Hitting a Set of Line Segments with One or Two Discrete Centers

Xiaozhou He* Zhihui Liu† Bing Su‡ Yinfeng Xu§ Feifeng Zheng¶ Binhai Zhu‖

Abstract

Given the scheduling model of bike-sharing, we consider the problem of hitting a set of n axis-parallel line segments in $\mathbb{R}^2$ by a square (and two squares) whose center(s) must lie on some line segment(s) such that the (maximum) edge length of the square(s) is minimized. Under a different model, we also consider the cases when one needs to compute one (and two) centers on some edge(s) of a tree of size m, where n labeled segments must be hit, and the objective is to minimize the maximum path length from the labeled segments to the nearer center(s). We give three linear-time algorithms and an $O(n^2 \log n)$ algorithm for the four problems in consideration.

1 Introduction

In recent years, the (private) bike-sharing business are booming in China (and in Singapore). To use a shared-bike, a user can use his/her smartphone to scan and unlock the bike. A small amount of fee, about US$0.16 currently, is charged for any use/transaction during that day. It is estimated that there are at least 30 million such transactions in major cities of China alone. Different from the traditional public bike-sharing services, wherein a user must return the bike to specified bike racks at fixed locations, in this bike-sharing service a user can lock and drop a bike anywhere after finishing using it. Of course, a lot of these bikes are dropped on some streets typically near bus/subway stations. In fact, right before and after rush hours, it is not uncommon to notice hundreds of bikes near some major subway stations in big cities like Beijing and Shanghai. This also holds when there is a major event near some site, like an open music show.

*Business School, Sichuan University, Chengdu, Sichuan, China, xiaozhouhe126@qq.com

†School of Computer Science and Technology, Shandong Technology and Business University, Yantai, Shandong, China, dane.zhihui.liu@gmail.com

‡School of Economics and Management, Xi'an Technological University, Xi'an, China, subing684@sohu.com

§School of Management, Xi'an Jiaotong University, Xi'an, China, yfxu@mail.xjtu.edu.cn

¶Glorious Sun School of Business and Management, Donghua University, Shanghai, China, ffzheng@dhu.edu.cn

‖Gianforte School of Computing, Montana State University, Bozeman, MT, 59717, USA, bhz@montana.edu

For the bike-sharing company, the objective is certainly to maximize the profit (i.e., the number of use of the bikes) and minimize the cost (i.e., collecting the scattered bikes quickly, and manually, to re-distribute them in bulk). Our research is motivated by this: given a set of roads (segments) scattered with shared-bikes, distribute these bikes in bulk from a center (or several centers) and transport them to these streets. Hence the problem is to find a center (resp. several centers) on these roads as the stations to store the bikes so that the distance to the farthest target road from the nearest station is minimal. Note that these centers change when the target roads are changed.

In this paper, we give two model of the streets in the cities. One is the classic grid network that is widely used in the urban streets model. In this model, we describe all the n target roads as some axis-parallel line segments and we use the ℓ_∞-norm to measure the distance. Here we also consider a practical restriction: the center (station) is exactly on one line segment (road) for the convenience of storage and scheduling, and we only need to touch every line segment (target road) at any point (position) to manually distribute the bikes. Also, note that a ℓ_∞ circle is an axis-parallel square.

Thus, follow this model the one-hitting-square problem is to find the minimum axis-parallel square whose center is on a line segment, to hit all the line segments, such that the edge length of the square is minimized. Analogously, the two-hitting-square problem can be defined.

In addition to the grid networks, we also consider the geometric tree network, with size m, to model the streets. The target roads are n segments/edges on the tree and the distance between any two points on the tree is the shortest distance between them along the tree edges. In this case we consider the one-center and two-center problems such that the centers must lie on some tree edges and the maximum distance between the target segments to the nearer centers is minimized. We next review some previous works.

When the target are n points and the distance is ℓ_2, the corresponding one-center [9, 12], two-center [2, 6, 11] (and discrete two-center [1]) problems have been well studied. In fact, even under ℓ_∞, the two-center and three-center problem can be solved in $O(n)$ time [4, 7] and a variation of the discrete two-center problem (where the centers of the congruent axis-parallel squares must be on some input points and the area of the squares

is minimized) can be solved in $O(n \log^2 n)$ time [8]. (There are other variations of these problems, like the target to cover is a convex polygon. We refer the readers to [5] for the references.) The research which is the closest to this one is by Sadhu et al., where the problem is to cover/hit a set of line segments using one or two (congruent) axis-parallel squares with the smallest size (edge length) [10]. Linear time algorithms are given for these problems. Our problems can be considered as the discrete version of these problems, where the centers must line on some input segments. We give $O(n)$ and $O(n^2 \log n)$ algorithms respectively. On the tree model, little is known for the corresponding two-center problem when the target is a set of edges, though the one-center solution (for edges) can be adapted to some forklore algorithm on computing the diameter of a tree in linear time.

This paper is organized as follows: In Section 2, we present some definitions and formally describe the four problems. Then, in Section 3-4 we give details for our solutions for the four problems. We conclude the paper in Section 5.

2 Preliminaries

2.1 Notations and Definitions

Coordinates: For every point $p \in \mathbb{R}^2$, we use $x(p)$ and $y(p)$ to denote its x-coordinate and y-coordinate, respectively.

Endpoints: We use $L(l_i)$, $R(l_i)$, $T(l_i)$ and $B(l_i)$ to denote the left endpoint, right endpoint, top endpoint and bottom endpoint of the line segment l_i $(1 \leq i \leq n)$ where $x(L(l_i)) < x(R(l_i))$ and $y(T(l_i)) > y(B(l_i))$.

Remark: A horizontal line segment has only a left endpoint and a right endpoint with the same y-coordinate, this is similar for a vertical line segment.

Distance: **(I)** In the first model, we use $d_\infty(p, l_i)$ to denote the *distance* between a point $p \in \mathbb{R}^2$ and a line segment l_i $(1 \leq i \leq n)$, and it is defined to be the minimum ℓ_∞-distance between p and some point q on l_i, where $d_\infty(p, q) = \max\{|x(p) - x(q)|, |y(p) - y(q)|\}$ and $|x(p) - x(q)|, |y(p) - y(q)|$ are the horizontal and vertical components respectively (also denoted as $d_h(p, l_i)$ and $d_v(p, l_i)$ as shown red in Fig. 1, where $p = s'$).

(II) In the tree network T, the *distance* between two points p, q (denoted as $d(p, q)$) is the length of the (unique) path between p and q along tree edges. And we denote the *distance* between a point p on an edge of T and a target edge (line segment) l_i by $d(p, l_i)$, which is the minimum distance between p and any point in l_i; formally, $d(p, l_i) = \min_{q \in l_j} d(p, q)$. Hence, $d(p, l_i)$ must be the shortest distance between p and an endpoint of l_i. Also, we use $d(l_i, l_j)$ to denote the *distance* between two line segments l_i and l_j as $d(l_i, l_j) = \min_{p \in l_i, q \in l_j} d(p, q)$,

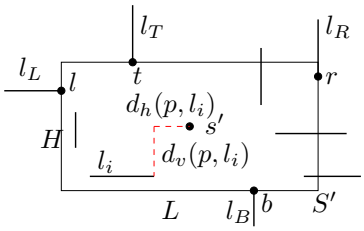

Figure 1: A minimum (unrestricted) rectangle S' hitting all segments.

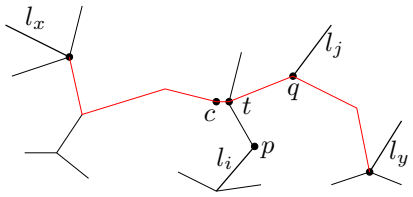

Figure 2: Distance on a tree T: $d(p, q) = d(p, t) + d(t, q)$, and $d(p, l_j) = d(l_i, l_j) = d(p, q)$.

which is the shortest distance between the endpoints of l_i, l_j. (See Fig. 2)

Diameter and radius: The *diameter* D and *radius* R of a set of line segments in a tree network T is defined as follows. For n line segments $U = \{l_1, l_2, ..., l_n\}$ on T, its *diameter* is the longest of the (shortest) path between any two line segments (in Fig. 2 the red path denotes the diameter). And the *diameter* D is also used to denote the length of this path, i.e., $D = \max_{1 \leq i,j \leq n} d(l_i, l_j)$. And naturally the radius is half of diameter $R = D/2$.

2.2 Problems

Throughout this paper, all squares are axis-parallel. Let S be an axis-parallel square with center s and its edge length (or *size*) is ℓ, then S can be defined as $S = \{p | d_\infty(p, s) \leq \ell/2\}$. We say a square S *hits* a segment l_i if there is a point $p \in S$ such that $d_\infty(p, l_i) = 0$. We now define the first two problems on finding one and two hitting squares.

Problem 1 (Discrete One-Hitting-Square): Given a set of axis-parallel line segments $U = \{l_1, l_2, ..., l_n\}$ in $\mathbb{R}^2$, the problem is to find a square S of minimum size such that S hits all the line segments in U and its center s is on a line segment in U. (See Fig. 3)

Problem 2 (Discrete Two-Hitting-Square): Given a set of axis-parallel line segments $U = \{l_1, l_2, ..., l_n\}$ in $\mathbb{R}^2$, the problem is to find two congruent squares S_1 and S_2 of minimum size such that each line segment in U is hit by at least one square and the center s_1 (resp. s_2) of S_1 (resp. S_2) is on a line segment in U.

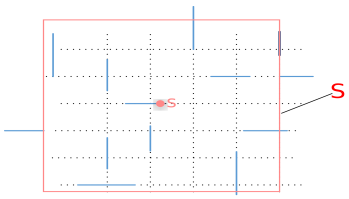

Figure 3: An example for the one hitting-square problem.

We next define the problems on a tree network. Note that in this case a hitting 'square' is virtual.

Problem 3 (One-tree-center): Given a set of edges (line segments) $U = \{l_1, l_2, ..., l_n\}$ on a geometric tree T in $\mathbb{R}^2$, the problem is to find a center c on an edge of T such that the maximum distance from a line segment in U to c is minimized. (See Fig. 2)

Problem 4 (Two-tree-center): Given a set of edges (line segments) $U = \{l_1, l_2, ..., l_n\}$ on a geometric tree T in $\mathbb{R}^2$, the problem is to find two centers c_1 and c_2 on some edges of T such that the maximum distance from a segment in U to the nearer center is minimized.

3 Solutions for Discrete Hitting-Square Problems

3.1 Discrete One-Hitting-Square

We first compute the minimum axis-parallel rectangle S' (with no restriction on its center s') such that all the line segments are hit by S' as in [10]. We then adjust this rectangle by moving s' to s and expanding the edge length to obtain the required square S whose center s must lie on a line segment in U.

To obtain the rectangle S', we present the definition of *boundary line segments* and *boundary points* at first. As shown in Fig. 1, we define four *boundary line segments*: the leftmost segment l_L, the rightmost segment l_R, the topmost segment l_T and the bottommost segment l_B to be the line segments that have *boundary points* l, r, t and b at one of their endpoints, respectively, where l, r, t and b are defined as follows:

$$l = \min_{\forall l_i \in U} x(R(l_i)),\ r = \max_{\forall l_i \in U} x(L(l_i))$$
$$t = \max_{\forall l_i \in U} y(B(l_i)),\ b = \min_{\forall l_i \in U} y(T(l_i))$$

Remark: If a boundary line segment is parallel to the boundary, then any point on it can be recognized as the boundary point since it is fine to hit any point on this line segment.

And we can then construct the rectangle S' by computing its four sides. This is based on the fact that S' hits all the line segments is equivalent to hitting the four boundary line segments l_L, l_R, l_T and l_B.

Now, we focus on how to adjust the rectangle to obtain the desired square. For every line segment l_i, we compute the distance $d_\infty(s', l_i)$, and record the vertical and horizontal components of the distance. Then we compute the expanding lengths γ_i's, i.e., the length that the longer side of the rectangle must be expanded into a square which hits all the segments and whose center lies on l_i. Finally, we choose the minimum γ^* to obtain the target square S.

Without loss of generality, we assume that the horizontal and vertical edge length of the rectangle S' are L and H $(L > H)$ respectively, as shown in Fig. 1.

Lemma 1 *To obtain a feasible square $S(i)$ which hits all the segments in U and whose center lies on some segment l_i, we need to expand the (horizontal) edge length of the minimum hitting rectangle S' by at least γ_i, where*

$$\gamma_i = 2 * \max\{\max\{d_v - \frac{L-H}{2}, 0\}, d_h\}, \qquad (1)$$

and $d_h = d_h(s', l_i)$, $d_v = d_v(s', l_i)$ are the horizontal and vertical components of $d_\infty(s', l_i)$ respectively, with s' being the center of S'.

Proof. As we need to move the center s' of S' horizontally by a value d_h to obtain $S(i)$, the edge length of $S(i)$ would be expanded by at least $2d_h$. (In this proof, imagine that S' is expanded smoothly in both directions, at the same pace.) At the vertical direction, the height of $S(i)$ would not be influenced by d_v if $d_v < \frac{L-H}{2}$. This is because after an expansion by $2d_v$ the height of S' is still shorter than the length. If $d_v > \frac{L-H}{2}$, the edge length of S' would have to be expanded by at least $2(d_v - \frac{L-H}{2})$ to have a feasible $S(i)$ (due to the vertical move of s'). Hence, $\gamma_i = 2 * \max\{\max\{d_v - \frac{L-H}{2}, 0\}, d_h\}$. □

Theorem 2 *The edge length of the smallest discrete hitting square S is $L + \gamma^*$, where $\gamma^* = \min_{\forall l_i \in U}\{\gamma_i\}$; moreover, S can be computed in $O(n)$ time.*

Proof. We first compute γ^*. Then, S can be computed from S' by finding the segment l^* which is γ^* distance away from s', i.e., $d_\infty(s', l^*) = \gamma^*/2$. The point s on l^* realizing $d_\infty(s', l^*) = d_\infty(s', s) = \gamma^*/2$ is the center of S, and the edge length of S is $L + \gamma^*$. □

3.2 Discrete Two-Hitting-Square

It seems hard to solve this 'two-hitting-square' version in the same way because simultaneously moving the two centers of the two hitting rectangles (presumably constructed as in [10]) to the destination is hard, and a brute-force method trying all the $O(n^2)$ combinations of the locations of the two centers s_1, s_2 (which must lie on some segments) would result in a high running time. Hence we use a different method. We try to fix one target square S_1 and then find the other square S_2. And

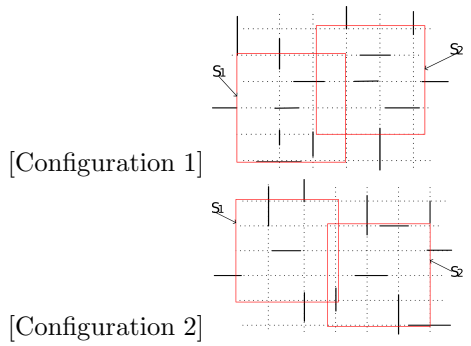

Figure 4: Two configurations for the two centers.

in order to fix S_1, it is natural to consider the corresponding decision problem and then obtain the optimal solution.

For the decision problem, the question is to determine whether there are two congruent (axis-parallel) squares $S_1(\alpha)$ and $S_2(\alpha)$ such that every line segments in U is hit by at least one of them, the center $s_1(\alpha)$ (resp. $s_2(\alpha)$) is on some line segment in U, and the edge length of $S_1(\alpha)$ and $S_2(\alpha)$ is 2α.

As the previous subsection, we find the smallest axis-parallel rectangle S' that hits all the segments in U. We assume that we also have the four *boundary line segments* and the four *boundary points*. Besides, we also assume that the two edge lengths of S' satisfy $L > H$. Then, there are two configurations of $S_1(\alpha)$ and $S_2(\alpha)$ similar to [10], see Fig. 4. Here we only discuss the configuration that $S_1(\alpha)$ (resp. $S_2(\alpha)$) hits l and b (resp. r and t), and the other configuration can be handled symmetrically.

We know that the coordinates of the center $s_1(\alpha)$ must satisfy $x(s_1(\alpha)) \leq x(l)+\alpha$ and $y(s_1(\alpha)) \leq y(b)+\alpha$ since the edge length of $S_1(\alpha)$ is 2α. Then, for every line segment $l_i (1 \leq i \leq n)$, we determine whether $s_1(\alpha)$ can lie on it (to constitute a feasible solution) following Observation 1.

Observation 1 *If $s_1(\alpha)$ is on a horizontal segment l_i, then the left endpoint u of l_i satisfies that $x(s_1(\alpha)) \geq x(u)$. Similarly, if $s_1(\alpha)$ is on a vertical segment l_j, then the bottom endpoint v of l_j satisfies that $y(s_1(\alpha)) \geq y(v)$.*

In fact, Observation 1 implies that, when l_i is horizontal, we could locate $s_1(\alpha)$ on it such that **(A)** $x(s_1(\alpha)) = \min\{x(l)+\alpha, x(R(l_i))\}$ and $y(s_1(\alpha)) = y(l_i)$, where the y-coordinate of l_i is $y(l_i)$. Similarly, when l_j is vertical, we could locate $s_1(\alpha)$ on it such that **(B)** $y(s_1(\alpha)) = \min\{y(b)+\alpha, y(T(l_i))\}$ and $x(s_1(\alpha)) = x(l_i)$, where the x-coordinate of l_i is $x(l_i)$.

Then, the decision procedure is straightforward: we locate $s_1(\alpha)$ (consequently $S_1(\alpha)$) on a candidate segment l_i according to (A) and (B), and for all the segments not covered by $S_1(\alpha)$ we use Theorem 2 to decide whether they can be covered by $S_2(\alpha)$ in $O(n)$ time. As there are n candidate segments l_i's, the decision procedure takes $O(n^2)$ time.

For the optimization problem, notice that the optimal solution value α^* must be in the form $d_\infty(l_i, l_j)$ or $d_\infty(l_i, l_j)/2$ (the corresponding optimal squares have an edge length $2d_\infty(l_i, l_j)$ or $d_\infty(l_i, l_j)$ respectively). Hence we can compute and sort this list of distances in $O(n^2 \log n)$ time. Then, we just use the decision procedure to perform a binary search to find α^* in $O(n^2 \log(n^2)) = O(n^2 \log n)$ time. Consequently, $S_1 \leftarrow S_1(\alpha^*), S_2 \leftarrow S_2(\alpha^*)$.

Theorem 3 *The Discrete Two-Hitting-Square problem can be solved in $O(n^2 \log n)$ time.*

4 Solution for Hitting the Line Segments on a Tree

In this section, we consider the problems on hitting a set of n segments on a tree T. We assume that T contains m edges, with $m > n$. Here a segment l_i is *hit* by a center c on T if $d(c, l_i)$ is bounded from above by some value β. Our problems are to hit all target segments with either one or two centers such that the corresponding β is minimized (Fig. 2).

4.1 One-tree-center

We present the algorithm to find c in the following algorithm. This algorithm is adapted from a folklore algorithm on computing the diameter of a tree.

1. Arbitrarily choose a node r_1 in the tree T as the root and find the line segment l_x that is the farthest from r_1 by breadth-first-search on T. Let $d(r_1, l_x) = d(r_1, x)$, where x is an endpoint of l_x.

2. Find the farthest line segment l_y from x by breadth-first-search on T. Let $d(x, l_y) = d(x, y)$, where y is an endpoint of l_y.

3. Compute the path between x and y as the diameter. The center c is the midpoint on the path between x and y (e.g., l_x and l_y).

Theorem 4 *The One-Tree-Center problem can be solved in $O(m+n)$ time; in fact, the optimal center c is just the midpoint of the diameter D; formally, $D = d(l_x, l_y) = \max_{1 \leq i,j \leq n} d(l_i, l_j)$ and c is on the path between l_x and l_y such that $d(c, l_x) = d(c, l_y)$.*

Proof. The correctness can be proved by contradiction. The details will be given in the full paper. The running time of the algorithm is obviously $O(m+n)$ as the main cost is two runs of the breadth-first-search algorithm [3]. □

4.2 Two-Tree-Center

In this problem, the objective is to find two centers c_1 and c_2 on the tree T such that

$$f \stackrel{def}{=} \max_{l_i \in U} \min\{d(c_1, l_i), d(c_2, l_i)\}$$

is minimized for any line segment l_i on the tree T.

To make our analysis more clean, we initially take c as a virtual root of the tree T and then perform some preprocessing, i.e., remove all the subtrees that do not contain target line segments and denote the position of every line segment by its endpoint (node) that is closer to c. Thus, every leaf node is the endpoint of a line segment in the transformed tree (we still call it T), and we abuse the terminology by calling these line segments as *leaves*.

For the sake of brevity, we use the notation f_1 (resp. f_2) to denote the distance between c_1 (resp. c_2) and the farthest line segment it hits. Thus, $f = \max\{f_1, f_2\}$.

In this subsection, we propose the algorithm first and sketch its correctness a bit later. We first implement the same algorithm as we did in the last subsection to obtain $l_x, l_y \in U$ and c. (Recall that $d(l_x, l_y)$ gives the diameter of the segments in U on T.) Then, we discuss the next steps in the following two cases:

(1) c is not a node of T, i.e., c is between two adjacent nodes in T. Cut T into two parts T' and T'' at c such that l_x, l_y are contained in T', T'' respectively. We can find the center c' (resp. c'') of T' (resp. T'') as done in Section 4.1. In Section 4.2.1 we give details to show that c' and c'' are just the two centers c_1 and c_2 of T' and T'', respectively, and $f = \max\{d(c', l_x), d(c'', l_y)\}$.

(2) c is exactly a node of T. In this case, there are two subcases to be discussed:

(2.1) There are exactly two subtrees (also denoted by T' and T'') of c: One contains l_x while the other contains l_y. Without loss of generality, it is assumed that c is contained in T' but not in T''. We can also compute c' and c'' similar to (1) and they are also the two centers of T', T''. And in this case it can be computed that $f = D/4 = d(l_x, l_y)/4$.

(2.2) There are more than two subtrees of c, denoted by $T^1, T^2, ..., T^k$, respectively. (Suppose c does not belong to any subtree.) Let the two subtrees that contains l_x and l_y be T^1 and T^2 respectively. Compute the centers c^1, c^2, c^{-1}, c^{-2} and radii R^1, R^2, R^{-1}, R^{-2} of T^1, T^2, T^{-1}, T^{-2}, where $T^{-1} = T^2 \cup T^3 \cup ... \cup T^k \cup c$ and $T^{-2} = T^1 \cup T^3 \cup ... \cup T^k \cup c$. In this case, $f = \min\{\max\{R^1, R^{-1}\}, \max\{R^2, R^{-2}\}\}$. We obtain c_1 and c_2 respectively as c^1 and c^{-1}, if $\max\{R^1, R^{-1}\} \leq \max\{R^2, R^{-2}\}$; and vice versa. In summary we have the following theorem.

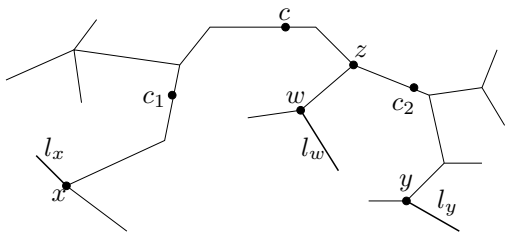

Figure 5: Illustration for the proof of Lemma 9, under the assumption that l_w can be hit by c_1 but cannot be hit c_2.

Theorem 5 *The Two-Tree-Center problem can be solved in $O(m + n)$ time.*

We next give some details for the above theorem, due to space constraints, we leave out some details for the final version of this paper.

4.2.1 Case 1. c is not a node of the tree T

For this case, we have the following properties which are intuitively obvious. Due to space limit, the proofs are omitted in this version.

Observation 2 *l_x and l_y must be two leaf nodes of T and are the farthest nodes from c.*

Lemma 6 *The two centers c_1 and c_2 of T must be in T' and T'' respectively.*

Observation 3 *l_x and l_y in T are hit by c_1 and c_2 respectively.*

Lemma 7 *l_x and l_y are the farthest line segments hit by c_1 and c_2 respectively, i.e., $d(c_1, l_x) = f_1$ and $d(c_2, l_y) = f_2$.*

Lemma 8 *c_1 (resp. c_2) is on the path between c and l_x (resp. l_y).*

Lemma 9 *In an optimal solution, even if there is a line segment l_w in T'' which is hit by c_1, i.e., $d(c_1, l_w) \leq f_1$, we can make a swap to use c_2 to hit it without making the solution worse. Similarly, even if there is a line segment l_v in T' which is hit by c_2, we can make a swap to hit l_v with c_1.*

Proof. Due to space limit, we only give a sketch of the proof, see Fig. 5. Assume that l_w in T'' is hit by c_1, i.e., $d(c_1, l_w) \leq f_1$, but cannot be hit by c_2, i.e., $d(c_2, l_w) > f_2$. We have $f = f_2 \geq f_1$, but we can show $f_1 > f_2$ to lead the needed contradiction. □

Corollary 1 *In an optimal solution of the Two-Tree-Center problem, c_1 hits all the line segments in T' and c_2 hits all the line segments in T''. Thus we can find c_1 (resp. c_2) by solving the One-Tree-Center problem on T' (resp. T'').*

4.2.2 Case 2. c is a node of the tree T

(2.1) *There are exactly two subtrees T' and T'' of c.* It is easy to see that c_1, c_2 are in T', T'' and hit l_x, l_y respectively, similar to Lemma 6 and Observation 3. Moreover, at least one of c_1 and c_2 hits l_x and c (or, l_y and c) simultaneously. (Otherwise the solution is not optimal.) Assume that c_1 hits both l_x and c, then $f_1 = d(l_x, l_y)/4 = D/4$ and f_2 cannot be greater, because l_x and l_y are the two line segments farthest from c. Hence, $f = \max\{f_1, f_2\} = \max\{d(c_1, l_x), d(c_2, l_y)\} = d(c_1, l_x) = D/4 = d(l_x, l_y)/4$.

(2.2) *There are more than two subtrees of c: $T^1, T^2, ..., T^k$.* Assume that l_x and l_y are in T^1 and T^2, respectively. We first claim that c_1 and c_2 must be in T^1 and T^2, respectively. (Otherwise, one of T^1 and T^2, say it is T^1, does not contain any center; thus $f = \max\{d(c_1, l_x), d(c_2, l_x)\} > d(c, l_x) = D/2$ which is even worse than the corresponding one-center solution. A contradiction.)

In this case we can also prove that $d(c_1, l_x) = f_1$, $d(c_2, l_y) = f_2$, and c_1 (resp. c_2) is on the path between c and l_x (resp. l_y) similar to Lemma 7 and Lemma 8. Thus we can also obtain the conclusion that all the line segments in T^1 (resp. T^2) are hit by c_1 and c_2, respectively, as in Lemma 9 and Corollary 1. Now we only need to consider the line segments in $T^3, ..., T^k$: Assume that the tree containing the farthest line segment from c other than T^1 and T^2 is T^3, and suppose that c_1 hits all the line segments in T^3. When we compute f_1 for c_1 to hit all the line segments in T^1 and T^3, it is obvious that all the line segments in $T^4, ..., T^k$ can also be hit by c_1 without increasing f_1. That is to say, the line segments in $T^1, T^3, ..., T^k$ are all hit by c_1. Similarly, if all the line segments in T^3 are hit by c_2, then all the line segments in $T^2, T^3, ..., T^k$ are hit by c_2. Hence we obtain the conclusion that either **(a)** c_1 hits all the line segments in T^1 and c_2 hits all the line segments in $T^2, T^3, ..., T^k$, or **(b)** c_1 hits all the line segments in $T^1, T^3, ..., T^k$ and c_2 hits all the line segments in T^2. Thus, $f = \min\{\max\{R^1, R^{-1}\}, \max\{R^2, R^{-2}\}\}$, and c_1 and c_2 can be computed accordingly. This concludes the correctness proof of Case 2.

5 Concluding Remarks

An extension of this research is to use a more realistic model, i.e., an irregular grid network (a grid network with some edges randomly deleted, e.g., something similar to a wall graph). It seems to take some effort to solve the discrete two-center problem in roughly $O(m^2)$ time or even better, where m is size of the network and there are $n (n < m)$ streets/segments to cover.

Acknowledgments

This research is partially supported by NNSF of China under project 61628207. XH is supported by China Scholarship Council under program 201706240214 and by the Fundamental Research Funds for the Central Universities under project 2012017yjsy219. ZL is supported by a Shandong Government Scholarship.

References

[1] P. Agarwal, M. Sharir and E. Welzl. The discrete 2-center problem, *Discrete and Computational Geometry*, 20(3):287-305, 1998.

[2] T. Chan. More planar two-center algorithms, *Computational Geometry: Theory and Applications*, 13(3):189-198, 1999.

[3] T. Cormen, C. Leiserson, R. Rivest and C. Stein. *Introduction to Algorithms*, second edition, MIT Press, 2001.

[4] Z. Drezner. On the rectangular p-center problem, *Naval Research Logistics*, 34(2):229-234, 1987.

[5] H. Du and Y. Xu. An approximation algorithm for k-center problem on a convex polygon, *J. Combinatorial Optimization*, 27(3):504-518, 2014.

[6] D. Eppstein. Faster construction of planar two-centers, In *Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*, pages 131-138, 1997.

[7] M. Hoffman. A simple linear algorithm for computing rectilinear 3-centers, *Computational Geometry: Theory and Applications*, 31(3):150-165, 2005.

[8] M. Katz, K. Kedem and M. Segal. Discrete rectilinear 2-center problem, *Computational Geometry: Theory and Applications*, 15(4):203-214, 2000.

[9] N. Megiddo. Linear-time algorithms for linear programming in R^3 and related problems, *SIAM J. Computing*, 12(4):759-776, 1983.

[10] S. Sadhu, S. Roy, S. Nandy and S. Roy. Optimal covering and hitting of line segments by two axis-parallel squares, In *Proc. 23rd International Computing and Combinatorics Conference (COCOON'17)*, LNCS 10392, pages 459-468, 2017.

[11] M. Sharir. A near-linear algorithm for the planar 2-center problem, *Discrete and Computational Geometry*, 18(2):125-134, 1997.

[12] E. Welzl. Smallest enclosing disks (balls and ellipsoids), In *New Results and New Trends in Computer Science* (Ed. H. Maurer), LNCS 555, pages 359-370, 1991.