Theoretical Computer Science 769 (2019) 63–74

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Linear time algorithm to cover and hit a set of line segments optimally by two axis-parallel squares ☆

Sanjib Sadhu [a,*], Sasanka Roy [b], Subhas C. Nandy [b], Suchismita Roy [a]

[a] Dept. of Computer Science and Engineering, National Institute of Technology Durgapur, India
[b] Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India

ARTICLE INFO

Article history:
Received 24 January 2018
Received in revised form 20 August 2018
Accepted 15 October 2018
Available online 17 October 2018
Communicated by M.J. Golin

Keywords:
Two-center problem
Covering line segments by squares
Hitting line segments by squares
Two pass algorithm
Computational geometry

ABSTRACT

This paper discusses the problem of covering and hitting a set of line segments $\mathcal{L}$ in $\mathbb{R}^2$ by a pair of axis-parallel congruent squares of minimum size. We also discuss the restricted version of covering, where each line segment in $\mathcal{L}$ is to be covered completely by at least one square. The proposed algorithms assume that the input segments are given in a read-only array. For each of these problems (i.e. *covering*, *hitting* and *restricted covering* problems), our proposed algorithm reports the optimum result by executing only two passes of reading the input array sequentially. All these algorithms need only $O(1)$ extra space. The solution of these problems also give a $\sqrt{2}$ approximation for covering and hitting those line segments $\mathcal{L}$ by two congruent disks of minimum radius with same computational complexity.

1. Introduction

Covering a point set by squares/disks has drawn interest to the researchers due to its applications in sensor network. Covering a given point set by k congruent disks of minimum radius, known as k-center problem, is NP-Hard [18]. For $k = 2$, this problem is referred to as the *two-center problem* [7,11,12,14,15,24].

A line segment ℓ_i is said to be covered (resp. hit) by two squares if every point (resp. at least one point) of ℓ_i lies inside one or both of the squares. For a given set $\mathcal{L}$ of line segments, the objective is to find two axis-parallel congruent squares such that each line segment in $\mathcal{L}$ is covered (resp. hit) by the union of these two squares, and the size of the squares is as small as possible. There are mainly two variations of the covering problem: standard version and discrete version. In discrete version, the center of the squares must be on some specified points, whereas there are no such restriction in standard version. In this paper, we focus our study on the standard version of covering and hitting a set $\mathcal{L}$ of line segments in $\mathbb{R}^2$ by two axis-parallel congruent squares of minimum size.

As an application, consider a sensor network, where each mobile sensor is moving along different line segment. The objective is to place two base stations of minimum transmission range so that each of mobile sensors are always (resp. intermittently) connected to any of the base stations. This problem is exactly same as to cover (resp. hit) the line segments by two congruent disks (in our case axis-parallel congruent squares) of minimum radius.

☆ A preliminary version of this paper appeared in COCOON 2017, pages 457–468.

* Corresponding author.
E-mail addresses: sanjib.sadhu@cse.nitdgp.ac.in (S. Sadhu), sasanka@isical.ac.in (S. Roy), nandysc@isical.ac.in (S.C. Nandy), suchismita.roy@cse.nitdgp.ac.in (S. Roy).

https://doi.org/10.1016/j.tcs.2018.10.013
0304-3975/

Most of the works on the *two-center problem* deal with covering a given point set. Kim and Shin [17] provided an optimal solution for the *two-center problem* of a convex polygon where the covering objects are two disks. As mentioned in [17], the major differences between the *two-center problem for a convex polygon P* and the *two-center problem for a point set S* are (i) points covered by the two disks in the former problem are *in convex positions* (instead of arbitrary positions), and (ii) the union of two disks should also cover the edges of the polygon P. The feature (i) indicates the problem may be easier than the standard two-center problem for points, but feature (ii) says that it might be more difficult.

Related Works: In the context of sensor network, *geometric k-covering problem* has a long history, where the objective is to cover a given region by k (≥ 1) disks. If $k = 1$ and the objective is to place the base-station on the boundary of a convex region $\mathcal{C}$ to cover the entire inside of $\mathcal{C}$ is studied in [22], and an algorithm is proposed with time linear in the number of vertices of $\mathcal{C}$. The same paper studies the problem with $k = 2$ (i.e., the 2-covering problem) where both the base-stations are placed on the same edge of $\mathcal{C}$, and shows that it is also linear time solvable. In [9], the aforesaid problem is generalized for arbitrary k, but the restriction of placing the base-stations on the same edge of $\mathcal{C}$ is retained. The proposed algorithm produces a $(1 + \epsilon)$-approximation result in $O((n + k)\log(n + k) + k\log(\lceil\frac{1}{\epsilon}\rceil))$ time. The same paper also studied the unrestricted 2-covering problem for a convex region $\mathcal{C}$, where the two base-stations can be placed anywhere on the boundary of $\mathcal{C}$. The time complexity of the proposed algorithm is $O(n^2)$. The unrestricted version of the k-covering problem was studied in [10], where a 1.8841-factor approximation algorithm is proposed. Basappa et al. [6] improved the approximation factor for very high values of k.

If the objective is to cover a set of points S, the best-known algorithm for the well-known one-center problem runs in $O(n)$ time, where $n = |S|$ [19]. Drenzer [8] covered a given point set S by two axis-parallel squares of minimum size in $O(n)$ time. Kim et al. [16] proposed an $O(n^2 \log n)$ time algorithm for covering a given point set S by two disjoint rectangles where one of the rectangles is axis-parallel and other one is of arbitrary orientation, and the area of the larger rectangle is minimized. Two congruent squares of minimum size covering all the points in S, where each one is of arbitrary orientation, can be computed in $O(n^4 \log n)$ time [1]. Almost linear time deterministic algorithm for the standard version of two-center problem for a point set S was first given by Sharir [24] that runs in $O(n \log^9 n)$ time. Eppstein [11] proposed a randomized algorithm for the same problem with expected time complexity $O(n \log^2 n)$. Later, Chan [7] improved the deterministic algorithm of Sharir to $O(n(\log n \log\log n)^2)$. Recently, Tan and Jiang [26] have proposed a new deterministic algorithm for this problem, which needs only $O(n \log^2 n)$ time. Hoffmann [13] solved the rectilinear three-center problem for a point set in $O(n)$ time. However none of the algorithms in [1,8,13] can handle the line segments.

The standard and discrete versions of the two-center problem for a convex polygon P was first solved by Kim and Shin [17] in $O(n \log^3 n \log\log n)$ and $O(n \log^2 n)$ time respectively. Becker [5] et al. has shown an $O(n^3)$ time heuristic algorithm for covering n axis-parallel rectangles by two axis-parallel rectangles of minimum total area. Recently, He et al. [27] has studied a special case of discrete version of hitting problem where the objective is to hit a set of n axis-parallel line segments with one (in one-center) and two (in two-center) minimum axis-parallel squares along with the constraint that the center(s) of square(s) must be on some input line segment. The one-center case can be solved in $O(n)$ time while the two-center case takes $O(n^2 \log n)$ time.

As an extension of the k-center problem for points, the problem of enclosing other geometric objects are also studied in the literature. If the object of interest is a convex polygon with n vertices, in $O(n)$ time it can be enclosed by a minimum area triangle [21] and by a minimum area parallelogram [23] of arbitrary orientation. Bhattacharya and Mukhopadhyay [4] showed that minimizing the perimeter of the triangle enclosing a convex polygon can also be done in linear time. The algorithm for computing a convex k-gon of minimum area that covers a convex n-gon can be computed in $O(n^2 \log k \log n)$ time [2]. Mitchell and Polishchuk [20] studied the perimeter minimization version of the problem, and proposed polynomial time algorithm. Alt et al. [3] studied an interesting version of packing problem, where a set P of convex polygons are given, and the objective is to find a rectangular suitcase S of minimum area such that each member of P can be accommodated in S with a suitable rotation and translation. They proposed an $O(n(2^{\alpha(n)}\alpha(k)\log k + \alpha(n)\log n))$ time algorithm for the problem, where k is the number of polygons to be packed, and n is the total number of vertices in those polygons. If the objective is to find a convex polygonal suitcase of minimum area, then the problem is NP-hard, and a PTAS is proposed in that paper. In [25], Schwartzkopf et al. studied an interesting variation of boundary covering problem of a convex polygon C using the annulus of a pair of homothetic rectangles R and r. Here the objective is to reduce the width of the annulus, or in other words the ratio λ of the side-lengths of R and r. In [25], it was shown that the lower bound of λ is 2. They also proposed a $O(\log^2 n)$ time algorithm for computing R and r with $\lambda = 2$, where the number of vertices in C is n.

Our Work: We propose in-place algorithms for covering and hitting n line segments in $\mathbb{R}^2$ by two axis-parallel congruent squares of minimum size. We also study the restricted version of the covering problem where each object needs to be completely covered by at least one of the reported squares. We assume that the input segments are given in a read-only array.

- The proposed algorithms for the *covering* problem, the *hitting* problem and the *restricted covering* problem, report the optimum result by executing only two passes of reading the input array sequentially using $O(1)$ work-space.
- The same algorithms work for covering/hitting a polygon, or a set of polygons by two axis-parallel congruent squares of minimum size.
- We show that the result of this algorithm can produce a solution for the problem of covering/hitting these line segments by two congruent disks of minimum radius with an approximation factor $\sqrt{2}$.

1.1. Notations and terminologies

Throughout this paper, unless otherwise stated a *square* is used to imply an axis-parallel square. We will use the following notations and definition.

Symbols used	**Meaning**
$\overline{pq}$ and $\|pq\|$	The line segment joining two points p and q, and its length
$x(p)$ (resp. $y(p)$)	x- (resp. y-) coordinate of the point p
$\|x(p) - x(q)\|$	*Horizontal distance* between a pair of points p and q
$\|y(p) - y(q)\|$	*Vertical distance* between a pair of points p and q
$s \in \overline{pq}$	The point s lies on the line segment $\overline{pq}$
$\square efgh$	An axis-parallel rectangle with vertices at e, f, g and h
$size\,(\mathcal{S})$	Size of square $\mathcal{S}$; it is the length of its one side
$p \in \mathcal{S}$	The point p lies on the area covered by the square $\mathcal{S}$
$LS(\mathcal{S})$, $RS(\mathcal{S})$	Left-side of square $\mathcal{S}$ and right-side of square $\mathcal{S}$
$TS(\mathcal{S})$, $BS(\mathcal{S})$	Top-side of square $\mathcal{S}$ and bottom-side of square $\mathcal{S}$

Definition 1. A square is said to be **anchored** with a vertex of a rectangle $\mathcal{R} = \square efgh$, if one of the corners of the square coincides with that vertex of $\mathcal{R}$.

2. Covering line segments by two congruent squares

LCOVER problem: Given a set $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_n\}$ of n line segments (possibly intersecting) in $\mathbb{R}^2$, the objective is to compute two congruent squares $\mathcal{S}_1$ and $\mathcal{S}_2$ of minimum size whose union covers all the members in $\mathcal{L}$.

In the first pass, a linear scan is performed among the objects in $\mathcal{L}$, and four points "a", "b", "c" and "d" are identified with minimum x-, maximum y-, maximum x- and minimum y-coordinate respectively among the end-points of the members in $\mathcal{L}$. This defines an axis-parallel rectangle $\mathcal{R} = \square efgh$ of minimum size that covers $\mathcal{L}$, where $a \in \overline{he}$, $b \in \overline{ef}$, $c \in \overline{fg}$ and $d \in \overline{gh}$. We use $L = |x(c) - x(a)|$ and $W = |y(b) - y(d)|$ as the length and width respectively of the rectangle $\mathcal{R}$, and we assume that $L \geq W$. We assume that $\mathcal{S}_1$ lies to the left of $\mathcal{S}_2$. The squares $\mathcal{S}_1$ and $\mathcal{S}_2$ may or may not overlap (see Fig. 1). We use $\sigma = size(\mathcal{S}_1) = size(\mathcal{S}_2)$.

Lemma 1. *(a) There exists an optimal solution of the problem where the left side of $\mathcal{S}_1$ ($LS(\mathcal{S}_1)$) and the right side of $\mathcal{S}_2$ ($RS(\mathcal{S}_2)$) pass through the points "a" and "c" respectively.*

(b) The top side (TS) of at least one of $\mathcal{S}_1$ and $\mathcal{S}_2$ pass through the point "b", and the bottom side (BS) of at least one of $\mathcal{S}_1$ and $\mathcal{S}_2$ pass through the point "d".

Proof. Since $\mathcal{S}_1$ and $\mathcal{S}_2$ cover $\mathcal{L}$ optimally, we have $a \in \mathcal{S}_1$ and $c \in \mathcal{S}_2$. If $a \notin LS(\mathcal{S}_1)$, then the square $\mathcal{S}_1$ can be translated horizontally towards right so that $LS(\mathcal{S}_1)$ passes through "a". Similarly, if $c \notin RS(\mathcal{S}_2)$, then the square $\mathcal{S}_2$ can be translated horizontally towards left so that $RS(\mathcal{S}_2)$ passes through "c". Observe that, any point $\alpha \in \mathcal{L}$ which was inside $\mathcal{S}_1$ (resp. $\mathcal{S}_2$)

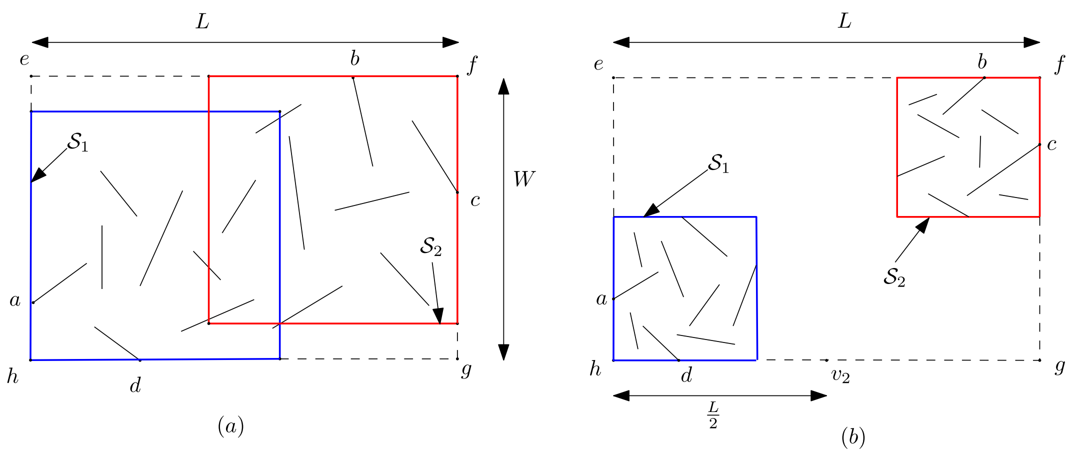

Fig. 1. Squares $\mathcal{S}_1$ and $\mathcal{S}_2$ are (a) overlapping, (b) disjoint.

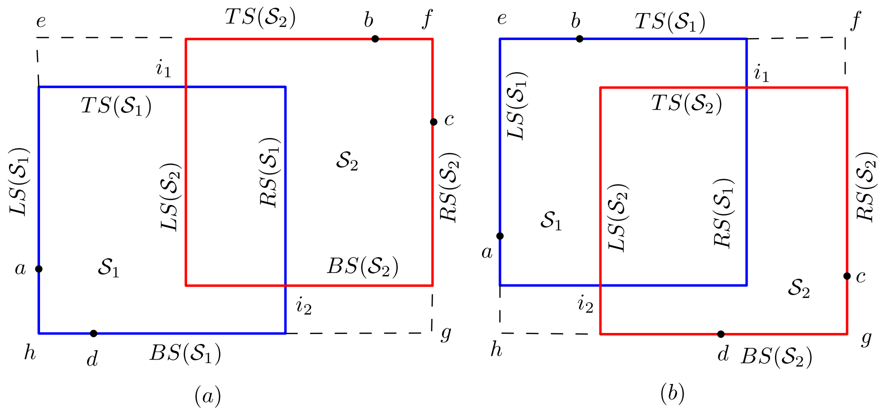

Fig. 2. (a) **Configuration 1** and (b) **Configuration 2** of squares $\mathcal{S}_1$ and $\mathcal{S}_2$.

before the translations, remains inside $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) after the translations also. To prove the next part, we need to consider the following possibilities:

(i) $\mathbf{d} \in \boldsymbol{\mathcal{S}_1}$ and $\mathbf{b} \in \boldsymbol{\mathcal{S}_2}$: Here $\mathcal{S}_1$ is moved vertically up, and $\mathcal{S}_2$ is moved vertically down so that "d" lies on $BS(\mathcal{S}_1)$ and "b" lies on $TS(\mathcal{S}_2)$. As argued above, here also any point $\alpha \in \mathcal{L}$ which was inside $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) before the translations, remains inside $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) after the translations also.

(ii) $\mathbf{d} \in \boldsymbol{\mathcal{S}_2}$ and $\mathbf{b} \in \boldsymbol{\mathcal{S}_1}$: Here $\mathcal{S}_2$ is moved vertically up, and $\mathcal{S}_1$ is moved vertically down so that "b" lies on $TS(\mathcal{S}_1)$ and "d" lies on $BS(\mathcal{S}_2)$.

(iii) Both $\mathbf{b}, \mathbf{d} \in \boldsymbol{\mathcal{S}_1}$: In this case, for the optimality of the size of $\mathcal{S}_1$, $b \in TS(\mathcal{S}_1)$ and $d \in BS(\mathcal{S}_1)$. As the size of $\mathcal{S}_2$ is same as that of $\mathcal{S}_1$, we can align $BS(\mathcal{S}_2)$ with $BS(\mathcal{S}_1)$ and $TS(\mathcal{S}_2)$ with $TS(\mathcal{S}_1)$.

(iv) Both $\mathbf{b}, \mathbf{d} \in \boldsymbol{\mathcal{S}_2}$: This case is similar to case (iii).

Thus, the lemma follows. □

Thus in an optimal solution of the **LCOVER** problem, $a \in LS(\mathcal{S}_1)$ and $c \in RS(\mathcal{S}_2)$. We need to consider two possible configurations of an optimum solution (i) $b \in TS(\mathcal{S}_2)$ and $d \in BS(\mathcal{S}_1)$, and (ii) $b \in TS(\mathcal{S}_1)$ and $d \in BS(\mathcal{S}_2)$. These are named as **Configuration 1** and **Configuration 2** respectively (see Fig. 2). The following observation is a consequence of Lemma 1.

Observation 1. (a) If the optimal solution of LCOVER problem satisfies **Configuration 1**, then the bottom-left corner of $\mathcal{S}_1$ will be anchored at the point h, and the top-right corner of $\mathcal{S}_2$ will be anchored at the point f.

(b) If the optimal solution of LCOVER problem satisfies **Configuration 2**, then the top-left corner of $\mathcal{S}_1$ will be anchored at the point e, and the bottom-right corner of $\mathcal{S}_2$ will be anchored at the point g.

We consider each of the configurations separately, and compute the two axis-parallel congruent squares $\mathcal{S}_1$ and $\mathcal{S}_2$ of minimum size whose union covers the given set of line segments $\mathcal{L}$. If σ_1 and σ_2 are the sizes obtained for **Configuration 1** and **Configuration 2** respectively, then we report $\min(\sigma_1, \sigma_2)$.

Consider the rectangle $\mathcal{R} = \square efgh$ covering $\mathcal{L}$, and take six points k_1, k_2, k_3, k_4, v_1 and v_2 on the boundary of $\mathcal{R}$ satisfying $|k_1 f| = |ek_3| = |hk_4| = |k_2 g| = W$ and $|ev_1| = |hv_2| = \frac{L}{2}$ (see Fig. 3). Throughout the paper we assume h as the origin in the co-ordinate system, i.e. $h = (0, 0)$. The distance between a pair of points "a" and "b" in L_∞ norm is given by $d_\infty(a, b) = \max(|x(a) - x(b)|, |y(a) - y(b)|)$.

The Voronoi partitioning line λ_1 of the corners f and h of $\mathcal{R} = \square efgh$ with respect to the L_∞ norm is the polyline $k_1 z_1 z_2 k_4$, where the coordinates of its defining points are $k_1 = (L - W, W)$, $z_1 = (L/2, L/2)$, $z_2 = (L/2, W - L/2)$ and $k_4 = (W, 0)$ (see Fig. 3(a)). Similarly, the Voronoi partitioning line λ_2 of the corners e and g of $\mathcal{R} = \square efgh$ in L_∞ norm is the polyline $k_3 z_1 z_2 k_2$ where $k_3 = (W, W)$ and $k_2 = (L - W, 0)$ (see Fig. 3(b)). Note that, if $W \le \frac{L}{2}$, then the Voronoi partitioning lines λ_1 for the point pair (f, h), and λ_2 for the point pair (e, g) will be the same, i.e., $\lambda_1 = \lambda_2 = \overline{v_1 v_2}$, where $v_1 = (\frac{L}{2}, 0)$ and $v_2 = (\frac{L}{2}, W)$. The property of "Voronoi diagram" suggests the following observations.

Observation 2. (a) For **Configuration 1**, all the points p inside the polygonal region $ek_1 z_1 z_2 k_4 h$ satisfy $d_\infty(p, h) < d_\infty(p, f)$, and all points p inside the polygonal region $k_1 fgk_4 z_2 z_1$ satisfy $d_\infty(p, f) < d_\infty(p, h)$ (see Fig. 3(a)).

(b) Similarly for **Configuration 2**, all points p inside polygonal region $ek_3 z_1 z_2 k_2 h$, satisfy $d_\infty(p, e) < d_\infty(p, g)$, and all points p that lie inside the polygonal region $k_3 fgk_2 z_2 z_1$, satisfy $d_\infty(p, g) < d_\infty(p, e)$ (see Fig. 3(b)).

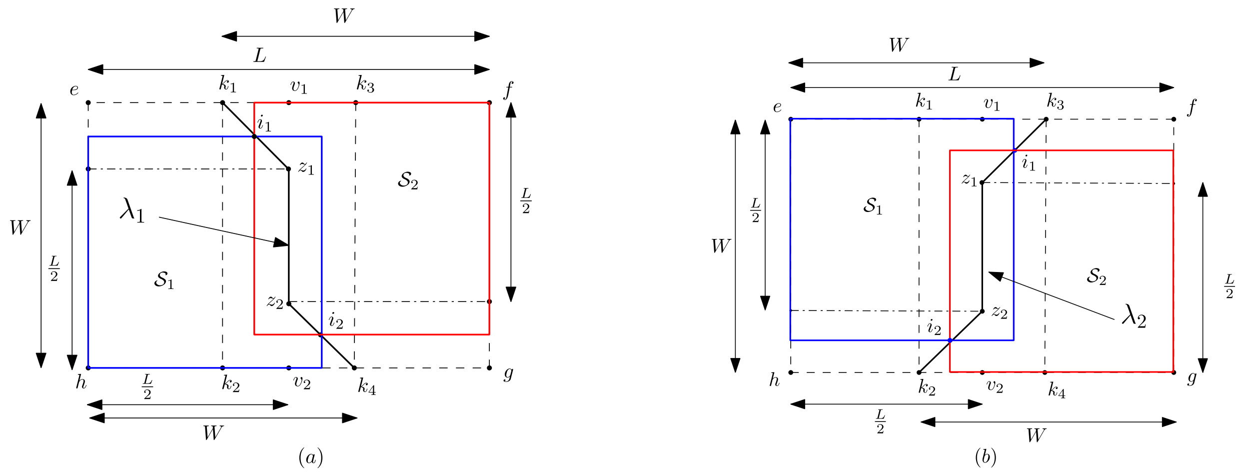

Fig. 3. Voronoi partitioning line (a) $\lambda_1 = k_1z_1z_2k_4$ of f and h in **Configuration 1** (b) $\lambda_2 = k_3z_1z_2k_2$ of e and g in **Configuration 2**.

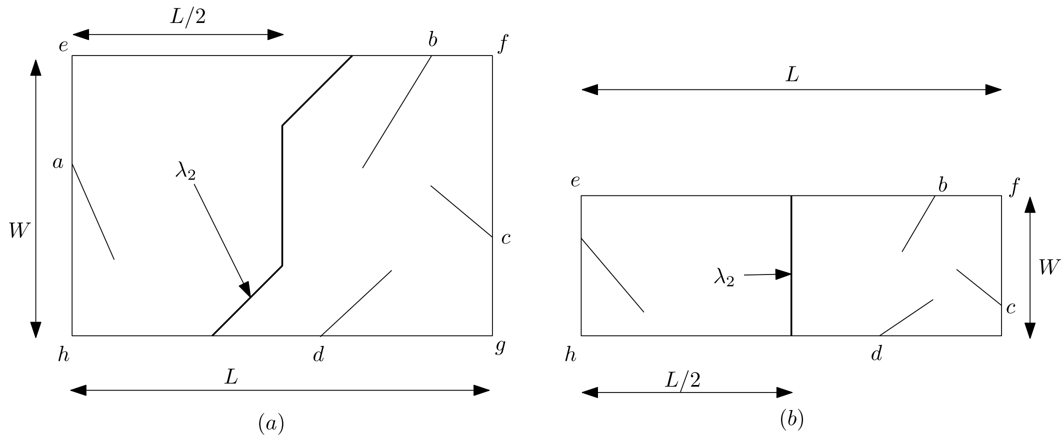

Fig. 4. The points "b" and "d" lie on the same side of λ_2 (a) $W > \frac{L}{2}$ (b) $W \leq \frac{L}{2}$.

Observation 3. If $\mathcal{S}_1$ and $\mathcal{S}_2$ intersect, then the points of intersection i_1 and i_2 will always lie on Voronoi partitioning line $\lambda_1 = k_1z_1z_2k_4$ (resp. $\lambda_2 = k_3z_1z_2k_2$) depending on whether $\mathcal{S}_1$ and $\mathcal{S}_2$ satisfy **Configuration 1** or **Configuration 2**.

Our algorithm consists of two passes. In each pass we sequentially read each element of the input array $\mathcal{L}$ exactly once. We consider $W > \frac{L}{2}$ only. The other case, i.e. $W \leq \frac{L}{2}$, can be handled in the similar way.

Pass-1: We compute the rectangle $\mathcal{R} = \square efgh$, and the Voronoi partitioning lines λ_1 and λ_2 (see Fig. 3) for handling **Configuration 1** and **Configuration 2**. We now discuss Pass 2 for **Configuration 1**. The same method works for **Configuration 2**. For both the configurations, the execution run simultaneously keeping $O(1)$ working storage.

Pass-2: λ_1 splits $\mathcal{R}$ into two disjoint parts, namely $\mathcal{R}_1 =$ region $ek_1z_1z_2k_4h$ and $\mathcal{R}_2 =$ region $fk_1z_1z_2k_4g$. We initialize $\sigma_1 = 0$, and read the elements in the input array $\mathcal{L}$ in sequential manner. For each element $\ell_i = [p_i, q_i]$, we identify its portion lying in one/both of $\mathcal{R}_1$ and $\mathcal{R}_2$. Now, considering Observations 1 and 2, we execute the following:

ℓ_i lies inside $\mathcal{R}_1$: Compute $\delta = \max(d_\infty(p_i, h), d_\infty(q_i, h))$.
ℓ_i lies inside $\mathcal{R}_2$: Compute $\delta = \max(d_\infty(p_i, f), d_\infty(q_i, f))$.
ℓ_i is intersected by λ_1: Let θ be the point of intersection of ℓ_i and λ_1, $p_i \in \mathcal{R}_1$ and $q_i \in \mathcal{R}_2$. Here, we compute $\delta = \max(d_\infty(p_i, h), d_\infty(\theta, h), d_\infty(q_i, f))$.

If $\delta > \sigma_1$, we update σ_1 with δ. Similarly, σ_2 is also computed in this pass considering the pair (e, g) and their partitioning line λ_2. Finally, $\min(\sigma_1, \sigma_2)$ is returned as the optimal size along with the centers of the squares $\mathcal{S}_1$ and $\mathcal{S}_2$.

Special Case: The points "b" and "d" lie on the same side of the Voronoi partitioning line in a configuration.

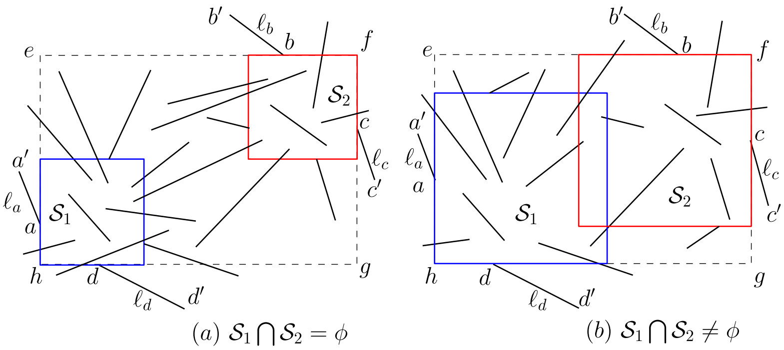

Fig. 5. Two axis-parallel congruent squares $\mathcal{S}_1$ and $\mathcal{S}_2$ hit line segments in $\mathcal{L}$.

Without loss of generality, we assume that the points "b" and "d" lie on the right side of λ_2 (see Fig. 4), i.e. $\mathcal{S}_1$ and $\mathcal{S}_2$ satisfy **Configuration 2**. Here $\mathcal{S}_2$ must cover both the points "b" and "d"; hence the size of $\mathcal{S}_2$ must be at least W.

Fact 1. The size of two congruent axis-parallel squares to cover an axis-parallel rectangle $\mathcal{R}$ of length L and width W $(W < L)$ will be $\max(W, \frac{L}{2})$.

Here, we have two possibilities:

(i) $\mathbf{W} > \frac{\mathbf{L}}{\mathbf{2}}$: Refer to Fig. 4(a). In this case, the size of $\mathcal{S}_2$ will be exactly W, and W will be the optimal size of both the squares $\mathcal{S}_1$ and $\mathcal{S}_2$ for this instance of **Configuration 2** since $\max(W, \frac{L}{2}) = W$ (see Fact 1). It needs to be noted that "b" and "d" may lie at the different side with respect to the other Voronoi partitioning line λ_1, and in such a case, the size of $\mathcal{S}_1$ and $\mathcal{S}_2$ (that satisfy **Configuration 1**) are to be computed following the aforesaid algorithm of **LCOVER problem** and $\max(size(\mathcal{S}_1), size(\mathcal{S}_2))$ (of **Configuration 1**) will be reported as the size of the congruent squares if this size is less than W.

(ii) $\mathbf{W} \leq \frac{\mathbf{L}}{\mathbf{2}}$: Refer to Fig. 4(b). In this case, $\lambda_1 = \lambda_2$ and the size of $\mathcal{S}_2$ will be at least W. So we anchor the squares $\mathcal{S}_1$ and $\mathcal{S}_2$ either at (e, g) or at (h, f) of the rectangle $\mathcal{R}$. We follow the algorithm of **LCOVER problem** and determine the optimal size of the congruent squares.

Thus, we have the following result:

Theorem 1. *Given a set of line segments $\mathcal{L}$ in $\mathbb{R}^2$ in an array, one can compute two axis-parallel congruent squares of minimum size whose union covers $\mathcal{L}$ by reading the input array only twice in sequential manner, and maintaining $O(1)$ extra work-space.*

Proof. The correctness of the algorithm follows from the facts that (i) we have only two configurations of the optimum solution (see Observation 1), (ii) in Configuration 1, for every point θ in the left-partition (resp. right-partition) $d_\infty(\theta, h) <$ (resp. $>$) $d_\infty(\theta, f)$ (similar observation holds in Configuration 2), and (iii) we are covering portions of the members in $\mathcal{L}$ in the left (resp. right) partition of λ_1 by $\mathcal{S}_1$ (resp. $\mathcal{S}_2$).

The time complexity follows from the fact that we scan the input array only twice in sequential manner, for each input element, we computed θ for both **Configuration 1** and **Configuration 2** in $O(1)$ time, and update σ_1 and σ_2 if needed. Finally we report $\min(\sigma_1, \sigma_2)$.

The extra space required for storing the variables e, f, g, h, λ_1, λ_2, σ_1, σ_2, δ is $O(1)$. □

3. Hitting line segments by two congruent squares

Definition 2. A geometric object Q is said to be *hit* by a square $\mathcal{S}$ if at least one point of Q lies inside (or on the boundary of) $\mathcal{S}$.

Line segment hitting (LHIT) problem: Given a set $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_n\}$ of n line segments in $\mathbb{R}^2$, compute two axis-parallel congruent squares $\mathcal{S}_1$ and $\mathcal{S}_2$ of minimum size whose union hits all the line segments in $\mathcal{L}$.

The squares $\mathcal{S}_1$ and $\mathcal{S}_2$ may or may not be disjoint (see Fig. 5). Without loss of generality, we assume that $\mathcal{S}_1$ lies to the left of $\mathcal{S}_2$. We now describe the algorithm for this **LHIT** problem.

For each line segment ℓ_i, we use $LP(\ell_i)$, $RP(\ell_i)$, $TP(\ell_i)$ and $BP(\ell_i)$ to denote its left end-point, right end-point, top end-point and bottom end-point using the relations $x(LP(\ell_i)) \leq x(RP(\ell_i))$ and $y(BP(\ell_i)) \leq y(TP(\ell_i))$. Now we compute

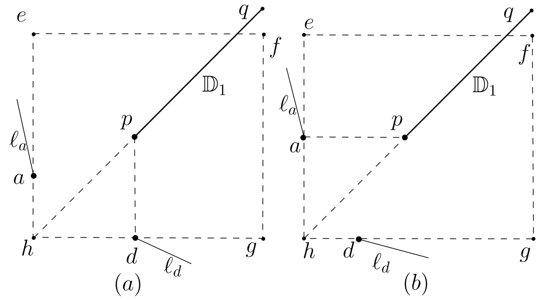

Fig. 6. $\mathbb{D}_1$ for $y(LP(\ell_a)) \geq y(RP(\ell_a))$ and $x(TP(\ell_d)) < x(BP(\ell_d))$.

four line segments ℓ_a, ℓ_b, ℓ_c, and $\ell_d \in \mathcal{L}$ such that one of their end-points "a", "b", "c" and "d", respectively satisfy the following

$$a = \min_{\forall \ell_i \in \mathcal{L}} x(RP(\ell_i)), \quad b = \max_{\forall \ell_i \in \mathcal{L}} y(BP(\ell_i))$$

$$c = \max_{\forall \ell_i \in \mathcal{L}} x(LP(\ell_i)), \quad d = \min_{\forall \ell_i \in \mathcal{L}} y(TP(\ell_i))$$

We denote the other end point of ℓ_a, ℓ_b, ℓ_c and ℓ_d by "a'", "b'", "c'" and "d'", respectively. The four points "a", "b", "c" and "d" define an axis-parallel rectangle $\mathcal{R} = \Box efgh$ of minimum size that hits all the members of $\mathcal{L}$ (as per Definition 2), where $a \in \overline{he}$, $b \in \overline{ef}$, $c \in \overline{fg}$ and $d \in \overline{gh}$ (see Fig. 5). We use $L = |x(c) - x(a)|$ and $W = |y(b) - y(d)|$ as the length and width of the rectangle $\mathcal{R}$, and assume $L \geq W$.

Lemma 2. *(a) The left side of $\mathcal{S}_1$ (resp. right side of $\mathcal{S}_2$) must not lie to the right of (resp. left of) the point "a" (resp. "c"), and (b) the top side (resp. bottom side) of both $\mathcal{S}_1$ and $\mathcal{S}_2$ cannot lie below (resp. above) the point "b" (resp. "d").*

Proof. If the left side of square $\mathcal{S}_1$ (where $\mathcal{S}_1$ lies to the left of $\mathcal{S}_2$) lies to the right of "a", then the line segment ℓ_a is not covered by any of the squares. Similarly, if the right side of square $\mathcal{S}_2$ lies to the left of "c", then the line segment ℓ_c is not covered by any of the squares. Similarly, if the top side (resp. bottom side) of both $\mathcal{S}_1$ and $\mathcal{S}_2$ lie below (resp. above) the point "b" (resp. "d"), then the line segment ℓ_b (resp. ℓ_d) is not covered by any of the squares. $\Box$

For the **LHIT** problem, we say $\mathcal{S}_1$ and $\mathcal{S}_2$ are in **Configuration 1**, if $\mathcal{S}_1$ hits both ℓ_a and ℓ_d, and $\mathcal{S}_2$ hits both ℓ_b and ℓ_c. Similarly, $\mathcal{S}_1$ and $\mathcal{S}_2$ are said to be in **Configuration 2**, if $\mathcal{S}_1$ hits both ℓ_a and ℓ_b, and $\mathcal{S}_2$ hits both ℓ_c and ℓ_d. Without loss of generality, we assume that $\mathcal{S}_1$ and $\mathcal{S}_2$ are in **Configuration 1**.

Definition 3. A square $\mathcal{S}$ "touches" a line segment ℓ (outside $\mathcal{S}$) if either (i) a corner of the $\mathcal{S}$ lies on the ℓ or (ii) an end point of ℓ lies on the boundary of $\mathcal{S}$.

We compute polyline $\mathbb{D}_1$ (resp. $\mathbb{D}_2$) which is the locus of the "top-right" corner (resp. the "bottom-left" corner) of a square $\mathcal{S}$ that *touches* both "ℓ_a" and "ℓ_d" (resp. "ℓ_b" and "ℓ_c"). We will term $\mathbb{D}_1$ and $\mathbb{D}_2$ as the "reference lines" for **Configuration 1**. Hence, the top-right corner of $\mathcal{S}_1$ (resp. bottom-left corner of $\mathcal{S}_2$) will lie on the "reference line" $\mathbb{D}_1$ (resp. $\mathbb{D}_2$).

Let $\mathbb{T}_1$ (resp. $\mathbb{T}_2$) be the line passing through h (resp. f) with slope 1.

Our algorithm consists of the following steps:

1 Compute of the reference lines $\mathbb{D}_1$ and $\mathbb{D}_2$.
2 For each line segment $\ell_i \in \mathcal{L}$, compute the size of the minimum square $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) required to hit ℓ_i, ℓ_a and ℓ_d (resp. ℓ_i, ℓ_b and ℓ_c), where the top-right (resp. bottom-left) corner of $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) lies on $\mathbb{D}_1$ (resp. $\mathbb{D}_2$).
3 Determine the pair $(\mathcal{S}_1, \mathcal{S}_2)$ that hit all the line segments in $\mathcal{L}$ and $\max(size(\mathcal{S}_1), size(\mathcal{S}_2))$ is minimized.

Computation of $\mathbb{D}_1$ and $\mathbb{D}_2$: The reference line $\mathbb{D}_1$ is computed based on the following four possible orientations of ℓ_a and ℓ_d

(i) $y(LP(\ell_a)) \geq y(RP(\ell_a))$ and $x(TP(\ell_d)) < x(BP(\ell_d))$: Here $\mathbb{D}_1$ is the segment $\overline{pq}$ on $\mathbb{T}_1$ where p is determined (i) by its x-coordinate i.e. $x(p) = x(d)$, if $|ha| < |hd|$ (see Fig. 6(a)), (ii) by its y-coordinate i.e. $y(p) = y(a)$, if $|ha| \geq |hd|$ (see Fig. 6(b)). The point q on $\mathbb{T}_1$ satisfy $x(q) = x(f)$.

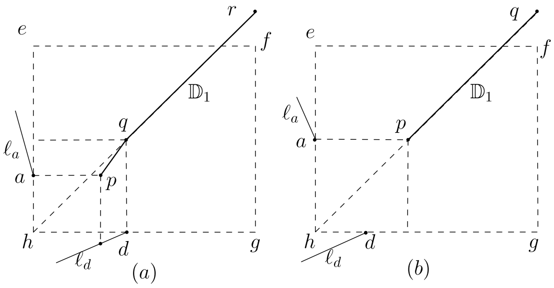

Fig. 7. $\mathbb{D}_1$ for $y(LP(\ell_a)) \geq y(RP(\ell_a))$ and $x(TP(\ell_d)) \geq x(BP(\ell_d))$.

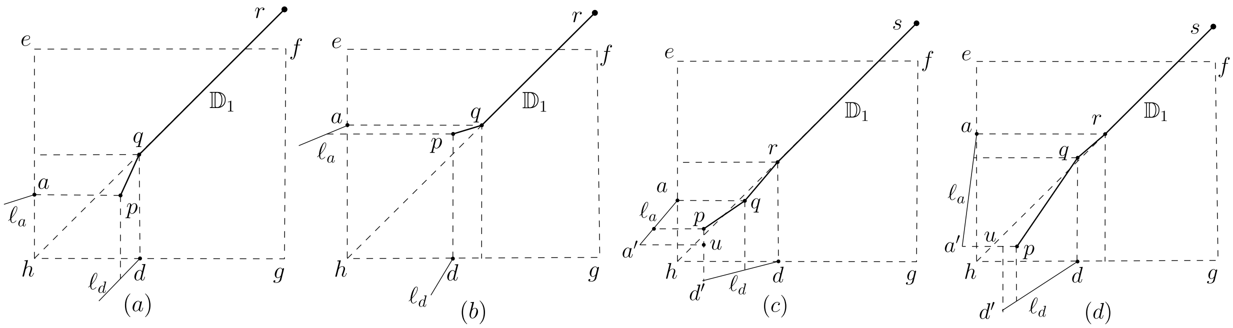

Fig. 8. $\mathbb{D}_1$ for $y(LP(\ell_a)) < y(RP(\ell_a))$ and $x(TP(\ell_d)) > x(BP(\ell_d))$.

(ii) $\boldsymbol{y(LP(\ell_a)) \geq y(RP(\ell_a))}$ and $\boldsymbol{x(TP(\ell_d)) \geq x(BP(\ell_d))}$: Here,
if $|ha| < |hd|$ (see Fig. 7(a)), then the reference line $\mathbb{D}_1$ is a polyline $\overline{pqr}$, where (i) $y(p) = y(a)$ and $x(p)$ satisfies $|x(p) - x(a)| =$ vertical distance of p from the line segment ℓ_d, (ii) the point q lies on $\mathbb{T}_1$ satisfying $x(q) = x(d)$ and (iii) the point r lies on $\mathbb{T}_1$ satisfying $x(r) = x(f)$.
If $|ha| \geq |hd|$ (see Fig. 7(b)), then the reference line $\mathbb{D}_1$ is a line segment $\overline{pq}$, where p, q lies on $\mathbb{T}_1$, and p satisfies $y(p) = y(a)$ and q satisfies $x(q) = x(f)$.

(iii) $\boldsymbol{y(LP(\ell_a)) < y(RP(\ell_a))}$ and $\boldsymbol{x(TP(\ell_d)) \leq x(BP(\ell_d))}$: This case is similar to case (ii), and we can compute the respective reference lines.

(iv) $\boldsymbol{y(LP(\ell_a)) < y(RP(\ell_a))}$ and $\boldsymbol{x(TP(\ell_d)) > x(BP(\ell_d))}$: There are two possible subcases:

(A) If ℓ_a and ℓ_d are parallel or intersect (after extension) at a point to the right of $\overline{he}$ (Fig. 8(a, b)), then the reference line $\mathbb{D}_1$ is a polyline $\overline{pqr}$, where **(a) if** $|\mathbf{ha}| < |\mathbf{hd}|$ (Fig. 8(a)), then (1) $y(p) = y(a)$ and $|x(p) - x(a)| =$ the vertical distance of p from ℓ_d, (2) the points q and r lie on $\mathbb{T}_1$ satisfying $x(q) = x(d)$ and $x(r) = x(f)$, **(b) if** $|\mathbf{ha}| > |\mathbf{hd}|$ (Fig. 8(b)), then (1) $x(p) = x(d)$ and $|y(p) - y(d)| =$ the horizontal distance of p from ℓ_a, (2) the points q and r lie on $\mathbb{T}_1$ satisfying $y(q) = y(a)$ and $x(r) = x(f)$.

(B) If extended ℓ_a and ℓ_d intersect at a point to the left of $\overline{he}$ (Fig. 8(c, d)), then $\mathbb{D}_1$ is a polyline $\overline{pqrs}$, where
(i) the line segment $\overline{pq}$ is such that for every point $\theta \in \overline{pq}$, the horizontal distance of θ from ℓ_a and the vertical distance of θ from ℓ_d are same.
(ii) the line segment $\overline{qr}$ is such that for every point $\theta \in \overline{qr}$, we have
if $|\mathbf{ha}| < |\mathbf{hd}|$ then $|x(\theta) - x(a)| =$ vertical distance of θ from ℓ_d (Fig. 8(c)), **else** $|y(\theta) - x(d)| =$ horizontal distance of θ from ℓ_a, (Fig. 8(d))
(iii) the point s lies on $\mathbb{T}_1$ satisfying $x(s) = x(f)$.

In the same way, we can compute the reference line $\mathbb{D}_2$ based on the four possible orientations of ℓ_b and ℓ_c. The break points/end points of $\mathbb{D}_2$ will be referred to as p', q', r', s' depending on the appropriate cases. From now onwards, we refer the position of the square $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) by mentioning the position of its top-right corner (resp. bottom-left corner).

Definition 4. The distance $d_\infty(p, \ell)$ (in L_∞ norm) of a point p from a line segment ℓ is defined by the L_∞ distance of p to its closest point lying on the line ℓ.

Lemma 3. *The point $p \in \mathbb{D}_1$ (resp. $p' \in \mathbb{D}_2$) gives the position of minimum sized axis-parallel square $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) that hit ℓ_a and ℓ_d (resp. ℓ_b and ℓ_c).*

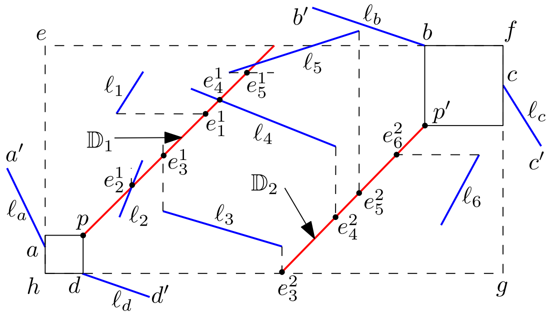

Fig. 9. Event points for **LHIT** problem under **Configuration 1**.

Proof. From the principle of construction of the $\mathbb{D}_1$ (resp. $\mathbb{D}_2$), it follows that among all points z lying on $\mathbb{D}_1$ (resp. $\mathbb{D}_2$), $\max(d_\infty(z, \ell_a), d_\infty(z, \ell_d))$ (resp. $\max(d_\infty(z, \ell_b), d_\infty(z, \ell_c))$) is minimized when $z = p \in \mathbb{D}_1$ (resp. $z = p' \in \mathbb{D}_2$). □

Computation of minimum sized squares $\mathcal{S}_1$ and $\mathcal{S}_2$ to hit a line segment ℓ_i: Let L_V (resp. L_H) denotes the vertical (resp. horizontal) half-line below (resp. to the left of) the point $p \in \mathbb{D}_1$. Similarly, L'_V (resp. L'_H) denotes the vertical (resp. horizontal) half-line above (resp. to the right of) the point $p' \in \mathbb{D}_2$. Observe that, if a line segment $\ell_i \in \mathcal{L}$ intersects with any of L_H or L_V, or if ℓ_i lie completely below L_H and to the left of L_V, then it (ℓ_i) will be hit by any square that hits both ℓ_a and ℓ_d. Similarly, if a line segment ℓ_i intersects with any of L'_H or L'_V; or if ℓ_i lies completely above L'_H and to the right of L'_V, then it (ℓ_i) will be hit by any square that hits both ℓ_b and ℓ_c. Thus, such line segments will not contribute any event point on $\mathbb{D}_1$ (resp. $\mathbb{D}_2$).

Computation of event points on the reference lines $\mathcal{D}_1$ and $\mathcal{D}_2$: For each of the line segments $\ell_i \in \mathcal{L}$, we create two event points $e_i^1 \in \mathbb{D}_1$ and $e_i^2 \in \mathbb{D}_2$, as follows:

(i) If ℓ_i lies completely above $\mathbb{D}_1$ (resp. $\mathbb{D}_2$), then we compute the event point $e_i^1 = (x_{i_1}, y_{i_1})$ on $\mathbb{D}_1$ (resp. $e_i^2 = (x_{i_2}, y_{i_2})$ on $\mathbb{D}_2$) satisfying $y_{i_1} = y(BP(\ell_i))$ (resp. $x_{i_2} = x(RP(\ell_i))$). (see the points e_1^1 for ℓ_1 and e_4^2 for ℓ_4 in Fig. 9).

(ii) If ℓ_i lies completely below $\mathbb{D}_1$ (resp. $\mathbb{D}_2$), we compute the event point $e_i^1 = (x_{i_1}, y_{i_1})$ on $\mathbb{D}_1$ (resp. $e_i^2 = (x_{i_2}, y_{i_2})$ on $\mathbb{D}_2$) satisfying $x_{i_1} = x(LP(\ell_i))$ (resp. $y_{i_2} = y(TP(\ell_i))$). (see e_3^1 for ℓ_3 and e_6^2 for ℓ_6 in Fig. 9).

(iii) If ℓ_i intersects with $\mathbb{D}_1$ (resp. $\mathbb{D}_2$) at point p_1 (resp. q_1), then we create the event point e_i^1 on $\mathbb{D}_1$ (resp. e_i^2 on $\mathbb{D}_2$) according to the following rule:

- **(a)** If the $x(BP(\ell_i)) > x(p_1)$ (resp. $x(TP(\ell_i)) < x(q_1)$), then we take p_1 (resp. q_1) as the event point e_1^i (resp. e_2^i) (see e_4^1 for ℓ_4 in Fig. 9).
- **(b)** If $x(BP(\ell_i)) < x(p_1)$ then if $BP(\ell_i)$ lies below $\mathbb{D}_1$ then we consider the point of intersection by $\mathbb{D}_1$ with the vertical line passing through the $BP(\ell_i)$ as the point e_i^1 (see e_2^1 for ℓ_2 in Fig. 9), and
if $BP(\ell_i)$ lies above $\mathbb{D}_1$ then we consider the point of intersection $\mathbb{D}_1$ with the horizontal line passing through $BP(\ell_i)$ as the event point e_i^1 (see e_5^1 for ℓ_5 in Fig. 9).
- **(c)** If $x(TP(\ell_i)) > x(q_1)$ then if $TP(\ell_i)$ lies above $\mathbb{D}_2$ then we consider the point of intersection by $\mathbb{D}_2$ with the vertical line passing through $TP(\ell_i)$ as the event point e_i^2, and if $TP(\ell_i)$ lies below $\mathbb{D}_2$ then we consider the point of intersection $\mathbb{D}_2$ with the horizontal line passing through $TP(\ell_i)$ as the event point e_i^2.

Lemma 4. *(i) An event e_i^1 on $\mathbb{D}_1$ shows the position of the top-right corner of the minimum sized square $\mathcal{S}_1$ that hits ℓ_a, ℓ_d and ℓ_i, and an event e_i^2 on $\mathbb{D}_2$ shows the position of the bottom-left corner of the minimum sized square $\mathcal{S}_2$ that hits ℓ_b, ℓ_c and ℓ_i.*

(ii) The square $\mathcal{S}_1$ whose top-right corner is at e_i^1 on $\mathbb{D}_1$ hits all those line segments ℓ_j whose corresponding event points e_j^1 on $\mathbb{D}_1$ satisfies $x(h) \le x(e_j^1) \le x(e_i^1)$. Similarly, the square $\mathcal{S}_2$ whose bottom-left corner is at e_2^i on $\mathbb{D}_2$ hits all those line segments ℓ_j whose corresponding event point e_j^2 on $\mathbb{D}_2$ satisfies $x(e_i^1) \le x(e_j^1) \le x(f)$.

Proof. Follows from the method of generating the event points on the "reference line". □

Thus for each line segment $\ell_i \in \mathcal{L}$, we have two parameters σ_{i_1} and σ_{i_2}, where σ_{i_1} (resp. σ_{i_2}) denotes the size of the minimum square required to hit ℓ_i, ℓ_a and ℓ_d (resp. ℓ_i, ℓ_b and ℓ_c). It is to be noted that p and p' are also the event points on $\mathbb{D}_1$ and $\mathbb{D}_2$, respectively (see Lemma 3). We now compute two minimum sized squares $\mathcal{S}_1$ and $\mathcal{S}_2$ to hit all the line segments in $\mathcal{L}$ as follows:

Let $\sigma_{\min_1}$ (resp. $\sigma_{\min_2}$) denote the size of the minimum square $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) required to hit the line segments ℓ_a and ℓ_d (resp. ℓ_b and ℓ_c). Initially we compute these $\sigma_{\min_1}$ and $\sigma_{\min_2}$ which are determined by the position of the point p and p' lying on $\mathbb{D}_1$ and $\mathbb{D}_2$, respectively. Then for each line segment $\ell_i \in \mathcal{L}$, we compute σ_{i_1} and σ_{i_2}, and compare between

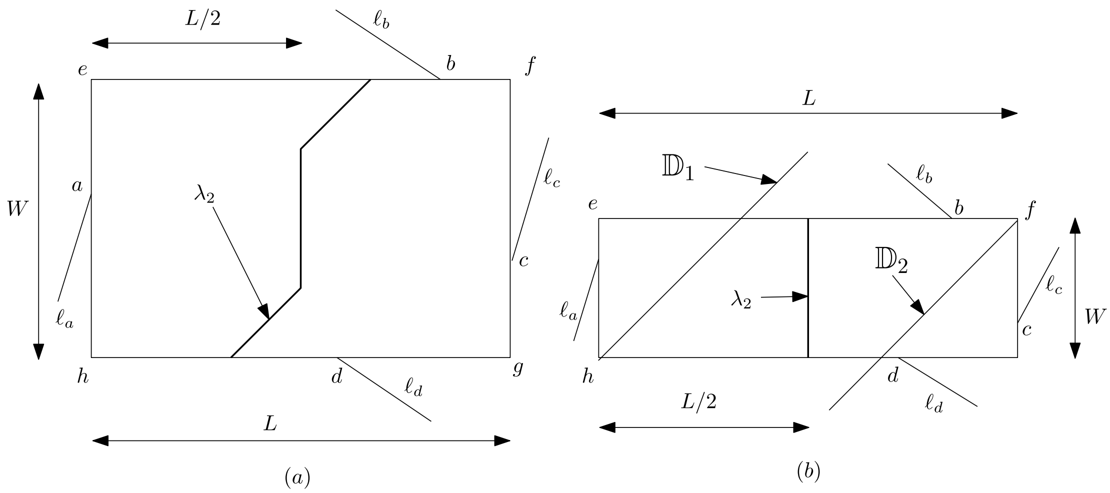

Fig. 10. The end-points "b" and "d" lie on the same side of λ_2 (a) $W > \frac{L}{2}$ (b) $W \leq \frac{L}{2}$.

them. Our objective is to reduce the size of the larger square; hence if $\sigma_{i_1} \leq \sigma_{i_2}$, then we choose the square $\mathcal{S}_1$ to hit ℓ_i, otherwise we choose square $\mathcal{S}_2$. If $\sigma_{i_1} \leq \sigma_{i_2}$, then we compare σ_{i_1} with $\sigma_{\min_1}$. If $\sigma_{i_1} > \sigma_{\min_1}$, then we update $\sigma_{\min_1}$ as σ_{i_1}, otherwise $\sigma_{\min_1}$ remains same. On the other hand, if $\sigma_{i_1} > \sigma_{i_2}$, we compare σ_{i_2} with $\sigma_{\min_2}$, and update $\sigma_{\min_2}$ as σ_{i_2} only if $\sigma_{\min_2}$ is less than σ_{i_2}. After all the line segments have been processed sequentially, the $\max(\sigma_{\min_1}, \sigma_{\min_2})$ will give the minimum size of the congruent squares $\mathcal{S}_1$ and $\mathcal{S}_2$ to hit all the line segments of $\mathcal{L}$ in **Configuration 1**. It is to be noted that while processing the line segments in $\mathcal{L}$ sequentially, for each line segment $\ell_i \in \mathcal{L}$, we need to generate the two event points (e_i^1, e_i^2), compute $(\sigma_{i_1}, \sigma_{i_2})$, and use it to update $(\sigma_{\min_1}, \sigma_{\min_2})$, and use the same locations for processing the next line segment $\ell_j \in \mathcal{L}$. Hence, the aforesaid steps can be executed in linear time using $O(1)$ space.

Similarly, in the same pass, we can determine the optimal size of the congruent squares $\mathcal{S}_1$ and $\mathcal{S}_2$ in **Configuration 2**. For **Configuration 2**, we use separate locations to store the corresponding reference lines, the event points (e_i^1, e_i^2) and the aforesaid variables $(\sigma_{i_1}, \sigma_{i_2})$ which needs $O(1)$ space. Hence, we can compute optimal size of the congruent squares for both the **Configuration 1** and **Configuration 2** during the same pass by processing each line segments in $\mathcal{L}$ sequentially in the aforesaid way.

Finally we consider that configuration for which the size of the congruent squares is minimized. This entire process takes linear amount of time.

Special Case: The end-points "b" and "d" lie on the same side of the Voronoi partitioning line in a configuration.

The two Voronoi partitioning lines for the pair of vertices (e, g) and (f, h) of the rectangle $\mathcal{R} = \square efgh$ are λ_1 and λ_2, respectively (see Fig. 3).

Without loss of generality, we assume that end-points "b" and "d" of the line segments ℓ_b and ℓ_d, respectively lie on the right side of λ_2 (see Fig. 10), i.e. $\mathcal{S}_1$ and $\mathcal{S}_2$ satisfy **Configuration 2**. The square $\mathcal{S}_2$ must cover both these end-points "b" and "d", and hence the size of the $\mathcal{S}_2$ must be at least W. Now there are two possibilities:

(i) $\boldsymbol{W > \frac{L}{2}}$: Refer to Fig. 10(a). In this case, the size of $\mathcal{S}_2$ will be exactly W and this W will be the optimal size of both the squares $\mathcal{S}_1$ and $\mathcal{S}_2$ under this **Configuration 2** since $\max(W, \frac{L}{2}) = W$ (see Fact 1 in Section 2). It needs to be noted that "b" and "d" may lie at the different side with respect to the other Voronoi partitioning line λ_1, and if such is the case, then the size of $\mathcal{S}_1$ and $\mathcal{S}_2$ (that satisfy **Configuration 1**) are to be computed following the aforesaid algorithm for **LHIT problem** and report $\max(size(\mathcal{S}_1), size(\mathcal{S}_2))$ (of **Configuration 1**) as the size of the congruent squares if this size is less than W.

(ii) $\boldsymbol{W \leq \frac{L}{2}}$: Refer to Fig. 10(b). In this case, $\lambda_1 = \lambda_2$ and the size of $\mathcal{S}_2$ will be at least W. So we anchor the squares $\mathcal{S}_1$ and $\mathcal{S}_2$ either at (e, g) or at (h, f) of the rectangle $\mathcal{R}$. In this case, the reference lines $\mathbb{D}_1$ and $\mathbb{D}_2$ will be two parallel rays of slope 1 originating from the corresponding two anchored vertices and we follow the algorithm for **LHIT problem** and determine the optimal size of the congruent squares in linear time.

Theorem 2. *The* **LHIT** *problem can be solved optimally by reading the input array only two times in sequential manner using $O(1)$ extra work-space.*

Proof. Our algorithm for **LHIT** problem needs two sequential passes over the input line segments. In the first pass, we compute the four line segments ℓ_a, ℓ_b, ℓ_c and ℓ_d which are required to determine the rectangle $\mathcal{R} = \square efgh$ and the two reference lines $\mathbb{D}_1$, $\mathbb{D}_2$. In the second pass, we compute the optimal size of the congruent squares by reading the input

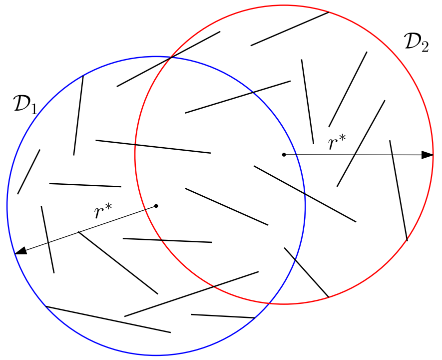

Fig. 11. Covering $\mathcal{L}$ by two disks $\mathcal{D}_1$ & $\mathcal{D}_2$.

array for both type of configurations. Finally, we report that configuration for which the size of the congruent squares is minimized. □

4. Restricted version of LCOVER problem

In the restricted version of the **LCOVER** problem, each line segment in $\mathcal{L}$ is to be covered completely by at least one of the two congruent axis-parallel squares $\mathcal{S}_1$ and $\mathcal{S}_2$. We compute the axis-parallel rectangle $\mathcal{R} = \square efgh$ passing through the four points "a", "b", "c" and "d" as in our algorithm for **LCOVER** problem. As in the **LCOVER** problem, here also we have two possible configurations for optimal solution. Without loss of generality, we assume that $\mathcal{S}_1$ and $\mathcal{S}_2$ satisfy **Configuration 1**. We consider two reference lines $\mathbb{D}_1$ and $\mathbb{D}_2$, each with unit slope that passes through h and f, respectively. These reference lines $\mathbb{D}_1$ and $\mathbb{D}_2$ are the locus of the top-right corner of $\mathcal{S}_1$ and bottom-left corner of $\mathcal{S}_2$, respectively. For each line segment ℓ_i, we create an event point $e_i^1 = (x_{i_1}, y_{i_1})$ on $\mathbb{D}_1$ (resp. $e_i^2 = (x_{i_2}, y_{i_2})$ on $\mathbb{D}_2$) as follows:

(i) If ℓ_i lies completely above $\mathbb{D}_1$ (resp. $\mathbb{D}_2$), then the event point e_i^1 on $\mathbb{D}_1$ (resp. e_i^2 on $\mathbb{D}_2$) will satisfy $y_{i_1} = y(TP(\ell_i))$ (resp. $x_{i_2} = x(LP(\ell_i))$).
(ii) If ℓ_i lies completely below $\mathbb{D}_1$ (resp. $\mathbb{D}_2$) then the event point e_i^1 on $\mathbb{D}_1$ (resp. e_i^2 on $\mathbb{D}_2$) will satisfy $x_{i_1} = x(RP(\ell_i))$ (resp. $y_{i_2} = y(BP(\ell_i))$).
(iii) If ℓ_i intersects with $\mathbb{D}_1$ then we create the event point e_i^1 on $\mathbb{D}_1$ as follows:
Let the horizontal line through $TP(\ell_i)$ intersect with $\mathbb{D}_1$ at point p, and the vertical line through $BP(\ell_i)$ intersect with $\mathbb{D}_1$ at point q. If $x(p) > x(q)$, then we take p (else q) as the event point on $\mathbb{D}_1$.
(iv) If ℓ_i intersects with $\mathbb{D}_2$, then we create the event point e_i^2 on $\mathbb{D}_2$ as follows:
Let the vertical line through $BP(\ell_i)$ intersect with $\mathbb{D}_2$ at point p, and the horizontal line through $TP(\ell_i)$ intersect with $\mathbb{D}_2$ at point q. If $x(p) > x(q)$, then we take q (else p) as the event point on $\mathbb{D}_2$.

Theorem 3. *The restricted version of* **LCOVER** *problem can be solved optimally by reading the input array only two times in sequential manner using $O(1)$ extra work-space.*

Proof. Observation similar to Lemma 4 for the **LHIT** problem also holds for this problem where $\mathcal{S}_1$ and $\mathcal{S}_2$ cover $\mathcal{L}$ with the restriction mentioned in the problem. Thus, here we can follow the same technique as in **LHIT** problem to prove the said result. □

5. Covering/hitting line segments by two congruent disks

In this section, we consider problems related to **LCOVER**, **LHIT** and **restricted LCOVER** problem, called *two-center problem*, where the objective is to cover, hit or restricted-cover the given line segments in $\mathcal{L}$ by two congruent disks so that their (common) radius is minimized. Fig. 11 demonstrates a covering instance of this *two-center problem*. Here, the algorithm is: first, we compute two axis-parallel squares $\mathcal{S}_1$ and $\mathcal{S}_2$ whose union covers/hits all the members of $\mathcal{L}$ optimally as described in the previous section. Then, we report the circum-circles $\mathcal{D}_1$ and $\mathcal{D}_2$ of $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively as an approximate solution of the *two-center problem*.

Lemma 5. *A lower bound for the optimal radius r^* of two-center problem for $\mathcal{L}$ is the radius r' of in-circle of the two congruent squares $\mathcal{S}_1$ and $\mathcal{S}_2$ of minimum size that cover/hit/restricted-cover $\mathcal{L}$; i.e. $r' \leq r^*$.*

Proof. We suppose that $r^* < r'$, D_1 and D_2 (of radius r') be the in-circles of $\mathcal{S}_1$ and $\mathcal{S}_2$. Let D_3 and D_4 (of radius r^*) be the optimum solution for the two-center problem, and S_3 and S_4 be the minimum size squares covering D_3 and D_4

respectively. Thus, S_3 and S_4 is also a solution for covering $\mathcal{L}$ by two axis-parallel congruent squares. Since $r^* < r'$, the size of S_3 (S_4) is less than the size of S_1 (S_2). This leads to the contradiction that (S_1, S_2) is the optimum solution to cover/hit/restricted-cover $\mathcal{L}$. $\square$

Theorem 4. *Algorithm stated in Section 5, generates a $\sqrt{2}$ approximation result for* LCOVER, LHIT *and* restricted LCOVER *problems for the line segments in $\mathcal{L}$.*

Proof. The radius r of the circum-circle $\mathcal{D}_1$ and $\mathcal{D}_2$ of the squares $\mathcal{S}_1$ and $\mathcal{S}_2$ is $\sqrt{2}$ times of the radius r' of their in-circles. Lemma 5 says that $r' \leq r^*$. Thus, we have $r = \sqrt{2}r' \leq \sqrt{2}r^*$. $\square$

6. Conclusion

In this paper, we have computed two axis-parallel congruent squares of minimum size that cover or hit a set of line segments $\mathcal{L}$ in $\mathbb{R}^2$, by reading the input array only two times in a sequential manner using $O(1)$ extra work-space. We have also considered a restricted version of the covering problem, where each line segment in $\mathcal{L}$ is to be covered completely by at least one of the two congruent axis-parallel squares, and to solve it optimally, we have shown a two-pass algorithm for it. Our algorithm gives a $\sqrt{2}$ approximation result to cover or hit the given line segments by two congruent disks of minimum size. It remains a challenge to design an exact algorithm to cover (resp. hit) a given set of line segments $\mathcal{L}$ by two congruent disks of minimum size.

References

[1] S. Bereg, B.K. Bhattacharya, S. Das, T. Kameda, P.R.S. Mahapatra, Z. Song, Optimizing squares covering a set of points, Theoret. Comput. Sci. 729 (2018) 68–83.
[2] A. Aggarwal, J.S. Chang, C.K. Yap, Minimum area circumscribing polygons, Vis. Comput. 1 (2) (1985) 112–117.
[3] H. Alt, E.M. Arkin, A. Efrat, G. Hart, F. Hurtado, I. Kostitsyna, A. Kroller, J.S.B. Mitchell, V. Polishchuk, Scandinavian thins on top of cake: new and improved algorithms for stacking and packing, Theory Comput. Syst. 54 (4) (2014) 689–714.
[4] B.K. Bhattacharya, A. Mukhopadhyay, On the minimum perimeter triangle enclosing a convex polygon, in: Proceedings JCDCG, 2002, pp. 84–96.
[5] B. Becker, P.G. Franciosa, S. Gschwind, T. Ohler, G. Thiemt, P. Widemayer, Enclosing many boxes by an optimal pair of boxes, in: Proceedings STACS, 1992, pp. 475–486.
[6] M. Basappa, R.K. Jallu, G.K. Das, Constrained k-center problem on a convex polygon, in: Proceedings ICCSA, in: LNCS, vol. 9155, 2015, pp. 209–222.
[7] T.M. Chan, More planar two-center algorithms, Comput. Geom. 13 (3) (1999) 189–198.
[8] Z. Drezner, On the rectangular p-center problem, Naval Res. Logist. 34 (2) (1987) 229–234.
[9] G.K. Das, S. Roy, S. Das, S.C. Nandy, Variations of base-station placement problem on the boundary of a convex region, Internat. J. Found. Comput. Sci. 19 (2) (2008) 405–427.
[10] H. Du, Y. Xu, An approximation algorithm for k-center problem on a convex polygon, J. Comb. Optim. 27 (3) (2014) 504–518.
[11] D. Eppstein, Faster construction of planar two-centers, in: Proceedings SODA, 1997, pp. 131–138.
[12] J. Hershberger, A fast algorithm for the two-center decision problem, Inform. Process. Lett. 47 (1) (1993) 23–29.
[13] M. Hoffmann, A simple linear algorithm for computing rectilinear 3-centers, Comput. Geom. 31 (3) (2005) 150–165.
[14] J.W. Jaromczyk, M. Kowaluk, An efficient algorithm for the Euclidean two-center problem, in: Proceedings SoCG, 1994, pp. 303–311.
[15] M.J. Katz, K. Kedem, M. Segal, Discrete rectilinear 2-center problems, Comput. Geom. 15 (4) (2000) 203–214.
[16] S.S. Kim, S.W. Bae, H.K. Ahn, Covering a point set by two disjoint rectangles, Internat. J. Comput. Geom. Appl. 21 (3) (2011) 313–330.
[17] S.K. Kim, C.S. Shin, Efficient algorithms for two-center problems for a convex polygon, in: Proceedings COCOON, 2000, pp. 299–309.
[18] A. Marchetti-Spaccamela, The p center problem in the plane is NP complete, in: Proceedings Allerton Conf. Commun. Control Comput., 1981, pp. 31–40.
[19] N. Megiddo, Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems, SIAM J. Comput. 12 (4) (1983) 759–776.
[20] J.S.B. Mitchell, V. Polishchuk, Minimum-perimeter enclosures, Inform. Process. Lett. 107 (2008) 120–124.
[21] J. O'Rourke, A. Aggarwal, S.R. Maddila, M. Baldwin, An optimal algorithm for finding minimal enclosing triangles, J. Algorithms 7 (2) (1986) 258–269.
[22] S. Roy, D. Bardhan, S. Das, Base station placement on boundary of a convex polygon, J. Parallel Distrib. Comput. 68 (2) (2008) 265–273.
[23] C. Schwarz, J. Teich, E. Welzl, B. Evans, On Finding a Minimal Enclosing Parallelogram, Tech. report: TR-94-036, International Computer Science Institute, Berkeley, CA, 1994.
[24] M. Sharir, A near-linear algorithm for the planar 2-center problem, Discrete Comput. Geom. 18 (2) (1997) 125–134.
[25] O. Schwarzkopf, U. Fuchs, G. Rote, E. Welzl, Approximation of convex figures by pairs of rectangles, Comput. Geom. 10 (1998) 77–87.
[26] X. Tan, B. Jiang, Simple $O(n \log^2 n)$ algorithms for the planar 2-center problem, in: Proceedings COCOON, 2017, pp. 481–491.
[27] X. He, Z. Liu, B. Su, Y. Xu, F. Zheng, B. Zhu, Hitting a set of line segments with one or two Discrete Centers, in: Proceedings CCCG, 2018, pp. 85–90 .