Computing Weighted k-Centers on a Line

Mohammad Motiei Shervin Daneshpajouh Sadra Yazdanbod

Abstract

We consider the weighted k-center problem. For a given set of point P on the plan we like to find k center on a line such that $\max_{p\in P}\{w(p).d(\Delta,p)\}$ is minimized, where $w(p)$ is the weight of a point p, $\Delta = \{c_1, c_2, \ldots, c_k\}$ is the set of centers, $d(\Delta,p) = min_{j=1,\ldots,k}d(c_j,p)$. We study three variantes of this problem. We first study 1-center problem where the line is given in advance. To solve this problem, we define a new sub-problem called supper point problem. We solve this problem by solving the sub-problem in total time $O(n\log^2 n)$. Then, we investigate the k-center case of the previous problem and we present an $O(n^2\log^2 n)$ time algorithm. Finally, we study the k-center case where the orientation of the line is given and solve it in $O(n^4\log^2 n)$ time.

1 Introduction

In this paper, we consider a variation of the k-center problem where the facilities are constrained to lie on a given line. We are given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n planar points lying on the plane where each point $p \in P$ is associated with a non-negative weight $w(p)$. Let $d(u,v)$ be the euclidean distance between the points u and v. We define $S(\Delta,P)$ which denotes the *service cost* of a set centers $\Delta = \{c_1, ..., c_k\}$ for a set of weighted points P, that is: $S(\Delta,P) = \max_{p\in P}\{w(p).d(\Delta,p)\}$, where $w(p)$ is the weight of point p and $d(\Delta,p) = min_{j=1,\ldots,k}d(c_j,p)$. A demand point p is called *dominating demand point* of the center set X if $w(p).d(X,p) = S(X,P)$. Our objective is to determine, for a fixed constant k, a set X of k centers on l such that $S(X,P)$ is minimized.

Motivation and Related Works

The above problem is a type of facility location problem and it has applications in different areas like shape simplification. The unweighted k-center problem, $w(p_i) = 1$, has been studied before. For $k = 1$ on a fixed line, Megiddo presented an $O(n)$ time algorithm [1]. For $k = 2$, Chan proposed a deterministic and a randomized algorithms that run $O(n\log^2 n(\log\log n)^2)$ and $O(n\log^2 n)$ time respectively [3]. For $k \geq 2$, Brass *et al.* presented an $O(n\log^2 n)$ time algorithm for a given line [2]. For the case where the orientation of the line is given, they presented an $O(n^2\log^2 n)$ time algorithm [2]. Also, they show that it is possible to find the k-center for arbitrary line in $O(n^4\log^2 n)$ expected time [2].

Our Results

In this paper, we study three cases of the problem: (i) weighted 1-center problem on a fixed line. (ii) weighted k-center on fixed line, and (iii) weighted k-center on a line with fixed orientation. We present an $O(n\log n)$ time algorithm for the first problem. We solve the second problem in $O(n^2\log n)$ time. We show that the third problem can be solved in $O(n^4\log n)$ time.

2 Weighted one center on a fixed line

In this Section we provide an $O(n\log n)$ algorithm to solve the weighted one center on a fixed line version of our problem. We are given a set of n weighted point P and a fixed line l, we want to find a center δ which lies on l with the minimum service cost. Note that, the service cost for a center δ' defines as $\max_{p\in P} d(\delta',p)w(p)$. Let us call the problem, *WOFLP*. We can assume the fixed line is parallel to x-axis without loss of generality.

We call a set of exact three point (x, ls, rs), a *super point* a if the point x lies on l and the points ls and rs be on the left side and on the right side of x receptively. For a super point a and a point p which lies on l, we define $d_s(a,p)$ such that if the point p be on the right side of $a.x$ we set $d_s(a,p) = d(a.ls,p) * w(p)$ and otherwise, we set $d_s(a,p) = d(a.rs,p)*w(p)$. Let call $d_s(a,p)$ the distance between a point and a super point. Now, consider a supper point problem *SPP* as follows:

SPP: we are given fixed line l and a set S which contains n super points. We want to find the center δ on l which has the minimum of maximum distance from the super points.

In this Section, we first prsent an $O(n\log n)$ time algorithm for the SPP problem and then we will use it to provide an $O(n\log n)$ algorithm to solve the WOFLP problem.

2.1 The SPP problem algorithm

In this section, we provide an $O(n\log n)$ algorithm for the SPP problem. For two super points a and b, we define $d(a,b) = d(a.x, b.x)$ as the distance between two super points.

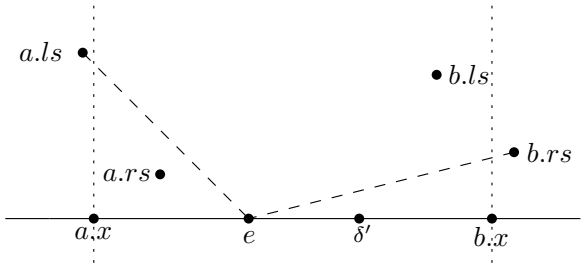

Figure 1: The center δ' is placed on the right side of e.

We start with the set S as S_0. In each step of our algorithm, we construct a new set of super points S_i from the set S_{i-1} until $|S_{i-1}| \geq 2$. For each step of our algorithm, we do as follow:

- We find the two super points a and b in S_{i-1} such that $d(a,b) = max_{c,d \in S_{i-1}}\ d(c,d)$.
- We construct the set S_i by adding new super point $e = (x', a.ls, b.rs)$ to S_{i-1}, such that $d(x', a.ls)w(a.ls) = d(x', b.rs)w(b.rs)$.
- We remove the points a and b from the set S_i.

We continue these steps until we find the set $|S_n| = 1$. Now, we set the center δ equal to $f \in S_n$, the only super point in S_n.

Now, we prove the correctness of the algorithm. It is obvious that we just need to prove, in each step of the algorithm the optimal centers δ_i and δ_{i-1} for the sets S_i and S_{i-1} are equal. Let the super points a and b be those super points which are found in the step i and without loss of generality assume the point $a.x$ is placed on the right side of the point $b.x$. Note that, the centers δ_i and δ_{i-1} are located between the points $a.x$ and $b.x$ because otherwise the centers are in right side of all super points or vice versa. We say a point p which lies on the line l is located on the right side of a super point a if p was located on the right side of the point $a.x$. In this case we can move the center to the left/right by an ϵ and get better service cost. Now, assume an arbitrary center δ' between the points $a.x$ and $b.x$. Let e be the super point that is added to the set in the step i. First let us, assume the center δ' are placed on the right side of the point $e.x$.

In this case we have $d_s(e,\delta') = d(e.ls,\delta')w(e.ls) = d(a.ls,\delta')w(a.ls)$. Note that we placed the point e at the place where $d(e.x, a.ls)w(a.ls) = d(e.x, b.rs)w(b.rs)$, so $d(a.ls,\delta')w(a.rs) > d(b.rs,\delta')w(b.rs)$. It is not hard to see that we can have the same result if the point δ' was placed in the left side of $d.x$. Therefore $d_s(e,\delta') = max(d(a.ls,\delta')w(a.ls), d_S(b.rs,\delta')w(b.rs))$. Since the center δ' was placed between $a.x$ and $b.x$, we have $d_s(e,\delta') = max(d_s(a,\delta'), d_s(b,\delta'))$. Now, we claim that for every arbitrary center δ' between $a.x$ abd $b.x$, the service cost for both sets S_i and S_{i-1} are equal since we have:

$$\begin{gathered}\max_{p\in S_{i-1}}(d_s(p,c')) = \\ \max(\max_{p\in S_{i-1}\setminus\{a,b\}}(d_s,(p,\delta')), \max(d_s(a,\delta'), d_s(b,\delta'))) \\ = \max(\max_{p\in S_{i-1}\setminus\{a,b\}}(d_s,(p,\delta')), d_s(d,\delta')) \\ = \max_{p\in S_i}(d_s,(p,\delta')).\end{gathered}$$

So, the centers δ_i and δ_{i-1} are equal and the proof of correctness of the algorithm is complete.

To implement the algorithm, we need a data structure that can provide the add, remove, and findmax/findmin operations on a set of elements of maximum size n in $O(\log n)$ time. It is possible to do these operations in such a time by employing a Red-black tree data structure. Therefore, each step of the algorithm takes $O(\log n)$ time and it is obvious that the algorithm has n steps, so the overall time of the algorithm is $O(n \log n)$.

2.2 The WOFLP problem algorithm

In this section we use the algorithm of the SPP problem to solve the WOFLP problem with $O(n \log n)$ time. In the WOFLP, we are given a set of n weighted points P and a fixed line l, and we want to find the center δ with the minimum service cost. We can assume the fixed line is parallel to x-axis without loss of generality. To solve the WOFLP, we do as follows:

- We construct a set of super points S by adding a super point (p', p, p) for every $p \in P$ such that p' is the nearest point on l to p.
- We use the $O(n \log n)$ algorithm to solve the SPP for the set S and the line l. Let call the output center δ'.
- We set the output center of the WOFLP equal to δ'.

To prove the correctness of the algorithm we need to prove for any arbitrary center δ'' the service cost on P is equal to the maximum distance from δ'' to the super points in S and it is easy because for every point in $p \in P$ and its corresponding super point $sp \in S$, we have $d(p,\delta'') = d_s(sp,\delta'')$ since $sp.ls = sp.rs = p$.

It is obvious that the time of the algorithm remains equal to $O(n \log n)$. Therefore, we conclude this in the following theorem.

Theorem 1 *Given a set of n point P and a fixed line l, we can find the center δ with the minimum service cost with $O(n \log n)$ time.*

3 Weighted k-centers on a fixed line

In this section, we consider the weighted k-center problem for a given fixed line l. Let σ^* be the *optimal service*

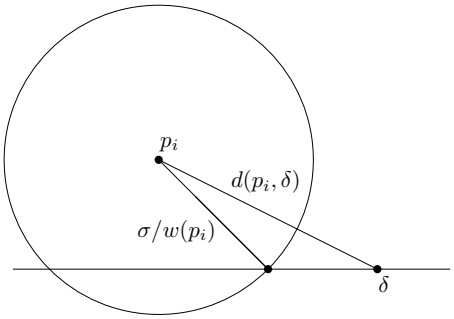

Figure 2: In this figure, δ lies outside of c_i. So, $\sigma/w(p_i) < d(p_i, \delta)$.

cost such that there exists k centers on the line l with $d(C, p_i)w(p_i) \leq \sigma^*, 1 \leq i \leq n$. Without loss of generality, we assume that l is parallel to x-axis. First, we present an algorithm that decides whether there is a solution for a fixed value σ or not. Then, we show how to find the optimal σ^*. Now, we prove the following lemma.

Lemma 2 *For a given line l and a fixed parameter σ. It is possible to locate k centers on l, such that for all $p_i \in P$ $d(\Delta, p_i)w(p_i) \leq \sigma$, iff for each circle c_i centered at the point $p_i \in P$ with the radious $\sigma/w(p_i)$ there would be a center δ on the line l, which lies inside that.*

Proof. To the contrary, let p_i be a point such that its' closest center δ, is located outside of c_i. Then, we have:

$$d(p_i, \delta) > r(c_i) \Rightarrow d(p_i, \delta) > \sigma/w(p_i)$$
$$\Rightarrow d(p_i, \delta)w(p_i) > \sigma \quad (I)$$

The equation I contradicts with the fact that σ is service cost (See Fig. 2). □

3.1 The decision algorithm

We define the decision problem as follows: Given a parameter σ and a line l decide if it is possible to locate k centers on l such that for all points $p_i \in P$, $d(C, p_i)w(p_i) \leq \sigma$. Let $r_i = \sigma/w(p_i)$. Let c_i be a circle centered at p_i with radious r_i. We say a point p is covered by a center δ if $d(\delta, p)w(p) \leq \sigma$. Let I_i be the intersection of c_i with line l such that $I_i = [a_i, b_i]$ be the interval on line l with two end points $a_i \leq b_i$. Let $I = I_i, 1 \leq i \leq n$, be a set of intervals. A *piercing set* of I is a set of points on l such that every interval in I contains at least one point in the piercing set. We call the minimum cardinality of a piercing set for I the *minimum piercing number* $k(\sigma)$ of I. This number can be computed in $O(n \log n)$ time by selecting a piercing point at the leftmost right endpoint, removing all the intervals containing the piercing point, and repeating the same process for the remaining intervals until all the intervals are removed, c.f. [4, 5]. As piercing set I has a center in each interval, by Lemma 2, we can conclude that P can be covered by the service cost σ. if and only if the minimum piercing number $k(\sigma)$ is not larger than k. Thus we can answer the decision problem in $O(n \log n)$ time.

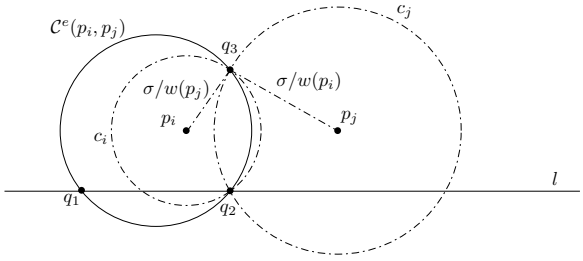

Figure 3: The equality circle for p_i and p_j in which $d(p_i, a)w(p_i) = d(p_j, a)w(p_j)$. q_1 and q_2 are the intersection points of $\mathcal{C}^e(p_i, p_j)$ with line l.

3.2 Computing a set of candidates

In this section, we compute a set of candidate service costs denoted by A. Then, we show that the minimum service cost is a member of A.

Definition: It is known that, the locus of the points on the plane with the same weighted distance from two different weighted points p and q is a circle. We define an *equality circle* $\mathcal{C}^e(p, q)$ as a set of points on the plane with the same weighted distance from points p and q (See Fig. 3).

Let p'_i be the closest point to p_i on the l. First, we compute $\max\{d(p_i, p'_i)w(p_i), 1 \leq i \leq n\}$, and add it as a candidate service cost to A. Clearly, this value is a lower bound for the optimal service cost. Let $p, p' \in P$ be two arbitrary points. We define $\mathcal{Q}(p, p', l)$ as the set of intersection points of $\mathcal{C}^e(p, p')$ with the line l. For each $p_i, p_j \in P$, if $\mathcal{Q}(p_i, p_j, l) \neq \varnothing$, then we add $d(\mathcal{Q}(p_i, p_j, l), p_i)w(p_i)$ to the set A. Obviously, the size of A is less than $n^2 + 1$.

Now, we prove the following lemma.

Lemma 3 *The set of candidate service costs A contains the optimal service cost σ^*.*

Proof. To the contrary, assume that the optimal service cost σ^* is not in A. If σ^* is the optimal service cost, then the set of intervals I on line l for this service cost, form two cases. In the first case, there exist two intervals I_i and I_j such that they overlap exactly in one point q (See Fig. 3 where the intervals overlaps in a point q_2). In this case, q has the same weighted distance from p_i and p_j, so it is the intersection point of $\mathcal{C}^e(p_i, p_j)$ with l. Hence, $d(p_i, q)w(p_i) = \sigma^*$ which is already added to the set A that contradicts our assumption.

In the second case, we know that each two intervals I_i and I_j that have intersection, overlap in more than a point. It is easy to see that in this case, there exist an $\epsilon > 0$, so we can reduce σ^* by ϵ, such that $min\{l^o_{i,j}, 1 \leq i, j \leq n\}$ is bigger than zero, where $l^o_{i,j}$ is the lenght of intersection I_i and I_j. Therefore, we can find k centers on l by service cost equal to $\sigma^* - \epsilon$, which contradicts our assumption that σ^* is the optimal service cost.

□

3.3 Finding the optimal service cost

First, we sort the set of candidate values in A. Then, the optimal service cost σ^* can be easily found by doing a binary search on the orderd values using the decision algorithm described in 3.1.

3.4 Running time and memory analysis

Computing the set of candidates A can be done in $O(n^2)$ time using $O(n^2)$ memory size. This set is sorted in $O(n^2 \log n)$ time. The decision algorithm runs in $O(n \log n)$ time. So, the binary search and call of decision algorithm, runs in $O(n \log^2 n)$ time. Hence, the whole algorithm runs in $O(n^2 \log n)$ time using $O(n^2)$ memory size.

We conclude this in the following theorem.

Theorem 4 *Given a set of weighted points P, an integer k, and a fixed line l on the plane, we can find a set of k centers $\{c_1, \ldots, c_k\}$ on l such that $max_{p \in P}\{w(p).d(C,p)\}$, is minimized, in $O(n^2 \log n)$ time, where $w(p)$ is the weight of point p, and $d(C,p)$ is the minimum distance of p from a center in C.*

4 Weighted k-centers on a line with fixed orientation

We now consider the case where only the orientation of the line l is fixed. Without loss of generality assume that l is parallel to x-axis. Similar to the previous case, we first present an algorithm that decides whether there is a solution for a fixed service cost σ or not. Then, we show how to find the optimal σ^* .

4.1 The decision algorithm

We define the decision problem as follows: Given a parameter σ decide whether it is possible to find a line l parallel to the x-axis with k centers on l such that for all points $p_i \in P$, $d(C,p_i)w(p_i) \leq \sigma$. It is easy to see the following lemma is correct.

Definition:

Lemma 5 *Suppose, a parameter σ and two lines l_1 and l_2 are given. Let $I(l_1) = \{[a_1,b_1],[a_2,b_2],\ldots\}$ and $I(l_2)$ be the set of intervals formed by the parameter σ on two lines l_1 and l_2 respectively. Let $I'(l_1)$ and $I'(l_2)$ be respectively the set of endpoints of intervals of $I(l_1)$ and $I(l_1)$ ordered by their x-coordinate. Let $\mathcal{I}(l_1)$ and $\mathcal{I}(l_2)$ be the sequence of indices of points in $I'(l_1)$ and $I'(l_2)$ respectively. Suppose $\mathcal{I}(l_1) = \mathcal{I}(l_2)$. If there exist a k-center on line l_1 such that $\forall p_i \in P : d(\Delta(l_1), p_i)w(p_i) \leq \sigma$, then there is k-centers on l_2 that have this property.*

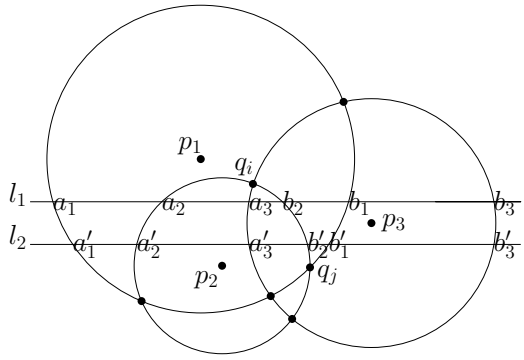

Figure 4: In this figure, we have $I(l_1) = \{[a_1,b_1],[a_2,b_2],[a_3,b_3]\}$ and $I(l_2) = \{[a'_1,b'_1],[a'_2,b'_2],[a'_3,b'_3]\}$. Also, $I'(l_1) = \{a_1,a_2,a_3,b_2,b_1,b_3\}$ and $I'(l_2) = \{a'_1,a'_2,a'_3,b'_2,b'_1,b'_3\}$. So, $\mathcal{I} = \mathcal{I}' = \{1,2,3,2,1,3\}$.

Let Q be a set of all the intersection points of the circles c_i centerd at each $p_i \in P$ with the radious $\sigma/w(p_i)$. Clearly, size of Q is less than n^2. Let Q' be the set of points in Q sorted by their y-coordinate. It is easy to see the following observation is true.

Observation 1 *Let l_1 and l_1 be two arbitrary lines parallel to x-axis between two consecutive points q_i and q_j in the ordered set of intersection points Q', that is $y(q_i) \leq y(l_2), y(l_2) \leq y(q_j)$. In this case, $\mathcal{I}(l_1)$ is equal to $\mathcal{I}(l_1)$ (See Fig. 4).*

Therefore, for each two consequtive intersection points q_i and q_j in the ordered set of intersection points Q', we can chose an arbitrary line parallel to x-axis between q_i and a_j and call the decision algorithm 3.1 for fixed line. The algorithm stops when the decision algorithm 3.1 returns true or we have checked all the consecutive points. The whole algorithm runs in $O(n^3 \log n)$ as there are at most n^2 intersection points and the decision algorithm 3.1 runs in $O(n \log n)$.

4.2 Computing a set of candidates

We first define the *critical situation* as follows.

Definition: The critical situation is a case in which for a set of centers X on a line l, any change in the position of centers, or line increase the service cost or the number of dominating demand points.

It is easy to see that if the problem has a solution, then there exist a critical situation that has service cost equal to the optimal one. Now, we compute a set A containing candidate valuses for σ^* by checking all critical situations. In what follows, we first present a lemma and an observation. Then, we use them to find candidate values σ by checking all the critical situations.

Lemma 6 *Suppose that for a given line l and a paramter σ, k centers located on l such that for all $p_i \in P : d(\Delta, p_i)w(p_i) \leq \sigma$ and we have a critical situation. Let δ be one of the k centers that covers a dominating demand point p_i. Then, δ must be placed in one of the following positions: (i) on the closest point of l to p_i, (ii) on one of the intersections of l with $\mathcal{C}^e(p_i, p_j)$, where $p_j \in P, j \neq i$.*

Proof. To the contrary, suppose that δ is not located in possisions (i) and (ii). In this case each $p_j, j \neq i$ that is covered by δ, cannot be a dominating demand point. Note that, if p_j is a dominating demand point, then δ has the same weighted distance from both p_i and p_j. So, p_j lies on the intersection point of $\mathcal{C}^e(p_i, p_j)$ with l which contradict our assumption (case ii). As δ has only one dominating demand point, we can move δ on line l by a size ε such that $d(\delta, p)w(p)$ decreases and the weighted distances between δ and each point p covered by δ, does not exceed σ. Thus, p is no longer a dominating demand point and the number of dominating demand point decreases. This contradicts with the fact that the number of dominating demand points is minimum. □

Observation 2 *Suppose that for a given line $l : y = y_0$ and a paramter σ, k centers located on l such that for all $p_i \in P : d(\Delta, p_i)w(p_i) \leq \sigma$ and we have a critical situation. Then, there must be two demand points p_i and p_j in P, such that $y(p_i) \geq y_0$ and $y(p_j) \leq y_0$.*

Now, in order to compute the candidate values σ, we identify all the critical situations. We have two cases:

Case (i) All of the dominating demand points are covered by one center. Let line l be a line parallel to x-axis. Let δ be the center on l that covers all the dominating demand points. Note that, in this case some non-dominating demand points might be covered by the other centers. We have following cases based on the number of dominating demand points.

Case (i.i) Having two dominating demand points. Let p_r and p_s be the two dominating demand points that are covered by δ. By Lemma 6, δ should lies on the intersection point $\mathcal{C}^e(p_r, p_s)$ and line l. Let $q_{r,s}$ be the point with the minimum weighted distance from points p_r and p_s. It is easy to see that $\delta = q_{r,s}$. So, l should passes through $q_{r,s}$. Note that, in this case $q_{r,s}$ is the intersection point of the connecting line of p_r to p_s and $\mathcal{C}^e(p_r, p_s)$ (See Fig. 5). So, for each pair $p_i, p_j \in P$,

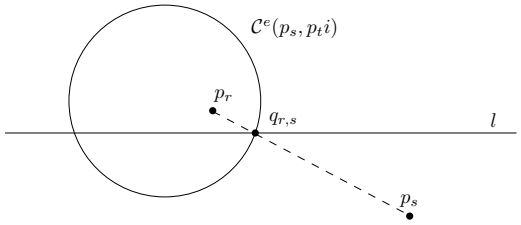

Figure 5: Case (i.i)

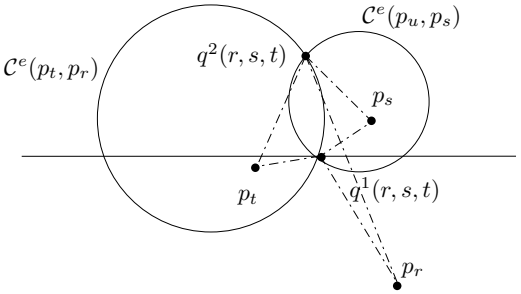

Figure 6: Case (i.ii)

we compute $d(q_{i,j}, p_i)w(p_i)$ and add it to the set A as a candidate value for σ.

Case (i.ii) Having more than two dominating demand points. Let points p_r, p_s, and p_t be the three arbitrary dominating demand points. Let $q^1_{r,s,t}$ and $q^2_{r,s,t}$ be the intersection point of $\mathcal{C}^e(p_r, p_s)$ and $\mathcal{C}^e(p_r, p_t)$. As $q^1_{r,s,t}$ and $q^2_{r,s,t}$ are the only points that have equal weighted distance from points p_r, p_s, and p_t, then δ lies on $q^1_{r,s,t}$ or $q^2_{r,s,t}$. Therefore, line l passes through $q^1_{r,s,t}$ or $q^2_{r,s,t}$. Now, for each three points p_i, p_j and $p_k \in P$, we add $d(q^1_{i,j,k}, p_i)w(p_i)$ and $d(q^2_{i,j,k}, p_i)w(p_i)$ to the set A (See Fig.6).

Case (ii) The dominating demand points are covered by more than a center.

Without loss of generality, Let p_r and p_s be two dominating demand points on both sides of l. Let δ_r and δ_s be the closest centers for p_r and p_s. Let q_r and q_s be the closes point on l to p_r and p_s respectively. We have the following cases for δ_r and δ_s.

Case (ii.i) δ_r is on q_r and δ_s is on q_s- In this case, for each pair of points $p_r, p_s \in P$, we check whether there is line l such that $d(p_r, q_r)w(p_r) = d(p_s, q_s)w(p_s)$. This check can be done in time $O(1)$ for each pair of point. If there exist such a line, we add $d(p_r, q_r)w(p_r)$ to the set A.

Case (ii.ii) δ_r is on q_r and δ_s is not on q_s- In this case, δ_s must lie on the $\mathcal{C}^e(p_s, p_i)$, by Lemma 6. For each three points $p_r, p_s, p_t \in P$, we check whether there is line l such that $d(p_r, q_r)w(p_r) = d(p_s, q_{s,t})w(p_s)$, where $q_{s,t}$ is a point on $\mathcal{C}^e(p_s, p_t)$ and l (See Fig. 7). This check can be done in time $O(1)$ for each pair of point (See Appendix A).If there exist such a line, we add all the $d(p_r, q_r)w(p_r)$ to the set A

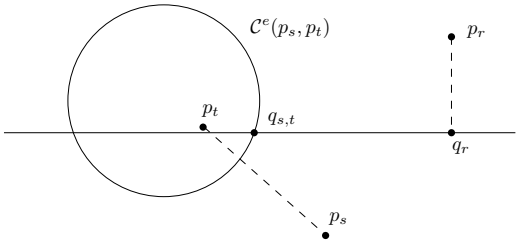

Figure 7: Case (ii-ii)

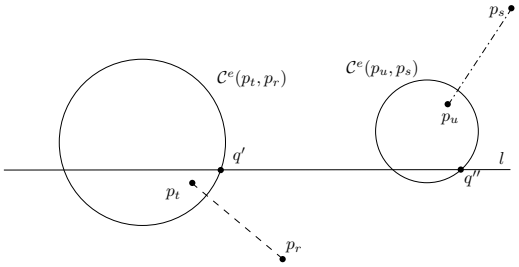

Figure 8: Case (ii-iv)

Case (ii.iii) δ_r **is on** q_r **and** δ_s **is not on** q_s This case is analogous to the previous case.
Case (ii.iv) δ_r **is not on** q_r **and** δ_s **is not on** q_s In this case, δ_r and δ_s must lie on the $\mathcal{C}^e(p_r, p_i)$ and $\mathcal{C}^e(p_s, p_j)$ respectively, by Lemma 6. For each four points $p_r, p_s, p_t, p_u \in P$, we check whether there is line l such that $d(p_r, q')w(p_r) = d(p_s, q'')w(p_s)$, where q' and q'' are the intersection points of l with $\mathcal{C}^e(p_r, p_t)$ and $\mathcal{C}^e(p_s, p_u)$ respectively (See Fig. 8). This check can be done in time $O(1)$ for each pair of point (See B). If there exist such a line, we add all $d(p_r, q')w(p_r)$ to the set A .

4.3 Finding the optimal service cost

First, we sort the set of candidate values in A. Then, the optimal service cost σ^* can be easily found by doing a binary search on the ordered values using the decision algorithm described in 4.1.

4.4 Running time and memory analysis

We compute the set of candidate values A in $O(n^4)$ time, using $O(n^2)$ memory size. Sorting the members of A can be done in $O(n^4 \log n)$ time. Also, as we mention before, the decision algorithm runs in $O(n^3 \log n)$ time. So, using a binary search and calling decision problem in each step of that, can be done in $O(n^3 \log^2 n)$ time. Hence, we can solve this problem in $O(n^4 \log n)$ time and $O(n^2)$ memory size.

we conclude this in following theorem.

Theorem 7 *Given a set of weighted points P, an integer k, and an orientaion for the line l, we can find the location of l and set k centers $\{c_1, \ldots, c_k\}$ on that such that $max_{p \in P}\{w(p).d(C, p)\}$, is minimized, in $O(n^4 \log n)$ time, where $w(p)$ is the weight of point p, and $d(C, p)$ is the minimum distance of p from a center in C.*

References

[1] N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems. *SIAM Journal on Computing 12*, 759–776, 1983.

[2] P. Brass, C. Knauer, H. Na, C. Shin, A. Vigneron. The aligned k-center problem. *International Journal of Computational Geometry and Applications*, 21(2):157–178, 2011.

[3] T.M. Chan. More planar two-center algorithms. *Computational Geometry Journal*, 13(3):189–19, 2009.

[4] T. M. Chan, A.-Al Mahmood. Approximating the piercing number for unit-height rectangles. *In Proc. of Canadian Conference on Computational Geometry* , 15–18, 2005.

[5] M. J. Katz, F. Nielsen, M. Segal. Maintenance of a piercing set for intervals with applications. *Algorithmica*, 36, 59–73, 2003.

A computation procedure of case ii.ii in the third Algorithm

Given three points $p_r, p_s, p_t \in P$. we check whether there is a line l such that $d(p_r, q_r)w(p_r) = d(p_s, q')w(p_s)$, where q_r is the closest point of l to p_r and q' is the intersection point of $\mathcal{C}^e(p_s, p_t)$ and l. Let a be the center of $\mathcal{C}^e(p_s, p_t)$ and r be the radious of that. We use capital letters to denote unknown variables and small letters for known variables. As q' lies on $\mathcal{C}^e(p_s, p_t)$ we have:

$$(x(a) - X(q'))^2 + (y(a) - Y(q'))^2 = r^2$$

also $d(p_r, q_r)w(p_r) = d(p_s, q')w(p_s)$ we have:

$$w(p_s)\sqrt{(x(a) - X(q'))^2 + (y(a) - Y(q'))^2} =$$

$$w(p_r)[y(p_r) - Y(q')]$$

The two previous equations can be easily solve as a *system of linear equations* in O(1) time. It is easy to see that the l is $y = Y(q')$.

B computation procedure of case (ii.iv) in the third Algorithm

Given four points $p_r, p_s, p_t, p_u \in P$, we want to find, whether there is a line l such that $d(p_r, q')w(p_r) = d(p_s, q'')w(p_s)$ or not, where q' and q'' are intersection points of l with $\mathcal{C}^e(p_r, p_t)$ and $\mathcal{C}^e(p_s, p_u)$ respectively. For each point $p = (x, y)$, we add a height $z = d(p_r, q')w(p_r)$. by defining height $z = d(p_r, q')w(p_r)$ for each point (x, y) on $\mathcal{C}^e(p_r, p_t)$, we map each point on the three dimension base space so we get an ellipsoid $\mathcal{E}(p_r, p_t)$ in space, we do the same for the points on

$\mathcal{C}^e(p_s, p_u)$ and get an ellipsoid $\mathcal{E}(p_s, p_u)$. Then, we project $\mathcal{E}(p_r, p_t)$ and $\mathcal{E}(p_s, p_u)$, on the $y-z$-plane and name the made shapes $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively. Then it is easy to see, we can set the line l equal to $y = y_0$ iff S_1 and S_2 have an intersection with y-coordinate y_0. (see fig. 9)

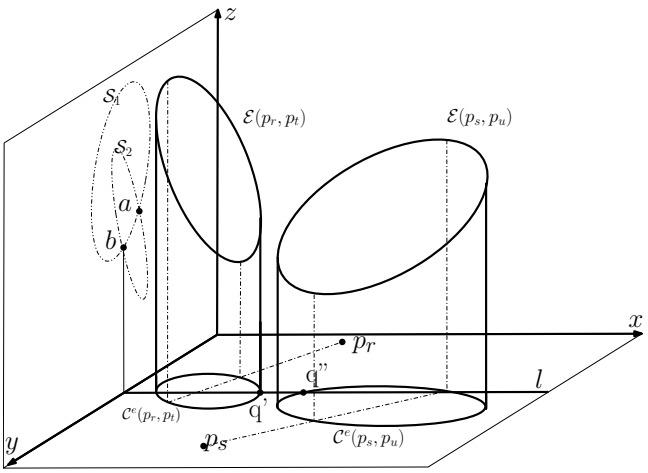

Figure 9: Equality circles for case (ii.iv)