2-2-2020

# Corrigendum to: “Linear time algorithm to cover and hit a set of line segments optimally by two axis-parallel squares” (Theoretical Computer Science (2019) 769 (63–74), (S0304397518306303), (10.1016/j.tcs.2018.10.013))

Sanjib Sadhu
*National Institute of Technology, Durgapur*

Xiaozhou He
*Sichuan University*

Sasanka Roy
*Indian Statistical Institute, Kolkata*

Subhas C. Nandy
*Indian Statistical Institute, Kolkata*

Suchismita Roy
*National Institute of Technology, Durgapur*

Theoretical Computer Science 806 (2020) 632–640

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Corrigendum

# Corrigendum to: "Linear time algorithm to cover and hit a set of line segments optimally by two axis-parallel squares" [Theor. Comput. Sci. 769 (2019) 63–74]

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

[a] Dept. of CSE, National Institute of Technology Durgapur, India
[b] Business School, Sichuan University, Chengdu, China
[c] Indian Statistical Institute, Kolkata, India

ARTICLE INFO

*Article history:*
Received 24 September 2019
Accepted 27 September 2019
Available online 17 October 2019
Communicated by P.G. Spirakis

*Keywords:*
Two-center problem
Hitting line segments by two axis-parallel squares

ABSTRACT

In the paper "Linear time algorithm to cover and hit a set of line segments optimally by two axis-parallel squares", Theor. Comput. Sci. 769 (2019) 63–74, the LHIT problem is proposed as follows:

> For a given set of non-intersecting line segments $\mathcal{L} = \{\ell_1, \ell_2, \dots, \ell_n\}$ 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 $\mathcal{L}$,

and a linear time algorithm was proposed. Later it was observed that the algorithm has a bug. In this corrigendum, we corrected the algorithm. The time complexity of the corrected algorithm is $O(n^2)$.



## 1. Introduction

For a given set of line segments $\mathcal{L} = \{\ell_1, \ell_2, \dots, \ell_n\}$ in $\mathbb{R}^2$, the following two problems were proposed in [1]:

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

**Line segment hitting (LHIT) problem:** Given a set $\mathcal{L} = \{\ell_1, \ell_2, \dots, \ell_n\}$ of $n$ non-intersecting 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 $\mathcal{L}$.

For both the problems, linear time algorithms were proposed. Later, we identified that there is a bug in the proposed algorithm for the LHIT problem. In this corrigendum, we present a revised algorithm for the LHIT problem. The time complexity of this algorithm is $O(n^2)$ in the worst case.

DOI of original article: https://doi.org/10.1016/j.tcs.2018.10.013.
* Corresponding author.
*E-mail address:* sanjibsadhu411@gmail.com (S. Sadhu).

https://doi.org/10.1016/j.tcs.2019.09.044
0304-3975/

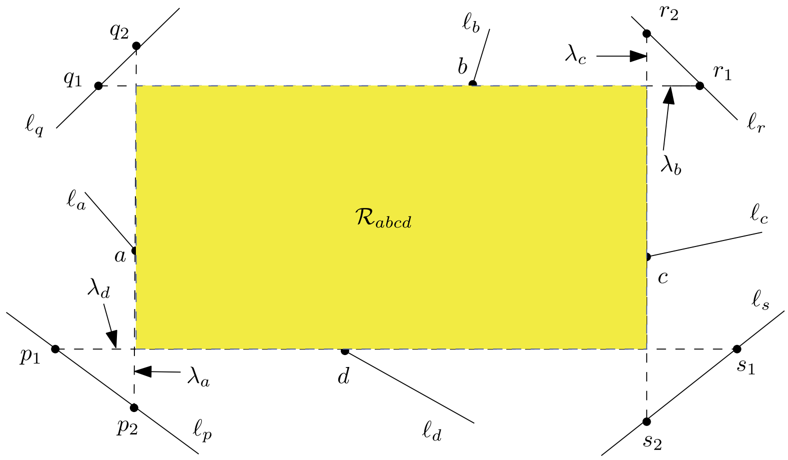


**Fig. 1.** The axis parallel rectangle $\mathcal{R}_{abcd}$ defined by the points $a$, $b$, $c$ and $d$ that does not hit all the members in $\mathcal{L}$.

*An axis parallel rectangle $\mathcal{R}$ is called a hitting rectangle if every member in $\mathcal{L}$ is either intersected by $\mathcal{R}$ or is completely contained in $\mathcal{R}$.* In [1], we performed a linear scan among the objects in $\mathcal{L}$ to identify four points $a$, $b$, $c$ and $d$, where $a$ is the right end-point of a segment $\ell_a \in \mathcal{L}$ having minimum $x$-coordinate, $b$ is the bottom end-point of a segment $\ell_b \in \mathcal{L}$ having maximum $y$-coordinate, $c$ is the left end-point of a segment $\ell_c \in \mathcal{L}$ having maximum $x$-coordinate, and $d$ is the top end-point of a segment $\ell_d \in \mathcal{L}$ having minimum $y$-coordinate (see Fig. 1). The axis-parallel rectangle whose "left", "top", "right" and "bottom" sides contain the points $a$, $b$, $c$ and $d$ respectively, is denoted by $\mathcal{R}_{abcd}$. In [1], we claimed that this axis-parallel rectangle $\mathcal{R}_{abcd}$ is a *hitting rectangle*. Using this rectangle, we computed two congruent squares of minimum size that hits all the line segments in $\mathcal{L}$. Later, we observed that $\mathcal{R}_{abcd}$ is not always a hitting rectangle (see Fig. 1). Thus, the proposed algorithm for the LHIT problem may fail in some pathological cases. In this corrigendum, we correct our mistake. As in [1], we first compute $\mathcal{R}_{abcd}$. If it hits all the segments in $\mathcal{L}$, our proposed linear time algorithm in [1] will work for the LHIT problem. However, if $\mathcal{R}_{abcd}$ does not hit all the segments in $\mathcal{L}$, we propose an $O(n^2)$ time algorithm for the LHIT problem.

As mentioned earlier, the members in $\mathcal{L}$ are non-intersecting. We use the following notations to describe our revised algorithm. Here, $\lambda_a$, $\lambda_b$, $\lambda_c$ and $\lambda_d$ denote the lines containing the left, top, right and bottom boundaries of $\mathcal{R}_{abcd}$ respectively. Let $\ell_p$ be the segment which is not hit by $\mathcal{R}_{abcd}$ and lies farthest from both "$a$" and "$d$" along vertically downward and horizontally leftward directions respectively. Similarly the other segments $\ell_q$, $\ell_r$ and $\ell_s$ are defined (see Fig. 1). Let $(p_1, p_2)$ be the two points of intersection of $\ell_p$ with $\lambda_a$ and $\lambda_d$ respectively. Similarly the point-pairs $(q_1, q_2)$, $(r_1, r_2)$ and $(s_1, s_2)$ are defined (see Fig. 1). Note that, all the segments $\ell_p$, $\ell_q$, $\ell_r$, $\ell_s$ may not exist. However, if at least one of these four segments exists, then our proposed algorithm in [1] will fail.

We first propose an algorithm for computing a minimum sized axis parallel square $\mathcal{S}$ that hits a given set of line segments $\mathcal{L}$. We use this result to compute the two axis parallel congruent squares $\mathcal{S}_1$ and $\mathcal{S}_2$ of minimum size for hitting all the segments in $\mathcal{L}$.

## 2. One hitting square

**Fact 1.** *A square, that hits $\ell_a$, $\ell_b$, $\ell_c$, $\ell_d$, $\ell_p$, $\ell_q$, $\ell_r$ and $\ell_s$ (those which exists), will hit all the segments in $\mathcal{L}$.*

**Proof.** Let $\mathcal{R}$ be a square that hit all the segments in $\{\ell_a, \ell_b, \ell_c, \ell_d, \ell_p, \ell_q, \ell_r, \ell_s\}$, and $\ell \in \mathcal{L} \setminus \{\ell_a, \ell_b, \ell_c, \ell_d, \ell_p, \ell_q, \ell_r, \ell_s\}$ be a segment that is not hit by $\mathcal{R}$. The square $\mathcal{R}$ must cover $\mathcal{R}_{abcd}$ (Fig. 1). So by our assumption, $\ell$ must not intersect $\mathcal{R}_{abcd}$. From the definition of the distinguished points "$a$", "$b$", "$c$" and "$d$", the segment $\ell$ must intersect both the members of at least one of the tuples $(\lambda_a, \lambda_b)$, $(\lambda_b, \lambda_c)$ and $(\lambda_c, \lambda_d)$, and $(\lambda_a, \lambda_d)$ outside $\mathcal{R}_{abcd}$. Without loss of generality, assume that $\ell$ hits $(\lambda_a, \lambda_d)$. In order to hit $\ell_p$ by $\mathcal{R}$, it must hit $\ell$. Thus, we have the contradiction. □

**Implication of Fact 1:** The minimum size square hitting all the segments in a given set $\mathcal{L}$ is defined by at most eight segments $\{\ell_a, \ell_b, \ell_c, \ell_d, \ell_p, \ell_q, \ell_r, \ell_s\}$ of $\mathcal{L}$.

**Observation 1.** *(i) The subset of $\mathcal{L}$ defining the possible minimum size squares hitting all the segments in $\mathcal{L}$ (if more than one such squares exist) is unique.*

*(ii) If $\mathcal{S}$ is the* minimum sized axis parallel square *that hits all the line segments in $\mathcal{L}$, then at least one of the vertices of $\mathcal{S}$ will lie on one of the four segments $\overline{p_1p_2}$, $\overline{q_1q_2}$, $\overline{r_1r_2}$ and $\overline{s_1s_2}$.*

**Proof. Part (i)**: A minimum sized square $\mathcal{S}$ hitting all the segments is defined by either two or three segments which are termed as the defining segments for $\mathcal{S}$.

(a) If the number of defining segments of $\mathcal{S}$ is two, then those two segments must touch the two opposite boundaries (left, right) or (top, bottom) of $\mathcal{S}$, or two diagonal vertices of $\mathcal{S}$. The defining segments must touch the boundary of square $\mathcal{S}$ externally i.e. from outside, otherwise $\mathcal{S}$ can be further reduced.

- **Two defining segments touch the two opposite sides of the square $\mathcal{S}$**: Here, the maximum of "minimum horizontal distance" and "minimum vertical distance" between "two defining segments" (say $\ell_1$ and $\ell_2$) will be the length of the side of $\mathcal{S}$. See Fig. 3(a), (b). If there exists another square $\mathcal{S}'$ that hits all the segment, then $\mathcal{S}'$ will also hit $\ell_1$ and $\ell_2$ indicating that the horizontal/vertical span will increase or remain at least same as that of $\mathcal{S}$. If $\mathcal{S}$ and $\mathcal{S}'$ are of same size (see Fig. 3(a), (b)), then the defining segments of $\mathcal{S}$ and $\mathcal{S}'$ are same.
- **Two defining segments touch the two diagonal vertices of the square $\mathcal{S}$**: If $\mathcal{S}$ is defined by two segments $\ell_1$ and $\ell_2$ touching its two diagonal vertices, then the segments are either parallel to each other (see Fig. 3(c)) or the minimum distance between two defining segments $\ell_1$ and $\ell_2$ is the length of diagonal of $\mathcal{S}$ (see Fig. 3(d)). Here also if there exists another square $\mathcal{S}'$ defined by other two segments $(\ell_1', \ell_2') \neq (\ell_1, \ell_2)$ then the horizontal/vertical span will increase or remain at least same as that of $\mathcal{S}$. If $\mathcal{S}$ and $\mathcal{S}'$ are of same size (in case $\ell_1$ and $\ell_2$ are parallel as shown in Fig. 3(c)), then the defining segments of $\mathcal{S}$ and $\mathcal{S}'$ are same.

(b) If the number of defining segments of $\mathcal{S}$ are three, say $\ell_1$, $\ell_2$ and $\ell_3$, then two of them must touch the two opposite boundaries (left, right) or (top, bottom) of the square $\mathcal{S}$. If there exists any square $\mathcal{S}'$ that hits all the segments in $\mathcal{L}$, then arguing as in the earlier case, it can be shown that the size of $\mathcal{S}'$ is at least as large as $\mathcal{S}$, and the defining segments will remain same.

**Part (ii)**: Assume that none of the vertices of the minimum sized axis parallel square $\mathcal{S}$ lies on $\overline{p_1p_2}$, $\overline{q_1q_2}$, $\overline{r_1r_2}$ and $\overline{s_1s_2}$. It can be shown that, one can translate $\mathcal{S}$ "horizontally towards left or right", and/or "vertically upward or downward" keeping its size unchanged, without missing any segment (i.e. each segment remains hit by $\mathcal{S}$ always) to move one of the vertices of $\mathcal{S}$ touching the respective segment. □

If there are multiple minimum sized congruent squares for hitting the segments (see Fig. 3(a), (b), (c)), then our proposed algorithm for the **LHIT problem** will also work. The reason is that after choosing an $\mathcal{S}_1$, our algorithm for computing $\mathcal{S}_2$ needs only the segments that are not hit by $\mathcal{S}_1$. We increase the size of $\mathcal{S}_1$ monotonically according to the event points corresponding to the top-right corner of $\mathcal{S}_1$. Now in each step, if $\mathcal{S}_1$ hits a defining segment of $\mathcal{S}_2$, then the size of $\mathcal{S}_2$ is reduced by eliminating that segment from it. If there exists multiple congruent $\mathcal{S}_2$ of minimum size that hit all the segments which are not hit by $\mathcal{S}_1$, we can choose any one of them as square $\mathcal{S}_2$, since all such $\mathcal{S}_2$'s are defined by the same subset segments (Observation 1(i)).

**Lemma 1.** *An axis parallel square of minimum size hitting all the members of a given set $\mathcal{L}$ of n line segments can be obtained in $O(n)$ time.*

**Proof.** Among the given set $\mathcal{L}$ of $n$ line segments, we can identify the special line segments $\ell_i$, $i \in \{a, b, c, d, p, q, r, s\}$ (see Fig. 1) in $O(n)$ time.

We now show that a minimum sized axis parallel square $\mathcal{S}^r$ whose "top-right" corner lies on $\overline{r_1r_2} \in \ell_r$ and hits all the segments, can be computed in $O(1)$ time. The same method works for computing the minimum sized squares $\mathcal{S}^p$, $\mathcal{S}^q$ and $\mathcal{S}^s$ whose one corner lies on $\overline{p_1p_2}$, $\overline{q_1q_2}$ and $\overline{s_1s_2}$ respectively and hits all the line segments. Finally we will choose minimum sized square among $\mathcal{S}^p$, $\mathcal{S}^q$, $\mathcal{S}^r$ and $\mathcal{S}^s$.

**Computation of $\mathcal{S}^r$**: For each $i \in \{a,\ p,\ q,\ d,\ s\}$, we compute the locus $loc(i)$ of the "bottom-left" corner of a minimum sized square $\mathcal{S}$ which hits the line segment $\ell_i$, while its "top-right" corner moving along the segment $\overline{r_2r_1}$. In Fig. 2(a), $loc(s)$ is demonstrated, while in Fig. 2(b) all the $loc(i)$, $i \in \{a,\ p,\ q,\ d,\ s\}$ are shown. We also compute the locus of the "bottom-left" corner of $\mathcal{S}$ (denoted by $loc(b, c)$ in Fig. 2(b)) that hits both $\ell_b$ and $\ell_c$ while the top-right corner of $S$ moves along the segment $\overline{r_2r_1}$. Each of the loci in $\{loc(i), i = a, p, q, d, s, (b, c)\}$ consists of at most three line segments (see Appendix A for details). We consider two lines $DL_1$ and $DL_2$ of unit slope passing through $r_1$ and $r_2$ respectively (see Fig. 2(b)). We can compute the upper envelope $U$ (as the distance is measured from $\overline{r_2r_1}$) of the loci $\{loc(i),\ i \in \{a, p, q, d, s, (b, c)\}\}$ within the strip bounded by $DL_1$ and $DL_2$ (colored red in Fig. 2(b)) in $O(1)$ time. The square whose "bottom-left" corner lies on the upper envelope $U$ while its "top-right" corner lies on $\overline{r_2r_1}$, hits all the segments $\ell_i$, $i \in \{a, b, c, d, p, q, r, s\}$. Thus, the upper envelope $U$ corresponds to the locus of the bottom-left corner of $\mathcal{S}^r$ that hits all the segment in $\mathcal{L}$ (see Fact 1) while its top-right corner moves along $\overline{r_2r_1}$. Note that $U$ consists of a constant number of segments and it can be computed in $O(1)$ time. As one moves along an edge of $U$, the size of the square $\mathcal{S}^r$ either monotonically increases or decreases or remains same. So, the minimum size of the square $\mathcal{S}^r$ occurs at some vertex of $U$, and it can be determined by inspecting all the vertices of $U$.

If any one of $\ell_p$, $\ell_q$, $\ell_r$ and $\ell_s$ does not exist in the given instance with the segments $\mathcal{L}$, then the corresponding locus is not present, and the same method works in such a situation with the available set of loci. □

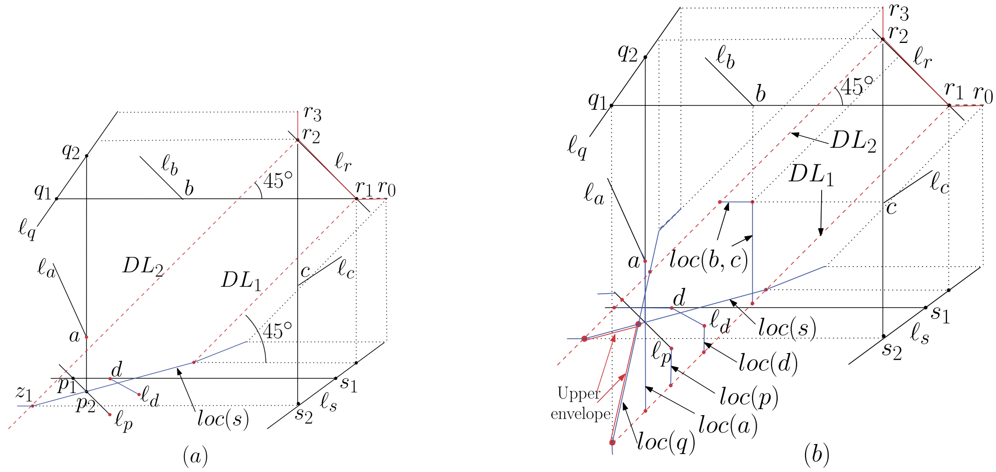


**Fig. 2.** (a) Computation of *loc*(*s*), (b) Computation of a minimum sized axis parallel square that hits all the segments. (For interpretation of the colors in the figures, the reader is referred to the web version of this article.)

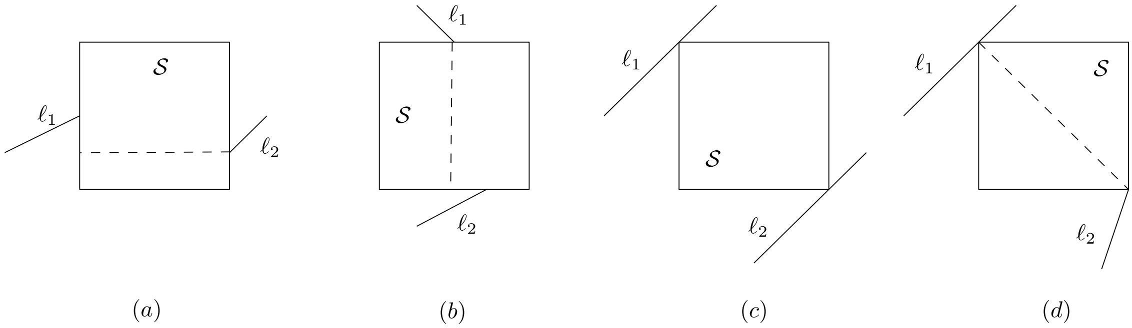


**Fig. 3.** Demonstration of multiple copies minimum sized square $\mathcal{S}$ defined by two segments $\ell_1$ and $\ell_2$: (a) at the left and right boundary of $\mathcal{S}$, (b) at the top and bottom boundary of $\mathcal{S}$, (c) at two diagonal vertices of $\mathcal{S}$ where the segments are parallel, (d) at two diagonal vertices of $\mathcal{S}$ where the segments are non-parallel.

## 3. Two hitting squares

We now discuss the hitting problem by two axis parallel squares ($\mathcal{S}_1$, $\mathcal{S}_2$) using the method described in Section 2 as a subroutine. We assume that $\mathcal{S}_1$ hits $\ell_p$ along with some other members in $\mathcal{L}$. $\mathcal{S}_2$ must hit the members that are not hit by $\mathcal{S}_1$. Our objective is to compute the pair ($\mathcal{S}_1$, $\mathcal{S}_2$) that minimizes $\max(size(\mathcal{S}_1), size(\mathcal{S}_2))$.

**Lemma 2.** *To minimize the* $\max(size(\mathcal{S}_1), size(\mathcal{S}_2))$*, the "bottom-left" corner of* $\mathcal{S}_1$ *will lie on* $\ell_p$*.*

**Proof.** Let $\mathcal{L}_1 \subset \mathcal{L}$ be the set of segments hit by $\mathcal{S}_1$ when $\max(size(\mathcal{S}_1), size(\mathcal{S}_2))$ is minimized. Let the "bottom-left" corner of $\mathcal{S}_1$ lie below $\ell_p$ i.e. both bottom boundary and left boundary of $\mathcal{S}_1$ properly intersect $\ell_p$ (see Fig. 4). Let $\ell_1, \ell_2 \in \mathcal{L}_1$ be two segments so that the $y$-coordinate (resp. $x$-coordinate) of top end-point (resp. right end-point) of $\ell_1$ (resp. $\ell_2$) is minimum among that of all the segment $\ell_k \in \mathcal{L}_1$. If the bottom (resp. left) boundary of $\mathcal{S}_1$ properly intersect $\ell_1$ (resp. $\ell_2$), we can translate $\mathcal{S}_1$ vertically upwards (resp. horizontally rightwards) keeping its size same, so that the bottom boundary (resp. left boundary) of $\mathcal{S}_1$ touches $\ell_1$ (resp. $\ell_2$) or the bottom-left corner of $\mathcal{S}_1$ touches $\ell_p$. If $\ell_p$ is touched, the result is justified. If $\ell_1$ (resp. $\ell_2$) is touched, we can translate $\mathcal{S}_1$ towards right (resp. above) to make the bottom-left corner of $\mathcal{S}_1$ touching $\ell_p$. The revised $\mathcal{S}_1$ also hits all the segments in $\mathcal{L}_1$. □

Lemma 2 says that a square $\mathcal{S}$ serves as $\mathcal{S}_1$ if the boundary of $\mathcal{S}$ touches $\ell_p$ and also hits a subset $\mathcal{L}' \subset \mathcal{L} \setminus \{\ell_p\}$ with at least one segment of $\mathcal{L}'$ touching the boundary of $\mathcal{S}$ from outside. The reason of defining $\mathcal{S}_1$ in such a manner is that if all the segments $\mathcal{L}'$ hit by $\mathcal{S}_1$ lie either inside $\mathcal{S}_1$ or properly intersect the boundary of $\mathcal{S}_1$, then we can reduce the size of $\mathcal{S}_1$ hitting the same set of segments. Now, we will introduce the concept of defining $\mathcal{S}_1$ using a subset of $\mathcal{L}$ as follows:

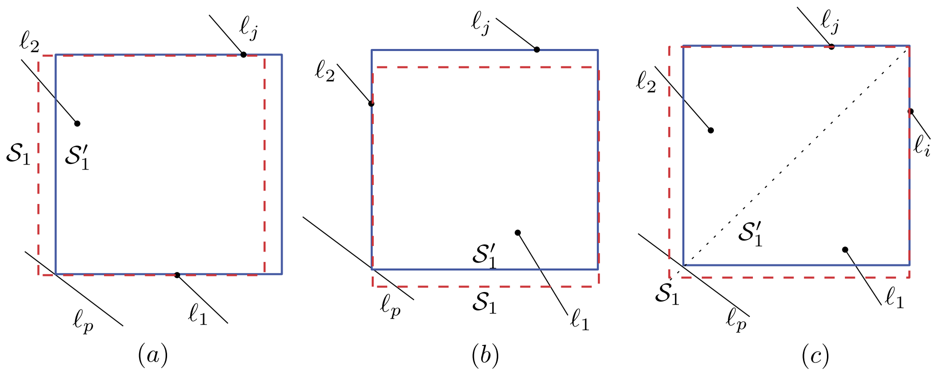


**Fig. 4.** Proof of Lemma 2.

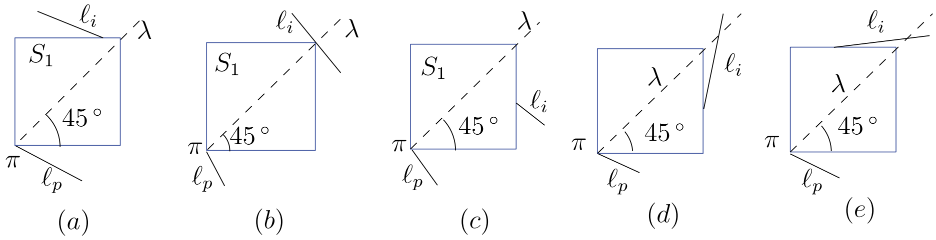


**Fig. 5.** The "bottom-right" corner of square $\mathcal{S}_1$ is at a segment end-point.

**Definition 1.** A subset $\mathcal{L}' \subseteq \mathcal{L} \setminus \{\ell_p\}$ is said to be *minimal* to define a square $\mathcal{S}$ (with bottom-left corner is on $\ell_p$) as $\mathcal{S}_1$ if the members of $\mathcal{L}'$ uniquely determine its top-right corner of $\mathcal{S}$, and no proper subset of $\mathcal{L}'$ can define the top-right corner of $\mathcal{S}$ uniquely.

We will consider possible subsets $\mathcal{L}_1 \subset \mathcal{L}$ that can define $\mathcal{S}_1$, and invoke the procedure described in Section 1 with the subset $\mathcal{L} \setminus (\mathcal{L}_1 \cup \{\ell_p\})$ to compute $\mathcal{S}_2$. The following Lemma 3 and Lemma 4 says that we need to consider the two cases separately depending on whether the bottom-left corner of $\mathcal{S}_1$, denoted by $\pi$, resides at (i) an end-point of $\ell_p$, and (ii) an intermediate point of $\ell_p$.

**Lemma 3.** *If $\pi$ coincides with an end-point of $\ell_p$ (Case (i)), then $\mathcal{S}_1$ is determined by a single segment of $\mathcal{L} \setminus \{\ell_p\}$.*

**Proof.** Here, the top-right corner $\pi'$ of $\mathcal{S}_1$ lies on a line of unit slope passing through $\pi$. We need to investigate the following three exhaustive cases.

- $\pi'$ lies on a segment $\ell_i \in \mathcal{L} \setminus \{\ell_p\}$ (see Fig. 5(b)), or
- $\pi'$ lies on the vertical line passing through the left end-point of a segment $\ell_i \in \mathcal{L} \setminus \{\ell_p\}$ (see Fig. 5(c), (d)), or
- $\pi'$ lies on the horizontal line passing through the bottom end-point of a segment $\ell_i \in \mathcal{L} \setminus \{\ell_p\}$ (see Fig. 5(a), (e)).

This is due to the fact that if none of these cases happen then we can get another square, say $\mathcal{S}'_1$, of reduced size whose bottom-left corner is at $\pi$ and it hits all the segments in $\mathcal{L}$ that are also hit by $\mathcal{S}_1$. Here $\mathcal{S}'_1$ serves the purpose of $\mathcal{S}_1$. Thus, the lemma follows. □

**Lemma 4.** *If $\pi$ coincides with an intermediate point of $\ell_p$ (Case (ii)), then $\mathcal{S}_1$ is determined by two segments of $\mathcal{L} \setminus \{\ell_p\}$.*

**Proof.** In this case, the bottom-left corner of $\mathcal{S}_1$ will be determined as follows:

- a segment $\ell_i \in \mathcal{L} \setminus \{\ell_p\}$ defines the bottom boundary of $\mathcal{S}_1$ whose horizontal projection $\pi$ on $\ell_p$ determines the bottom-left corner of $\mathcal{S}_1$ (see Fig. 6(d), (e)), or
- a segment $\ell_i \in \mathcal{L} \setminus \{\ell_p\}$ defines the left boundary of $\mathcal{S}_1$ whose vertical projection $\pi$ on $\ell_p$ determines the bottom-left corner of $\mathcal{S}_1$ (see Fig. 6(a), (b)), or
- a pair of segments $\ell_i$ and $\ell'_i$ defines the top-right corner $\pi'$ of $\mathcal{S}_1$, and the point of intersection of a line of unit slope passing through $\pi'$ with the line segment $\ell_p$ determines the bottom-left corner of $\mathcal{S}_1$ (see Fig. 6(c)).

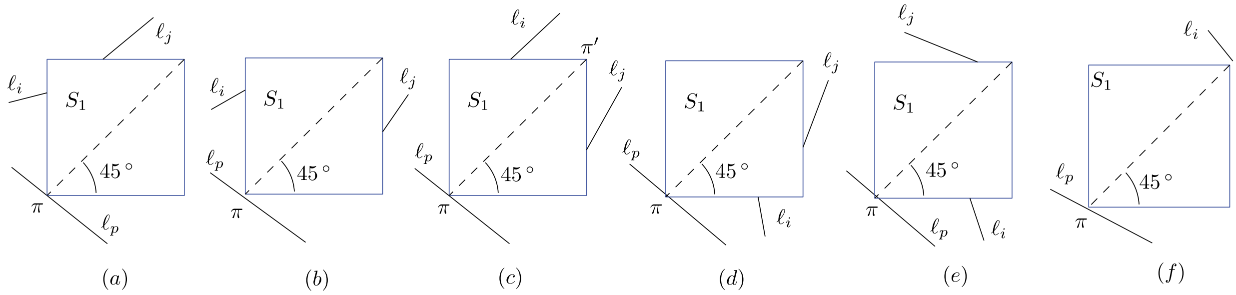


**Fig. 6.** The "top-right" corner of $\mathcal{S}_1$ that hits $\ell_p$ is defined by two segments $\ell_i$ and $\ell_j$.

In the first and second bulleted case, Lemma 3 says that one more segment $\ell_j$ is required to define the top-right corner of $\mathcal{S}_1$. In the third bulleted case, both the bottom-left and the top-right corners of $\mathcal{S}_1$ are already defined. Thus, the lemma follows. □

In the following two subsections we will compute $\mathcal{S}_1$ considering the two cases where (i) $\mathcal{S}_1$ is defined by one segment in $\mathcal{L} \setminus \{\ell_p\}$ and (ii) two segments in $\mathcal{L} \setminus \{\ell_p\}$ respectively. Note that, if a single segment $\ell \in \mathcal{L}$ touches a corner of $\mathcal{S}_1$, then $\ell$ is said to touch both the boundaries of $\mathcal{S}_1$ adjacent to that corner (see Fig. 6(f)).

**(A) $\mathcal{S}_1$ is defined by one line segment:** We draw a straight line $\lambda$ of slope "1" through an end-point $\pi$ of $\ell_p$. Next, we consider each segment $\ell_i \in \mathcal{L} \setminus \{\ell_p\}$, and create an array $Q$ of event points as follows:

- If $\ell_i$ is strictly above $\lambda$ (Fig. 5(a)), store the horizontal projection $q$ of the bottom end-point of $\ell_i$ on the line $\lambda$ in $Q$.
- If $\ell_i$ with negative slope intersects $\lambda$ at a point $q$ (Fig. 5(b)), we store $q$ in $Q$.
- If $\ell_i$ with positive slope ($\leq 1$) intersects $\lambda$ (Fig. 5(e)), store the horizontal projection $q$ of the bottom end-point of $\ell_i$ on the line $\lambda$ in $Q$.
- If $\ell_i$ with positive slope ($> 1$) intersects $\lambda$ (Fig. 5(d)), store the vertical projection $q$ of the left end-point of $\ell_i$ on the line $\lambda$ in $Q$.
- If $\ell_i$ is strictly below $\lambda$ (Fig. 5(c)), then store the vertical projection $q$ of the left end-point of $\ell_i$ on $\lambda$ in $Q$.

We consider each member $q \in Q$. Define $\mathcal{S}_1$ with its (bottom-left, top-right) corner points as $(\pi, q)$. Identify the subset $\mathcal{L}_1$ of segments in $\mathcal{L}$ that are hit by $\mathcal{S}_1$. Call the procedure of Section 1 with the set of segments $\mathcal{L} \setminus \mathcal{L}_1$ to compute $\mathcal{S}_2$. Replace the current optimum square-pair by $\max(size(\mathcal{S}_1), size(\mathcal{S}_2))$ if needed.

**Lemma 5.** *The minimum of the size of the optimum pair of squares where $\mathcal{S}_1$ is defined by one line segment of $\mathcal{L} \setminus \{\ell_p\}$ can be computed in $O(n^2)$ time.*

**Proof.** The array $Q$ can be computed in $O(n)$ time. For each member $q \in Q$, (i) the subset $\mathcal{L}_1$ of $\mathcal{L}$ can be identified in $O(n)$ time, and then (ii) the time required for computing $\mathcal{S}_2$ is also $O(n)$. As $|Q| = O(n)$, the result follows. □

**(B) The top-right corner of $\mathcal{S}_1$ is defined by two line segments:** By Lemma 4, assuming that the bottom-left corner of $\mathcal{S}_1$ lies in the interior of $\ell_p$, we need to consider the following cases to uniquely define the possible bottom-left corner of $\mathcal{S}_1$.

B1: The bottom-left corner of $\mathcal{S}_1$ is defined by the top end-point of a segment $\ell_i$ touching its bottom boundary (see Fig. 6(d), (e)).
B2: The bottom-left corner of $\mathcal{S}_1$ is defined by the right end-point of a segment $\ell_i$ touching its left boundary (see Fig. 6(a), (b)).
B3: The bottom-left corner of $\mathcal{S}_1$ is defined by its top-right corner $\pi'$, defined by a pair of segments $\ell_i$ and $\ell_j$ touching the "top" and "right" boundaries of $\mathcal{S}_1$ (see Fig. 6(c)).

Note that, Fig. 6(f) is basically the case B3, where $\ell_i$ is assumed to touch both the "top" and "right" boundaries of $\mathcal{S}_1$.

We use four arrays $\mathcal{L}_l$, $\mathcal{L}_r$, $\mathcal{L}_t$ and $\mathcal{L}_b$, each with the members in $\mathcal{L}$ sorted with respect to their left, right, top, and bottom end-points respectively. In addition, we keep a sorted array $\mathcal{L}_d$ containing the points of intersection of the line containing $\ell_p$ and the lines of slope 1 (called diagonal lines) at both the end-points of each member in $\mathcal{L} \setminus \{\ell_p\}$. Each element $\ell_i \in \mathcal{L}$ maintains six pointers to the corresponding element in $\mathcal{L}_l$, $\mathcal{L}_r$, $\mathcal{L}_t$, $\mathcal{L}_b$ and to two elements of $\mathcal{L}_d$ corresponding to its two end-points. Also, each element of $\mathcal{L}_i$, $i = l, r, t, b, d$ points to the corresponding segment $\ell \in \mathcal{L}$. In addition, we also maintain four ordered arrays, namely $\mathcal{I}^{v1}(\tau)$, $\mathcal{I}^{v2}(\tau)$ $\mathcal{I}^{h}(\tau)$ and $\mathcal{I}^{d}(\tau)$ for each end-point $\tau$ of the members in $\mathcal{L}$. $\mathcal{I}^{v1}(\tau)$ (resp. $\mathcal{I}^{v2}(\tau)$) is the list of segments hit by an upward (resp. downward) vertical ray from $\tau$, and $\mathcal{I}^{h}(\tau)$ (resp. $\mathcal{I}^{d}(\tau)$) is the

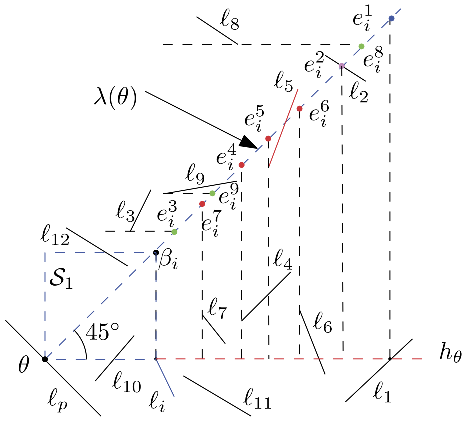


**Fig. 7.** Generation of $\mathcal{D}_\theta$ where $\theta$ is horizontal projection of top end-point of $\ell_i$ on $\ell_p$.

list of segments in $\mathcal{L}$ intersected by the *horizontal line* (resp. *diagonal line*) passing through the point $\tau$ in sorted order. Each segment $\ell_i \in \mathcal{L}$ maintains eight pointers to point the lists $\mathcal{I}^{v1}(\tau)$, $\mathcal{I}^{v2}(\tau)$, $\mathcal{I}^h(\tau)$, $\mathcal{I}^d(\tau)$, $\mathcal{I}^{v1}(\tau')$, $\mathcal{I}^{v2}(\tau')$, $\mathcal{I}^h(\tau')$ and $\mathcal{I}^d(\tau')$ where $\tau$ and $\tau'$ are two end-points of $\ell_i$. The arrays $\mathcal{L}_i$, $i = l, r, t, b, d$ can be created in $O(n \log n)$ time. Also, the arrays $\mathcal{I}^{v1}(\tau)$, $\mathcal{I}^{v2}(\tau)$, $\mathcal{I}^h(\tau)$ and $\mathcal{I}^d(\tau)$ for all the $2n$ end-points ($\tau$) of the segments in $\mathcal{L}$ can be created in $O(n^2)$ time and will be stored using $O(n^2)$ space.

Let us now consider the generation of the instances in B1. Lemma 2 says that if $\ell_p$ exists, then the bottom-left corner of $\mathcal{S}_1$ lies on $\ell_p$. We first generate all possible bottom-left corners $\mathcal{C}$ of $\mathcal{S}_1$ on $\ell_p$ in sorted order whose bottom boundary is supported by the top end-point of a segment $\ell_i$ in $\mathcal{L}$ by traversing the list $\mathcal{L}_t$. For each element $\theta \in \mathcal{C}$ (corresponding to the top-end point of a line segment $\ell_i$), we consider a half-line $\lambda(\theta)$ of slope "1" at the point $\theta$, and generate the array $\mathcal{D}_\theta$ that contains the top-right corner of all possible squares $\mathcal{S}_1$ lying on $\lambda(\theta)$, in order of their distances from the point $\theta$ (see Fig. 7). We denote the horizontal line at $\theta$ by $h_\theta$. The elements (known as event points) of the array $\mathcal{D}_\theta$ are the points of intersection of $\lambda(\theta)$ with

(i) the vertical lines at the *left end-point* of all the segments in $\mathcal{L}$ whose left end-point lies below the line $\lambda(\theta)$ and above the line $h_\theta$ (see *red* points e.g. $e_i^4$, $e_i^5$, $e_i^6$ in Fig. 7),
(ii) the vertical lines at the point of intersection of $h_\theta$ with the segments $\mathcal{L}' \subseteq \mathcal{L}$, provided the slope of the segments in $\mathcal{L}'$ are positive (see *blue* points e.g. $e_i^1$ in Fig. 7),
(iii) the horizontal line at the bottom end-point of all the segments whose bottom end-point lies above $\lambda(\theta)$ (see *green* points e.g. $e_i^3$, $e_i^8$, $e_i^9$ in Fig. 7), and
(iv) the segments in $\mathcal{L}$ with negative slope that intersects $\lambda(\theta)$ (see *pink* points $e_i^2$ in Fig. 7).

Since $\mathcal{S}_1$ hits $\ell_i$, we need to remove all the events generated on $\lambda(\theta)$ whose $x$-coordinates are less than that of the top end-point $\tau$ of $\ell_i$ (e.g. events for $\ell_{10}$, $\ell_{12}$ in Fig. 7).

The Type (i) (resp. Type (iii)) events are generated in increasing order of their $x$-coordinates by scanning the array $\mathcal{L}_l$ (resp. $\mathcal{L}_b$). Type (ii) events are created in increasing order of $x$-coordinates from the list $\mathcal{I}^h(\tau)$, where the horizontal projection of the top end-point $\tau$ of the line segment $\ell_i$ on $\ell_p$ is $\theta$. Type (iv) events are identified from the two ordered arrays $\mathcal{I}^d(p_1)$ and $\mathcal{I}^d(p_2)$ where $p_1$ and $p_2$ are two end-points of (same or different) line segments that generated two consecutive event points $e$ and $e'$ in the array $\mathcal{L}_d$, and $x(e) \leq x(\theta) \leq x(e')$. Note that we need to consider only the segments of negative slope in $\mathcal{I}^d(p_1) \cup \mathcal{I}^d(p_2)$ in ordered manner to compute Type (iv).

Now, we merge the events of Types (i) to (iv) to get the list $\mathcal{D}_\theta$ containing all possible events on $\lambda_\theta$ arranged in increasing order of their $x$-coordinates. We process each event of $\delta \in \mathcal{D}_\theta$ by executing the steps (i) compute an $\mathcal{S}_1$ square with (bottom-left, top-right) corners at $(\theta, \delta)$, (ii) identify the segments in $\mathcal{L}' \subseteq \mathcal{L}$ that are hit by $\mathcal{S}_1$, and (iii) for the remaining segments $\mathcal{L} \setminus \mathcal{L}'$, we compute $\mathcal{S}_2$ in $O(1)$ amortized time as described below.

**Initialization step:** For the first event $\delta_1 \in \mathcal{D}_\theta$, we apply the algorithm of Section 2 to compute $\mathcal{S}_2$. This also identifies the segments $\ell_a, \ell_b, \ell_c, \ell_d, \ell_p, \ell_q, \ell_r, \ell_s \in \mathcal{L} \setminus \mathcal{L}'$ as defined in Lemma 1. This needs $O(n)$ time.

**Iterative step:** Below, we show that, after processing $\delta_i \in \mathcal{D}_\theta$, when we process $\delta_{i+1} \in \mathcal{D}_\theta$ in order, at most one among the eight segments $\ell_a, \ell_b, \ell_c, \ell_d, \ell_p, \ell_q, \ell_r, \ell_s \in \mathcal{L} \setminus \mathcal{L}'$ for $\mathcal{S}_2$ (see the eight situations in Fig. 8), may change, and it can be obtained in $O(1)$ time.

In Fig. 8(a), if $\mathcal{S}_1$ is increased to $\mathcal{S}_1'$ (dotted square), then none of the 8 segments of $\mathcal{S}_2$ gets changed.

In Fig. 8(a), if $\mathcal{S}_1$ is increased to $\mathcal{S}_1''$ (dashed square), then $\ell_d$ of $\mathcal{S}_2$ gets changed, which can be obtained by scanning $\mathcal{L}_t$ array.

In Fig. 8(b) $\ell_a$ of $\mathcal{S}_2$ gets changed, which can be obtained by scanning $\mathcal{L}_r$ array.

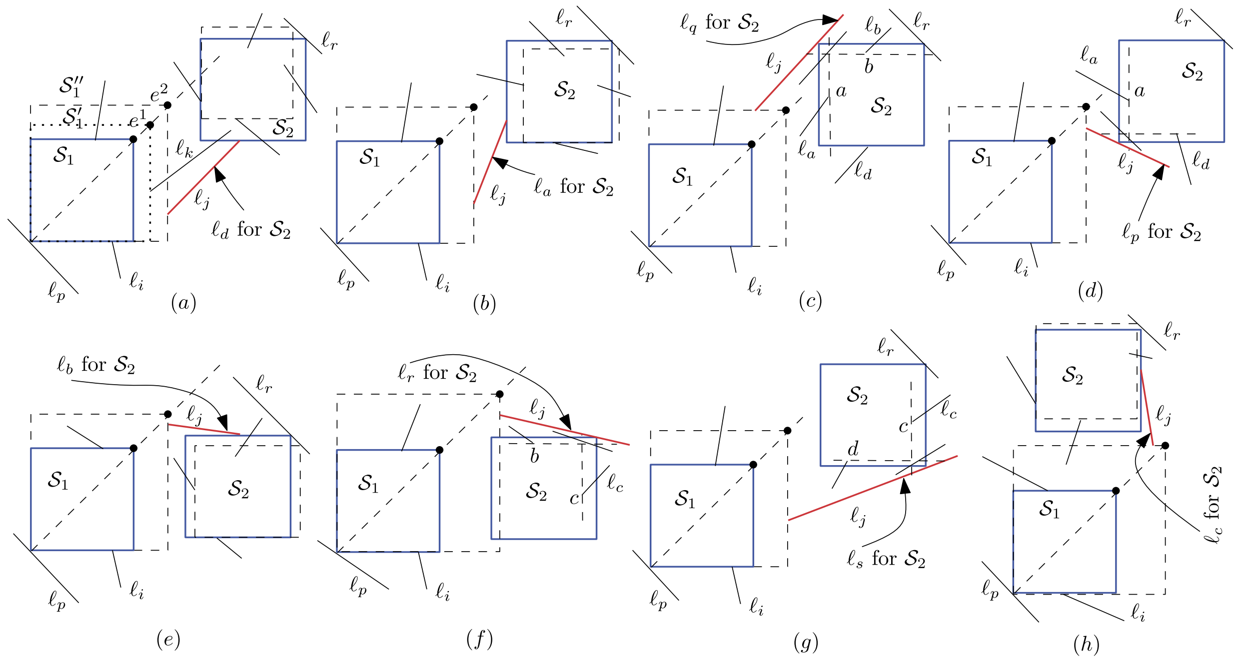

**Fig. 8.** Demonstration of Iterative steps of computing $\mathcal{S}_2$ for different elements of $\mathcal{D}_\theta$.

In Fig. 8(c) $\ell_q$ of $\mathcal{S}_2$ gets changed, which can be obtained by scanning $\mathcal{I}^{v1}(a)$ array.
In Fig. 8(d) $\ell_p$ of $\mathcal{S}_2$ gets changed, which can be obtained by scanning $\mathcal{I}^{v2}(a)$ array.
In Fig. 8(e) $\ell_b$ of $\mathcal{S}_2$ gets changed, which can be obtained by scanning $\mathcal{L}_b$ array.
In Fig. 8(f) $\ell_r$ of $\mathcal{S}_2$ gets changed, which can be obtained by scanning $\mathcal{I}^{v1}(c)$ array.
In Fig. 8(g) $\ell_s$ of $\mathcal{S}_2$ gets changed, which can be obtained by scanning $\mathcal{I}^{v2}(c)$ array.
In Fig. 8(h) $\ell_c$ of $\mathcal{S}_2$ gets changed, which can be obtained by scanning $\mathcal{L}_l$ array.

The processing of all the elements in $\mathcal{D}_\theta$ needs exactly one scan of the arrays $\mathcal{L}_b$, $\mathcal{L}_r$, $\mathcal{L}_t$, $\mathcal{L}_l$, $\mathcal{I}^{v1}(\tau)$, $\mathcal{I}^{v2}(\tau)$, $\mathcal{I}^{h}(\tau)$, $\mathcal{I}^{d}(\tau)$, $\mathcal{I}^{v1}(\tau')$, $\mathcal{I}^{v2}(\tau')$. Thus, we can compute the required $\mathcal{S}_2$ for each element in $\delta \in \mathcal{D}_\theta$ in amortized $O(1)$ time. The generation of the instances in B2 are similar to that of B1. To generate the instances of B3 with the segment $\ell_j$ on its right boundary, we need to consider a vertical line $V_j$ at the left end-point on $\ell_j$, and include the horizontal projection of the bottom end-point of all the segments in $\mathcal{L} \setminus \{\ell_p\}$ on $V_j$ provided the concerned bottom end-points lie to the left of $V_j$ and above the left end-point of $\ell_j$. For all the segments in $\mathcal{L}$ with negative slope that intersects $V_j$ above the left end-point of $\ell_j$, we include those points of intersection in $V_j$. We also include the left end-point of $\ell_j$ as an event in $V_j$. These events can be generated in $O(n)$ time using the array $\mathcal{L}_b$. For each of these events the corresponding $\mathcal{S}_1$ square and hence the corresponding $\mathcal{S}_2$ square are well-defined. The $\mathcal{S}_2$ squares for all the events in $V_j$ can also be computed in $O(n)$ time. Thus, we have the following theorem:

**Theorem 1.** *If $\mathcal{R}_{abcd}$ does not hit all the line segments in $\mathcal{L}$, we can compute the optimal axis parallel square pair $(\mathcal{S}_1, \mathcal{S}_2)$ that combinedly hit all the segments in $\mathcal{L}$ in $O(n^2)$ time.*

**Proof.** Lemma 5 says that if the $\mathcal{S}_1$ square is defined by one line segment in $\mathcal{L} \setminus \{\ell_p\}$, we can compute the optimum pair of squares $(\mathcal{S}_1, \mathcal{S}_2)$ in $O(n^2)$ time. The instances where $\mathcal{S}_1$ is defined by two line segments in $\mathcal{L} \setminus \{\ell_p\}$, are classified into three cases B1, B2, B3. For handling the case B1, we created $O(n)$ events on $\ell_p$ in the array $\mathcal{C}$ in $O(n)$ time using the $\mathcal{L}_t$ array. These correspond to the bottom left corners of possible $\mathcal{S}_1$. For each event $\theta \in C$, we create another array $\mathcal{D}_\theta$ with $O(n)$ sub-events; each of them may be the top-right corners of $\mathcal{S}_1$ square whose bottom-left corner is $\theta$. We can process these $O(n)$ events in $\mathcal{D}_\theta$ in amortized $O(n)$ time. Thus, all possible instances of type B1 can be generated in $O(n^2)$ time. Similarly, all possible instances of type B2 also can be generated in $O(n^2)$ time. Regarding the instances of type B3, we need to consider the left end-points of all the $O(n)$ segments in $\mathcal{L}$. As mentioned earlier, the number of events (top-right corner of $\mathcal{S}_1$ squares) generated is $O(n)$, and they can be processed in amortized $O(n)$ time. In special case of B3 (see Fig. 6(f)), both the top and right boundaries of the square $\mathcal{S}_1$ is touched by a segment $\ell_i$, and the corresponding $\mathcal{S}_2$ can be determined in $O(n)$ time. Since there are $n$ such line segments $\ell_i \in \mathcal{L}$, the total time complexity result for identifying all such instances is also $O(n^2)$. Thus the result follows. □

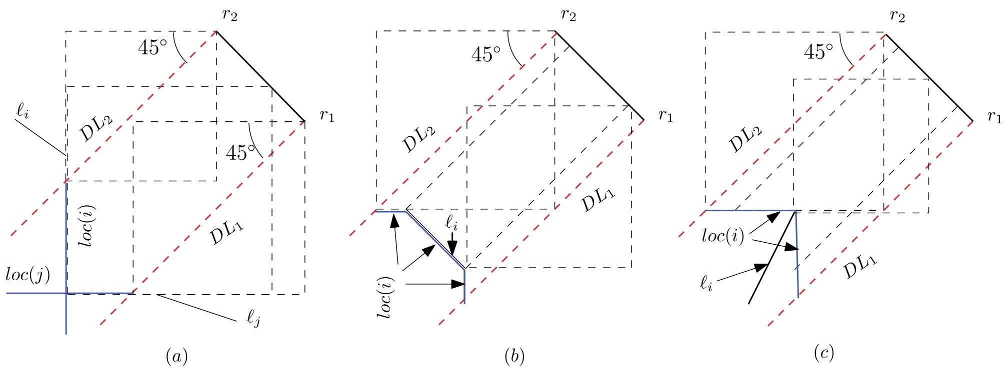


**Fig. A.1.** The locus $loc(i)$ of the bottom left corner of square that hits the segment $l_i$.

## Appendix A

**Size of (i.e. the number of segments in) $loc(i)$, $i = \{a, p, q, d, s, (b, c)\}$:** The $loc(i)$ is the locus of the "bottom-left" corner of a minimum sized square $\mathcal{S}^r$ which hits the line segment $\ell_i$, while its "top-right" corner moves along the segment $\overline{r_2 r_1}$ (Fig. 2(a) demonstrates $loc(s)$). The $loc(i)$ (within the strip $\Gamma$ bounded by the line $DL_2$ and $DL_1$ of unit slope passing through $r_2$ and $r_1$ respectively) is as follows:

- If the segment $\ell_i$ (resp. $\ell_j$) lies above $DL_2$ (resp. below $DL_1$), then the required locus will be a vertical line (resp. horizontal line) inside the strip $\Gamma$ (see Fig. A.1(a)).
- If $\ell_i$ lies inside the strip $\Gamma$, then there are two possibilities:
  (a) Slope of $\ell_i$ is negative (see Fig. A.1(b)): The required locus will be a horizontal segment passing through the top end-point of $\ell_i$ (to the left of it), until the bottom-left corner of the square coincides with the top end-point of $\ell_i$; then it will move along $\ell_i$ till the bottom end-point of $\ell_i$ is reached, and finally it will be vertically downwards, until it hits the boundary of $\Gamma$.
  (b) Slope of $\ell_i$ is positive (see Fig. A.1(c)): The required locus will be a horizontal segment as in case (a) until the bottom-left corner of square hits the top end-point of $\ell_i$, then finally it will be vertically downwards, until the boundary of $\Gamma$ is hit.
- If $\ell_i$ intersects the boundary of $\Gamma$, then also we can construct the required locus in a similar way as in the aforesaid cases.

Thus, in all the situations $loc(i)$ consists of at most three segments within $\Gamma$, where at most one of them is non-axis-parallel.

## References


[1] S. Sadhu, S. Roy, S.C. Nandy, S. Roy, Linear time algorithm to cover and hit a set of line segments optimally by two axis-parallel squares, Theor. Comput. Sci. 769 (2019) 63–74.