left most interval and right end point of the right most interval in $I^c$, in $O(1)$ time, and place the two facilities centered at these points.

Therefore, overall time of the parametric algorithm to solve COFL for $k = 2$ is $O(T_p \cdot n \cdot \log n + T_p \cdot T_s \cdot \log n) = O(\log n \cdot n \cdot \log n + \log n \cdot n \cdot \log n) = O(n \log^2 n)$ time, where $T_p$ denotes a parallel time and $T_s$ denotes a serial time for solving $\text{DCOFL}(P, 2, L)$ with $n$ processors.

This is an improvement over the earlier FPTAS as well as the two proposed exact algorithms here for
Remark 3.2 The COFL under the Euclidean norm for $k = 2$ can be solved in $O(n \log^2 n)$ time using the improved parametric technique.

4 Circular COFL problem

In this section, we define a variant of the COFL problem in which the centers of the disks are restricted to lie on the boundary arc of a predetermined circle. The motivation for this variant of the problem is as follows. Consider a city, and now we need to place $k$ number of facilities such as dumping yards or fuel stations around the city such that the minimum distance between these facilities and between the facilities and demand points has to be maximized. It is also required to place these $k$ facilities as far apart from each other as possible along the boundary arc $\partial \mathcal{C}$ to avoid self competition among fuel stations and to avoid one of two or more close-by dumping yards getting overloaded quickly.

The circular constrained obnoxious facility location problem (CCOFL) problem is defined as follows: Given a set $\mathcal{P} = \{p_1, p_2, p_3, \ldots, p_n\}$ of $n$ demand points in the plane, a predetermined circle $\mathcal{C}$ with radius $r_c$ and a positive integer $k$, locate $k$ facility sites on the boundary arc $\partial \mathcal{C}$ of $\mathcal{C}$ such that each demand point in $\mathcal{P}$ is farthest from its closest facility site (i.e., in terms of their Euclidean distance) and the facility sites are placed farthest from each other along $\partial \mathcal{C}$ (i.e., in terms of arc length). Observe here that two disks representing two consecutive facility sites placed on $\partial \mathcal{C}$ may overlap strictly inside $\mathcal{C}$.

The decision version of this problem DCCOFL($P, k, L$) can be solved by using a similar method that was used to solve the DCOFL($P, k, L$) problem under the assumption that $r_c >> L$. When $r_c < L$, it is trivial that only one disk with radius $L$ will be packed on $\partial \mathcal{C}$ as the center of $\mathcal{C}$ will lie inside that packed disk. Now, to solve the decision version of CCOFL problem, we consider two circles $\mathcal{C}_1$ and $\mathcal{C}_2$ which are concentric with $\mathcal{C}$ and whose radii are $r_c - L$ and $r_c + L$, where $L$ is a real number (see Figure 16). We can observe that the points lying inside $\mathcal{C}_1$ and outside of $\mathcal{C}_2$ will not influence the packing of the disks. Similar to that of DCOFL($P, k, L$) problem here, we can obtain at least one center point and at most two center points on the boundary of $\mathcal{C}$ which are at a distance of $L$ from each point lying outside of $\mathcal{C}_1$ and inside of $\mathcal{C}_2$ (see $p_1, p_2$ and $p_k$ in Figure 16).

Let $c_{i,1}$ and $c_{i,2}$ be the center-points corresponding to $p_i$, then none of the $k$ disks in an optimal solution to DCOFL($P, k, L$) will have their center points lying on the open arc interval $(c_{i,1}, c_{i,2})$ of the boundary of $\mathcal{C}$.

Now, let $(c_{j,1}, c_{j,2})$ and $(c_{k,1}, c_{k,2})$ be the center points corresponding to $p_j$ and $p_k$ respectively. In Figure 16, we can observe that the intervals $[c_{i,1}, c_{i,2}]$, $[c_{j,1}, c_{j,2}]$ and $[c_{k,1}, c_{k,2}]$ formed by $c_{i,1}c_{i,2}$, $c_{j,1}c_{j,2}$ and $c_{k,1}c_{k,2}$ are overlapping. Hence, none of the $k$ disks in the optimal solution will have their centers lying on the interval $([c_{i,1}, c_{i,2}] \cup [c_{j,1}, c_{j,2}] \cup [c_{k,1}, c_{k,2}]) \setminus \{c_{i,1}, c_{i,2}\}$, excluding the end-points of the union of the two intervals.

Without loss of generality, let $\{p_1, p_2, \ldots, p_m\}$ be the points of $\mathcal{P}$ lying strictly outside of $\mathcal{C}_1$ and inside of $\mathcal{C}_2$, ordered clockwise on the boundary of $\mathcal{C}$, where $m \leq n$. We know that for every point $p_i$ lying strictly outside of $\mathcal{C}_1$ and inside of $\mathcal{C}_2$ there will be two center-points on the boundary of $\mathcal{C}$ which are at distance $L$, i.e., there is an interval $[l_i, r_i]$ for every point $p_i$, where $l_i = c_{i,1}$ and $r_i = c_{i,2}$. Merge all the overlapping intervals and then update the end-points of the new intervals on boundary of $\mathcal{C}$. Let $I = \{[l_1, r_1], [l_2, r_2], \ldots, [l_m', r_m']\}$ be the set of resulting pairwise disjoint intervals ordered clockwise, where $m' \leq m$. 

$k = 2$. 

The COFL under the Euclidean norm for $k = 2$ can be solved in $O(n \log^2 n)$ time using the improved parametric technique.
Consider the complement of $I$ with respect to boundary of the circle $C$, denoted as
$$I^c = \{ [r_1, l_2], \ldots, [r_m', l_1] \}.$$  

The arc length of the complemented intervals in $I^c$ can be calculated using the law of cosines formula as follows: The angle ($\theta$) subtended by arc at the center of $C$ is
$$\theta = \arccos \left(1 - \frac{d^2}{2r_c^2}\right)$$
where $d$ is the Euclidean distance between the end-points of the arc segment. Then, the length of the arc interval $[r_i, l_{i+1}]$ is $||r_i, l_{i+1}|| = r_c \theta$, where $\theta = \arccos \left(1 - \frac{d^2}{2r_c^2}\right)$ for $i = 1, 2, \ldots, m'$.

**Observation 4.1** Without loss of generality, we can assume that the first disk $d_1$ is centered at one of the end-points of arc segment in $I^c$.

Since we don’t know, the position of the disks for given $L$ in the optimal packing, we greedily pack disks by placing centers on $\partial C$ with the first disk $d_1$ at every endpoint of the arc segments in $I^c$. As there are $O(n)$ end-points in $I^c$, we have the following theorem.

**Lemma 4.2** Given the set $I^c$ of complemented intervals and an integer $k > 0$, Algorithm 1 solves the DCCOFL($P, k, L$) problem in $O(n (n + k))$ time.

**Proof:** Follows from Observation 4.1 and the fact that there are $O(n)$ end points of intervals in $I^c$.

**Theorem 4.3** We can get an $(1 - \epsilon)$-factor approximation algorithm with $\epsilon > 0$ (FPTAS) for the CCOFL problem, that runs in $O(n(n+k) \log \left(\frac{||pq||}{2(k-1)^2}\right))$ time, by employing doubling search and bisection methods.

**Proof:** Follows from Lemma 4.2 and Theorem in [6].

**Remark 4.1** [7] Unconstrained COFL problem can be solved in $O(n \log n)$ time for $k = 1$ by finding the largest empty circle in a Voronoi diagram formed by $n$ demand points as the voronoi sites.
5 Conclusion and future work

We proposed a brute force algorithm based on binary search that solves the COFL problem exactly in $O((nk)^2 \log(nk) + (n+k) \log(nk))$ time for both rectilinear and Euclidean cases. We showed that using Megiddo's parametric search technique, we can improve the running time to $O((n+k)^2)$ time. For $k = 2$, we have a parametric algorithm that runs in $O(n \log^2 n)$ which would improve the parametric algorithm. It is not clear how to design a parallel algorithm for DCOFL($P, k, L$) for any $k$. We also gave a FPTAS for CCOFL problem that runs in $O(n(n+k) \log(\frac{|pq|}{2(k-1)\epsilon}))$ time.

For any $k \geq 2$, unconstrained continuous OFL is an elusive open problem since it was first mentioned by Kartz et al. [3]. However, for rectilinear variant UCOFL is tractable at least from exact exponential algorithm, and for $k = 1, 2$, there are polynomial time algorithms.

References

[1] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM (JACM)*, 30(4):852–865, 1983.

[2] S. Abravaya and M. Segal. Maximizing the number of obnoxious facilities to locate within a bounded region. *Computers & operations research*, 37(1):163–171, 2010.

[3] M. J. Katz, K. Kedem and M. Segal. Improved algorithms for placing undesirable facilities. *Computers & Operations Research*, 29(13):1859–1872, 2002.

[4] G. Frederickson and D. Johnson. Generalized selection and ranking: sorted matrices. *SIAM Journal of Computing*, 13:14–30, 1984.

[5] D. S. Hochbaum and W. Maass, 1985. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM (JACM)*, 32(1):130–136, 1985.

[6] Sigireddy V., R. and Basappa, M.: Constrained Obnoxious Facility Location on a Line Segment. In: Proceedings of the 33rd Canadian Conference on Computational Geometry, 362–367 (2021).

[7] Shamos, M. I., and Hoey, D.: Closest-point problems. 16th Annual Symposium on Foundations of Computer Science (sfcs 1975), 151–162 (1975).

[8] Agarwal, P. K., Sharir, M. and Toledo, S., 1994. Applications of parametric searching in geometric optimization. *Journal of Algorithms*, 17(3), pp.292-318.

[9] Díaz-Báñez, J. M., Hurtado, F., Meijer, H., Rappaport, D., Sellarès, J. A., 2003. The largest empty annulus problem. *International Journal of Computational Geometry & Applications*, 13(04), pp.317-325.

[10] Bae, S.W., Baral, A. and Mahapatra, P.R.S., 2021. Maximum-width empty square and rectangular annulus. *Computational Geometry*, 96, p.101747.

[11] Church, R.L. and Drezner, Z., 2022. Review of obnoxious facilities location problems. *Computers & Operations Research*, 138, p.105468.

[12] Qin, Z., Xu, Y. and Zhu, B., 2000, July. On some optimization problems in obnoxious facility location. In *International Computing and Combinatorics Conference* (pp. 320-329). Springer, Berlin, Heidelberg.