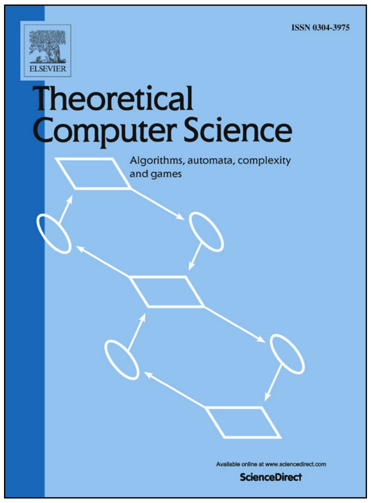


# Randomized approximation algorithms for Planar visibility counting problem

Sharareh Alipour, Mohammad Ghodsi, Amir Jafari

| | |
|---|---|
| PII: | S0304-3975(17)30737-5 |
| DOI: | https://doi.org/10.1016/j.tcs.2017.10.009 |
| Reference: | TCS 11347 |

To appear in: *Theoretical Computer Science*

| | |
|---|---|
| Received date: | 29 April 2016 |
| Revised date: | 28 September 2017 |
| Accepted date: | 12 October 2017 |

# Randomized approximation algorithms for Planar visibility counting problem

Sharareh Alipour Mohammad Ghodsi Amir Jafari

October 16, 2017

**Abstract**

Given a set $S$ of $n$ disjoint line segments in $\mathbb{R}^2$, the visibility counting problem (VCP) is to preprocess $S$ such that the number of segments in $S$ visible from any query point $p$ can be computed quickly. This problem can be solved trivially in $O(\log n)$ query time using $O(n^4 \log n)$ preprocessing time and $O(n^4)$ space.

Gudmundsson and Morin in (2010), proposed a 2-approximation algorithm for this problem with a tradeoff between the space and the query time. For any constant $0 \le \alpha \le 1$, their algorithm answers any query in $O_\epsilon(m^{(1-\alpha)/2})$ time with $O_\epsilon(m^{1+\alpha})$ of preprocessing time and space, where $\epsilon > 0$ is a constant that can be made arbitrarily small and $O_\epsilon(f(n)) = O(f(n)n^\epsilon)$ and $m = O(n^2)$ is a number that depends on the configuration of the segments.

In this paper, we propose two randomized approximation algorithms for VCP. The first algorithm depends on two constants $0 \le \beta \le \frac{2}{3}$ and $0 < \delta \le 1$, and the expected preprocessing time, the expected space, and the expected query time are $O(m^{2-3\beta/2} \log m)$, $O(m^{2-3\beta/2})$, and $O(\frac{1}{\delta^2} m^{\beta/2} \log m)$, respectively. The algorithm, in the preprocessing phase, selects a sequence of random samples, whose size and number depend on the tradeoff parameters. When a query point $p$ is given by an adversary unaware of the random sample of our algorithm, it computes the number of visible segments from $p$, denoted by $m_p$, exactly, if $m_p \le \frac{3}{\delta^2} m^{\beta/2} \log(2m)$. Otherwise, it computes an approximated value, $m'_p$, such that with the probability of at least $1-\frac{1}{m}$, we have $(1-\delta)m_p \le m'_p \le (2+2\delta)m_p$. The preprocessing time and space of the second algorithm are $O(n^2 \log n)$ and $O(n^2)$, respectively. This algorithm computes the exact value of $m_p$ if $m_p \le \frac{1}{\delta^2}\sqrt{n} \log n$, otherwise it returns an approximated value $m''_p$ in expected $O(\frac{1}{\delta^2}\sqrt{n} \log n)$ time, such that with the probability at least $1 - \frac{1}{\log n}$, we have $(1-3\delta)m_p \le m''_p \le (1.5+3\delta)m_p$.

**Keywords.** computational geometry, visibility, randomized algorithm, approximation algorithm, graph theory.

## 1 Introduction

*Problem Statement*

Let $S = \{s_1, s_2, \dots, s_n\}$ be a set of $n$ disjoint closed line segments in the plane contained in a bounding box $\mathbb{B}$. Two points $p$ and $q$ in the bounding box are visible to each other with respect to $S$, if the open line segment $\overline{pq}$ does not intersect any segments of $S$. A segment $s_i \in S$ is said to be visible from a point $p$, if there exists a point $q \in s_i$ such that $q$ is visible from $p$. *The visibility counting problem (VCP)* is to find $m_p$, the number of segments of $S$ visible from a query point $p$. Throughout this paper, we assume that the configuration of line segments and the query point is in a general position. That is, no three end-points or the query point and two end-points are colinear.

*The visibility polygon of a given point* $p \in \mathbb{B}$ (*i.e.* $p$ is inside the bounding box) is defined as

$$VP_S(p) = \{q \in \mathbb{B} : p \text{ and } q \text{ are visible}\},$$

and *the visibility polygon of a given segment* $s_i$ is defined as

$$VP_S(s_i) = \bigcup_{q \in s_i} VP_S(q).$$

Consider the $2n$ end-points of the segments of $S$ as vertices of a geometric graph. Add a straight-line-edge between each pair of visible vertices. The result is *the visibility graph of* $S$ or $VG(S)$. To construct *the extended visibility graph of* $S$ or $EVG(S)$ from $VG(S)$, we continue each edge of $VG(S)$ in both directions until it hits some other segments in $S$ or the bounding box. This creates at most two new vertices and two new edges for each edge of $VG(S)$. Adding all these vertices and edges to $VG(S)$, we get $EVG(S)$. The extended visibility graph, reflects all the visibility information from which the visibility polygon of any segment $s_i \in S$ can be computed [12].

## *Related Work*

The visibility polygon of a point $p$, $VP_S(p)$, can be computed in $O(n \log n)$ time using $O(n)$ space [5, 17]. Vegter proposed an output sensitive algorithm that reports $VP_S(p)$ in $O(|VP_S(p)| \log(\frac{n}{|VP_S(p)|}))$ time, by preprocessing the segments in $O(m \log n)$ time using $O(m)$ space, where $m = O(n^2)$ is the number of edges of $VG(S)$ and $|VP_S(p)|$ is the number of vertices of $VP_S(p)$ [18].

We can use $EVG(S)$ to solve VCP. One can compute $EVG(S)$ optimally in $O(n \log n + m)$ time [10]. If a vertex is assigned to any intersection point of the edges of $EVG(S) \bigcup S$, we have a planar graph, which is called the planar arrangement of the edges of $EVG(S)$. All points in any face of this arrangement have the same number of visible segments and this number can be computed for each face in the preprocessing step [12]. Since, there are $O(n^4)$ faces in the planar arrangement of $EVG(S)$, a point location structure of size $O(n^4)$ can answer each query in $O(\log n)$ time. But, $O(n^4)$ preprocessing time and space is high. We also know that for any query point $p$, by computing $VP_S(p)$, $m_p$ can be computed in $O(n \log n)$ with no preprocessing. This has led to several results with a tradeoff between the preprocessing cost and the query time [4, 6, 11, 16, 19].

There are two approximation algorithms for VCP by Fischer *et al.* [8, 9]. One of these algorithms, for any $1 \leq r \leq n$, uses a data structure of size $O((m/r)^2)$ to build a $(r/m)$-cutting for $EVG(S)$ by which the queries are answered in $O(\log n)$ time with an absolute error of $r$ compared to the exact answer. The second algorithm uses a random sampling method to build a data structure of size $O((m^2 \log^{O(1)} n)/l)$ to answer any query in $O(l \log^{O(1)} n)$ time, where $1 \leq l \leq n$. In the latter method, the answer of VCP is approximated up to an absolute value of $\delta n$ for any constant $\delta > 0$ ($\delta$ affects the constant factor of both data structure size and the query time).

In [17], Suri and O'Rourke represent the visibility polygon of a segment by a union of triangles. Gudmundsson and Morin [12] improved the covering scheme of [17]. Their method builds a data structure of size $O_\epsilon(m^{1+\alpha}) = O_\epsilon(n^{2(1+\alpha)})$ in $O_\epsilon(m^{1+\alpha}) = O_\epsilon(n^{2(1+\alpha)})$ preprocessing time, from which each query is answered in $O_\epsilon(m^{(1-\alpha)/2}) = O_\epsilon(n^{1-\alpha})$ time, where $0 \leq \alpha \leq 1$. This algorithm returns $m'_p$ such that $m_p \leq m'_p \leq 2m_p$. The same result can be achieved from [2] and [14]. In [2], it is proven that the number of visible end-points of the segments in $S$ from $p$, denoted by $ve_p$, is a 2-approximation of $m_p$, that is $m_p \leq ve_p \leq 2m_p$.

*Our Results*

In this paper, first we present a randomized $(2+2\delta)$-approximation algorithm, where $0 < \delta \leq 1$. The expected preprocessing time and space of our algorithm are $O(m^{2-3\beta/2}\log m)$ and $O(m^{2-3\beta/2})$ respectively, and our expected query time is $O(\frac{1}{\delta^2}m^{\beta/2}\log m)$, where $0 \leq \beta \leq \frac{2}{3}$ is chosen arbitrarily in the preprocessing time. Next, we present another randomized algorithm. This algorithm computes the exact value of $m_p$ if $m_p \leq \frac{1}{\delta^2}\sqrt{n}\log n$, otherwise it returns an approximated value $m''_p$ in expected $O(\sqrt{n}\log n)$ time, such that with probability at least $1-\frac{1}{\log n}$, we have: $(1-\delta)m_p \leq m''_p \leq (1.5+\delta)m_p$.

In the second proposed algorithm, a graph $G(p)$ is associated to each query point $p$. The construction of $G(p)$ is explained in Section 2. It will be shown that $G(p)$ has a planar embedding.

Using Euler's Formula for planar graphs, we will show that if $p$ is inside a bounded face of $G(p)$, then $m_p = ve_p - C(G(p)) + 1$, otherwise $m_p = ve_p - C(G(p))$, where $C(G(p))$ is the number of connected components of $G(p)$ and $ve_p$ is the number of visible end-points of the segments in $S$ from $p$. In Section 3 and 4, we will present algorithms to approximate $ve_p$ and $C(G(p))$. This leads to an overall approximation for $m_p$. A preliminary and less detailed version of this paper was presented in [3].

Table 1 compares the performance of our algorithms with the best known result for this problem. Note that if we choose a constant number $0 < \delta \leq 1$, then our expected query time is better than [12], however our algorithm returns a $(2+2\delta)$-approximation of the answer with a high probability. Our query time, space and preprocessing time are in expectation, but their method is deterministic. Note that $m = O(n^2)$ and one can easily express the running times in terms of $n$. But, the advantage of having the running times in terms of $m$ appears when $m$ is small relative to $n^2$. The second algorithm returns a $(1.5+3\delta)$-approximation, the time complexity of this algorithm depends on $n$. Since replacing $2\delta$ and $3\delta$ with $\delta$ does not change the order of the running time, space and preprocessing time, for simplicity in the table we use $\delta$.

To clarify our randomization approach of the first algorithm for counting the number of visible end-points, let us explain it in few words. Given an input set of $n$ segments and tradeoff parameters $0 < \beta \leq \frac{2}{3}$ and $0 < \delta \leq 1$, in the preprocessing phase, we select a sequence of random samples, whose size and number depend on these parameters. Now, if a query point $p$ is given, using these samples, we calculate exactly the number of visible end-points and according to the size and the number of the samples produce an approximated value for $m_p$. Hence, the randomization is only in the preprocessing phase and not in the query phase. If a query is given by an adversary unaware of the random samples selected by our algorithm, this method results in an approximated value with given approximation factor with a probability that approaches zero as the size of input increases. When the random sample is known to the adversary, there is a possibly large part of the region, that if the adversary repeatedly chooses the query point from that region, the correctness with high probability claimed, will be ruined. However we hope that, in real applications, such phenomena rarely happen. In the second algorithm for counting the number of visible end-points and in the algorithm for counting the number of connected components, the randomization occurs in the query phase and the adversary can not affect the outcome of the algorithm as before.

Table 1: Comparison of our method and the best known result for VCP. Note that $0 \leq \beta \leq \frac{2}{3}$ is chosen in the preprocessing time and $0 < \delta \leq 1$. Our first algorithm returns an approximated answer with probability $1 - \dfrac{1}{m}$ and our second algorithm returns an approximated answer with probability $1 - \dfrac{1}{\log n}$. The running time and space are in expectation in the first algorithm. In our second algorithm only the query time is in expectation.

| Reference | Preprocessing time | Space | Query | Approx-Factor |
|---|---|---|---|---|
| [12] | $O_\epsilon(m^{2-3\beta/2})$ | $O_\epsilon(m^{2-3\beta/2})$ | $O_\epsilon(m^{3\beta/4})$ | 2 |
| Our result | $O(m^{2-3\beta/2}\log m)$ | $O(m^{2-3\beta/2})$ | $O(\frac{1}{\delta^2}m^{\beta/2}\log m)$ | $2+\delta$ |
| Our result | $O(n^2\log n)$ | $O(n^2)$ | $O(\sqrt{n}\log n)$ | $1.5+\delta$ |

## 2 Definitions and the main theorems

For each point $a' \in s_i$, let $\overrightarrow{pa'}$ be the ray emanating from the query point $p$ toward $a'$ and let $a = pr(a')$ be the first intersection point of $\overrightarrow{pa'}$ and a segment in $S$ or the bounding box right after touching $a'$. We say that $a = pr(a')$ is covered by $a'$ or the projection of $a'$ is $a$. Also, suppose that $\overline{x'y'}$ is a subsegment of $s_i$ and $\overline{xy}$ is a subsegment of $s_j$, such that $pr(x') = x$ and $pr(y') = y$ and for any point $z' \in \overline{x'y'}$, $pr(z') \in \overline{xy}$, then we say that $\overline{xy}$ is covered by $\overline{x'y'}$.

For each query point $p$, we construct a multi-graph denoted by $G(p)$ as follows: a vertex $v_i$ is associated to each segment $s_i \in S$, and an undirected edge between $v_i$ and $v_j$ is put if $s_j$ covers one end-point of $s_i$ (or vice-versa; that is, if $s_i$ covers one end-point of $s_j$). Obviously, there are two edges between $v_i$ and $v_j$, if $s_j$ (or $s_i$) covers both end-points of $s_i$ (or $s_j$). As an example, refer to Fig. 1.(a) and (d). Note that the bounding box is not considered in the construction of $G(p)$.

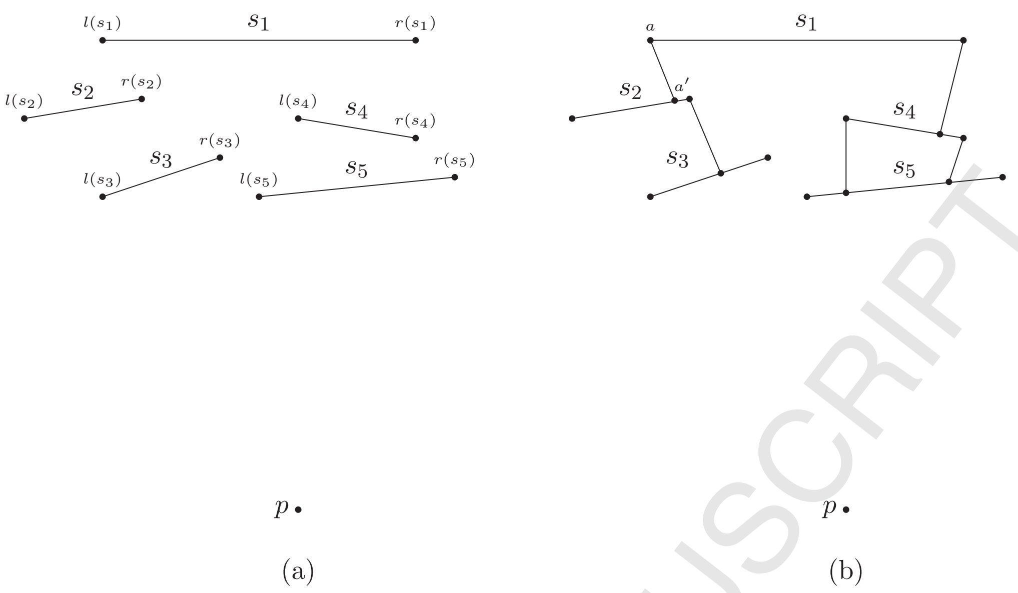
For any segment $s \in S$, let $l(s)$ and $r(s)$ be the first and second end-points of $s$, respectively swept by a ray around $p$ in clockwise order (Fig. 1.(a)).

**Lemma 2.1.** *$G(p)$ has a planar embedding.*

*Proof.* A planar embedding for $G(p)$ is as follows. For each end-point $a \in s_i$ not visible from $p$, there is a point $a' \in s_j$ such that $pr(a') = a$. Draw the segment $\overline{aa'}$. Doing this, we have a collection of non-intersecting segments. For each $s_k \in S$, we create a vertex $v_k$ located very close to the mid-point of $s_k$, on the side of $p$. Also, for each segment $\overline{aa'}$ as above, we connect $a$ to $v_i$ and $a'$ to $v_j$. This creates an edge consisting of three consecutive straight-line segments $\overline{v_ia}$, $\overline{aa'}$, and $\overline{a'v_j}$ that connects $v_i$ to $v_j$. Obviously, none of these edges intersect. Finally, all the original segments are removed. The remaining is the vertices and edges of a planar embedding of $G(p)$ (These steps can be seen in Fig. 1). ∎

From now on, we use this planar embedding of the graph $G(p)$. Euler's Formula for any non-connected planar graph $G$ with multiple edges is:

$$V(G) - E(G) + F(G) = 1 + C(G),$$



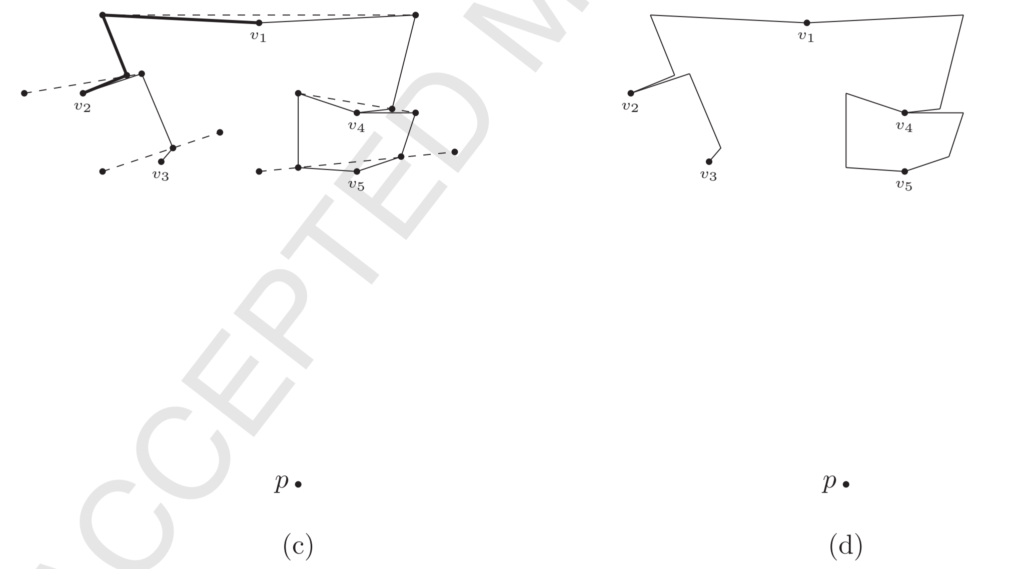


Figure 1: The steps to draw a planar embedding of $G(p)$. (a) The segments are $s_1, \ldots, s_5$ with their left and right end-points and a given query point is $p$. (b) For each end-point $a \in s_i$ not visible to $p$ and $a' \in s_j$ such that $pr(a') = a$, we draw $\overline{aa'}$. (c) Put a vertex $v_i$ for each segment $s_i$ in a distance sufficiently close to the middle of $s_i$ on the side of $p$. For each $a$ and $a'$ (described in (b)), connect $a$ to $v_i$ and $a'$ to $v_j$. This creates an edge between $v_i$ and $v_j$ (shown by thick lines) (d) Remove the segments and the remaining is the planar embedding of $G(p)$. Note that the final embedding has 5 vertices and 5 edges and each edge is drawn as a chain of 3 straight-line segments.

where $E(G)$, $V(G)$, $F(G)$, and $C(G)$ are the number of edges, vertices, faces, and connected components of $G$, respectively. The following theorem provides a method to calculate $m_p$, using $G(p)$.

**Theorem 2.1.** *The number of segments not visible from p is equal to $F(G(p)) - 2$ if p is inside a bounded face of $G(p)$, or is equal to $F(G(p)) - 1$, otherwise.*

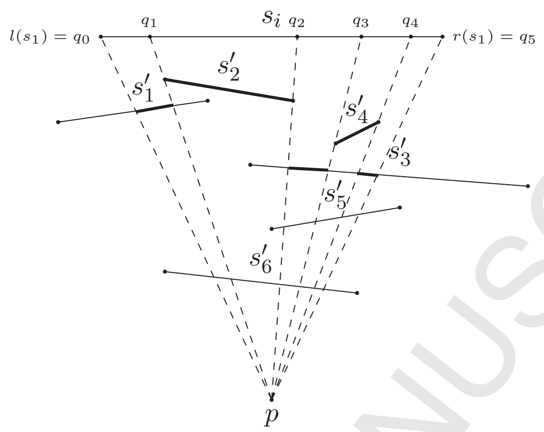


Figure 2: The segment $s_i$ is not visible from $p$. It can be partitioned into 5 subsegments $\overline{q_0q_1}, \overline{q_1q_2}, \overline{q_2q_3}, \overline{q_3q_4}$, and $\overline{q_4q_5}$, each one is covered respectively by subsegment of $s'_1, s'_2, s'_3, s'_4$, and $s'_3$ shown above.

*Proof.* We construct a bijection $\phi$ between the segments not visible from $p$ to the faces of $G(p)$ except the unbounded face and the face that contains $p$. This will complete the proof of our theorem.

Suppose that $s_i$ is a segment not visible from $p$. Then, we can partition $s_i$ into $k$ subsegments, $\overline{q_0q_1}, \overline{q_1q_2}, \overline{\ldots, q_{k-1}q_k}$ such that $q_0 = l(s_i)$, $q_k = r(s_i)$, and for each $\overline{q_iq_{i+1}}$, there is a subsegment $\overline{q'_iq'_{i+1}} \subseteq s_j$ that covers $\overline{q_iq_{i+1}}$. Let $s'_1, s'_2, \ldots, s'_k$ be the set of segments such that $\overline{xy} \subseteq s'_{i+1}$ covers $\overline{q_iq_{i+1}}$ (note that some segments may appear more than once in the above sequence) (Fig. 2). We claim that the vertices $v_i, v'_1, v'_2, \ldots, v'_k$ corresponding to $s_i, s'_1, s'_2, \ldots, s'_k$ form a bounded face of $G(p)$ that does not contain $p$. In $\phi$, we associate this face to $s_i$. Since $v'_1$ is the vertex associated to the first segment that covers $\overline{q_0q_1}$, $s'_1$ covers $l(s_i)$ and $v_i$ is adjacent to $v'_1$. Similarly, since $s'_k$ covers $r(s_i)$, $v_i$ is adjacent to $v'_k$. The next subsegment that covers a subsegment of $s_i$ comes from $s'_2$. This means that $r(s'_1)$ is covered by $s'_2$ or $l(s'_2)$ is covered by $s'_1$. This implies that $v'_1$ is adjacent to $v'_2$. Similarly, we can show that $v'_i$ is adjacent to $v'_{i+1}$ for all $1 \leq i < k$. To complete the construction, we need to show that the closed path formed by $v_i \to v'_1 \to v'_2, \cdots \to v'_k \to v_i$ is a bounded face not containing $p$. Consider a ray around $p$ in clockwise order. The area that this ray touches under $s_i$ and above $s'_1, \ldots, s'_k$ is a region bounded by $v_i, v'_1, v'_2, \ldots, v'_k$. Obviously, $p$ is not inside this region. To show that the loop constructed above constitute a face of $G(p)$, note that since we take all the segments which cover parts of $s_i$, the region bounded by this loop is empty, *i.e.* it contains no other segment or rays from $p$ other than those on its boundary. This shows that even if we have other edges that correspond to the vertices of our loop, all these edges must be outside this region. Hence, this loop is a face of $G(p)$.

Now, we show that our map $\phi$ is one-to-one and onto. The proof of injectivity is easier. If $\phi(s_i) = \phi(s_j)$, then according to the construction of $\phi$, a subsegment of $s_i$ covers a subsegment of $s_j$ and a subsegment of $s_j$ covers a subsegment of $s_i$. This is a contradiction since these segments do not intersect. To prove the surjectivity, we need to show for any bounded face $f$ that does not contain $p$, there is a vertex $v_i$ corresponding to a segment $s_i$ that is not visible to $p$ such that $\phi(s_i) = f$.

To find $s_i$, we use the sweeping ray around $p$. Since $f$ is assumed to be bounded and not containing $p$, the face $f$ is between two rays from $p$; one from the left and the other from the right. If we start sweeping from left to right, there is a segment $s_i$ corresponding to a vertex of $f$ whose end-point is the first to be covered by the other segments corresponding to the vertices of $f$. We claim that $s_i$ is the desired segment , *i.e.* $s_i$ is not visible to $p$ and $\phi(s_i) = f$. For example in Fig. 2, the closed path $v_i \to v'_1 \to v'_2 \to v'_3 \to v'_4, \to v'_3, \to v_i$ forms a face and $s_i$ is the first segment among $\{s_i, s'_1, s'_2, s'_3, s'_4\}$ such that $l(s_i)$ is covered by one of the segments in $\{s_i, s'_1, s'_2, s'_3, s'_4\}$.

Obviously, $l(s_i)$ is not visible from $p$. Also, $v'_1$ is adjacent to $v_i$ which means that a subsegment of $s'_1$ covers a subsegment of $s_i$. Since $v'_1$ and $v'_2$ are adjacent, this means that a subsegment of $s'_2$ consecutively covers the next subsegment of $s_i$ right after $s'_1$. Continuing this procedure, we conclude that a subsegment of each $s'_i$ covers some subsegment of $s_i$ continuously right after $s'_{i-1}$. Also, $v'_k$ and $v_i$ are adjacent, so $r(s_i)$ is not visible from $p$. We conclude that subsegments of $s'_1, s'_2 \dots, s'_k$ completely cover $s_i$ and hence $s_i$ is not visible from $p$.

So, if $p$ is in the unbounded face of $G(p)$, the number of segments which are not visible from $p$ is $F(G(p)) - 1$, otherwise it is $F(G(p)) - 2$. ■

Euler's Formula is used to compute $F(G(p))$. Obviously, $V(G(p))$ is $n$. For each end-point not visible from $p$, an edge is added to $G(p)$; therefore, $E(G(p))$ is $2n - ve_p$ ( Note that, $ve_p$ was defined above as the number of visible end-points from $p$). Euler's Formula and Theorem 2.1 indicate the following lemma.

**Lemma 2.2.** *If $p$ is inside a bounded face of $G(p)$, then $m_p = ve_p - C(G(p)) + 1$, otherwise, $m_p = ve_p - C(G(p))$.*

In the rest of this paper, three algorithms are presented; two algorithms to approximate $ve_p$ and another one to approximate $C(G(p))$. By applying Lemma 2.2, an approximated value of $m_p$ is calculated. The main result of this paper is thus derived from the following two theorems.

**Theorem 2.2.** *For any $0 < \delta \leq 1$ and $0 \leq \beta \leq \frac{2}{3}$, VCP can be approximated in $O(\frac{1}{\delta^2} m^{\beta/2} \log m)$ expected query time, using $O(m^{2-3\beta/2} \log m)$ expected preprocessing time and $O(m^{2-3\beta/2})$ expected space. This algorithm returns a value $m'_p$ such that with the probability at least $1 - \frac{1}{m}$, $(1-\delta)m_p \leq m'_p \leq (2+2\delta)m_p$ when $m_p > \frac{3}{\delta^2} m^{\beta/2} \log(2m)$ and returns the exact value when $m_p \leq \frac{3}{\delta^2} m^{\beta/2} \log(2m)$.*

**Theorem 2.3.** *For any $0 < \delta \leq 1$, VCP can be approximated in $O(\frac{1}{\delta^2}\sqrt{n} \log n)$ expected query time, using $O(n^2 \log n)$ preprocessing time and $O(n^2)$ space. This algorithm returns a value $m''_p$ such that with the probability at least $1 - \frac{1}{\log n}$, $(1-3\delta)m_p \leq m'_p \leq (1.5+3\delta)m_p$ when $m_p > \frac{1}{\delta^2}\sqrt{n} \log n$ and returns the exact value when $m_p \leq \frac{1}{\delta^2}\sqrt{n} \log n$.*

## 3 Two approximation algorithms to compute the number of visible end-points

In this section, we present two algorithms to approximate $ve_p$, the number of visible end-points. Both algorithms are similar.

### 3.1 The first algorithm to approximate the number of visible end-points

In the preprocessing phase, we build the data structure of the algorithm presented in [18] which calculates $VP_S(p)$ in $O(|VP_S(p)|\log(n/|VP_S(p)|))$ time, where $|VP_S(p)|$ is the number of vertices of $VP_S(p)$. In [18], the algorithm for computing $VP_S(p)$, consists of a rotational sweep of a line around $p$. During the sweep, the subsegments visible from $p$ along the sweep-line are collected. In the preprocessing phase, we choose a fixed parameter $\beta$, where $0 \leq \beta \leq \frac{2}{3}$. In the query time we also choose a fixed parameter $0 < \delta \leq 1$ where $2+\delta$ is the value of approximation factor of the algorithm.

We use the algorithm presented in [18] to find the visible end-points, but for any query point, we stop the algorithm if more than $\frac{3}{\delta^2}m^{\beta/2}\log(2m)$ of the visible end-points are found.

If the sweep line completely sweeps around $p$ before counting $\frac{3}{\delta^2}m^{\beta/2}\log(2m)$ of the visible end-points, then we have completely computed $VP_S(p)$ and we have $|VP_S(p)| \leq \frac{3}{\delta^2}m^{\beta/2}\log(2m)$. In this case, the number of visible segments can be calculated exactly in $O(\frac{1}{\delta^2}m^{\beta/2}\log m)$ time. Otherwise, $ve_p > \frac{3}{\delta^2}m^{\beta/2}\log(2m)$ and the answer is calculated in the next step of algorithm, that we now explain.

Let $m(a)$ be the number of edges of $EVG(S) \cup S$ incident to an end-point $a$. The visibility polygon of $a$ is a star shaped polygon consisting of $m(a)+1 = O(n)$ non-overlapping triangles [5, 17], which are called *the visibility triangles of* $a$, denoted by $VT_S(a)$. The query point $p$ is visible to an end-point $a$, if and only if it lies inside one of the visibility triangles of $a$. Let $VTE_S$ be the set of visibility triangles of all the end-points of the segments in $S$. Then, the number of visible end-points from $p$ is the number of triangles in $VTE_S$ containing $p$. We can construct $VTE_S$ in $O(m\log m) = O(n^2\log n)$ time using $EVG(S)$ because to construct the visibility triangles of each end-point we need the edges of $EVG(S)$ that are incident to that end-point and these edges can be extracted from $EVG(S)$. Notice that $|VTE_S| = O(m) = O(n^2)$.

We can preprocess a given set of triangles using the following lemma to count the number of triangles containing any query point.

**Lemma 3.1.** *Let $\Delta$ be a set of $m$ triangles. There exists a data structure of size $O(m^2)$, such that in the preprocessing time of $O(m^2\log m)$, the number of triangles containing a query point $p$ can be calculated in $O(\log m)$ time.*

*Proof.* Consider the planar arrangement of the edges of the triangles in $\Delta$ as a planar graph. Let $f$ be a face of this graph. Then, for any pair of points $p$ and $q$ in $f$, the number of triangles containing $p$ and $q$ are equal. Therefore, we can compute these numbers for each face in a preprocessing phase and then, for any query point locate the face containing that point. There are $O(m^2)$ faces in the planar arrangement of $\Delta$, so a point location structure of size $O(m^2)$ can answer each query in $O(\log m)$ time as in [13]. Note that, in the preprocessing time we compute the exact value of an arbitrary face in $O(m)$, then we move to an adjacent face through an edge $e$ of this planar graph. The difference of these two adjacent faces depends on the triangles that contain $e$ as a part of their edges. We can find each of these triangles in $O(\log m)$ and compute the number of that face. We continue moving to adjacent faces until visiting all of them.

So, if an edge of a triangle is divided into $t$ parts in the planar graph, then it has to be considered at most $t$ times. Since the total number of edges of this planar graph is $O(m^2)$, then in $O(m^2 \log m)$, we can calculate the numbers associated to all the faces. ∎

## 3.2 The algorithm

Here, we present an algorithm to approximate $ve_p$. We use this algorithm when the first attempt for finding the exact value of $ve_p$, using the algorithm of [18], does not finish in $\frac{3}{\delta^2} m^{\beta/2} \log(2m)$ steps. In the preprocessing phase, we take a random subset $RVTE_1 \subset VTE_S$ such that each member of $VTE_S$ is chosen with the probability of $\frac{1}{m^\beta}$.

**Lemma 3.2.** $E(|RVTE_1|) = O(m^{1-\beta})$.

*Proof.* Let $VTE_S = \{\Delta_1, \Delta_2, \ldots, \Delta_{m'}\}$, where $m' = O(m) = O(n^2)$ and $X_i = 1$ if $\Delta_i \in RVTE_1$, and $X_i = 0$ otherwise. We have,

$$E(|RVTE_1|) = E(\textstyle\sum_{i=1}^{m'} X_i) = \sum_{i=1}^{m'} E(X_i) = \sum_{i=1}^{m'} \frac{1}{m^\beta} = \frac{m'}{m^\beta} = O(m^{1-\beta}).$$

∎

Suppose that in the preprocessing time, we choose $m^{\beta/2}$ independent random subsets $RVTE_1, \ldots, RVTE_{m^{\beta/2}}$ as above of $VTE_S$. Using Lemma 3.1, for any query point $p$, the number of triangles of each $RVTE_i$ containing $p$ denoted by $(ve_p)_i$, is calculated in $O(\log m)$ time by $O(m^{2-2\beta} \log m)$ expected preprocessing time and $O(m^{2-2\beta})$ expected space. Then, $ve'_p = m^\beta \frac{\sum_{i=1}^{m^{\beta/2}} (ve_p)_i}{m^{\beta/2}}$ is returned as the approximation value of $ve_p$.

## 3.3 Analysis of the approximation factor

In this section the approximation factor of the algorithm will be given. The following lemma is trivial. Let $Z_i = m^\beta (ve_p)_i$.

**Lemma 3.3.** *$E(Z_i) = ve_p$ and therefore $E(\frac{\sum_{i=1}^{m^{\beta/2}} Z_i}{m^{\beta/2}}) = ve_p$.*

So, $Z_1, Z_2, \ldots, Z_{m^{\beta/2}}$ are random variables with $E(Z_i) = ve_p$. We use the following well known lemma:

**Lemma 3.4.** *(Chernoff's Lemma) Given a sequence $X_1, X_2, \ldots, X_N$ of i.i.d. Bernoulli random variables with finite expected value $E(X_1) = E(X_2) = \cdots = \mu$ and for any $0 < \delta < 1$, we have,*

$$P(|X_1 + \cdots + X_N - N\mu| > N\mu\delta) \leq 2\exp(-N\mu\delta^2/3).$$

**Lemma 3.5.** *With the probability at least $1 - \frac{1}{m}$ we have,*

$$(1-\delta)ve_p \leq ve'_p \leq (1+\delta)ve_p.$$

*Proof.* Let $X_{i,1}, \ldots, X_{i,ve_p}$ be the random variables for the $ve_p$ visibility triangles of the end-points that contain the query point $p$. That is $X_{i,j} = 1$ if the triangle $\Delta_j$ is in the $i$th sample and is zero otherwise. Then $(ve_p)_i$ is $(X_{i,1} + \cdots + X_{i,ve_p})$. If $|ve'_p - ve_p| > \delta ve_p$, then

$$|\sum_{i=1}^{m^{\beta/2}} \sum_{j=1}^{ve_p} X_{i,j} - m^{-\beta/2} ve_p| > m^{-\beta/2} ve_p \delta.$$

Using Lemma 3.4 with $\mu = \frac{1}{m^\beta}$ and $N = m^{\beta/2} ve_p$ we have:

$$\mathbb{P} = P(|ve'_p - ve_p| > \delta ve_p) \leq 2\exp(-m^{-\beta/2} ve_p \delta^2/3).$$

We know that $ve_p \geq \frac{3}{\delta^2} m^{\beta/2} \log(2m)$, so

$$\mathbb{P} = P(|ve'_p - ve_p| > \delta ve_p) \leq \tfrac{1}{m}.$$

With the probability of at least $1 - \mathbb{P}$, we have,

$$(1-\delta) ve_p \leq ve'_p \leq (1+\delta) ve_p.$$

Also, for a large $m$, we have $\mathbb{P} \sim 0$.

■

## 3.4 Analysis of time and space complexity

In the first step of the query time, we run the algorithm of [18]. The preprocessing time and space for constructing the data structure of [18] are $O(m \log m)$ and $O(m)$, respectively, which computes $VP_S(p)$ in $O(|VP_S(p)| \log(n/|VP_S(p)|))$ time. As we run this algorithm for at most $\frac{3}{\delta^2} m^{\beta/2} \log(2m)$ steps, the query time of the first step is $O(\frac{1}{\delta^2} m^{\beta/2} \log m)$.

According to Lemma 3.2, $E(|RVTE_i|) = O(m^{1-\beta})$. Using Lemma 3.1, the expected preprocessing time and space for each $RVTE_i$ are $O(m^{2-2\beta} \log m)$ and $O(m^{2-2\beta})$ respectively, such that in $O(\log m)$ we can calculate $(ve_p)_i$. So, the expected preprocessing time and space are $m^{\beta/2} O(m^{2-2\beta} \log m) = O(m^{2-\frac{3}{2}\beta} \log m)$ and $m^{\beta/2} O(m^{2-2\beta}) = O(m^{2-\frac{3}{2}\beta})$ respectively.

In the second step, for each $RVTE_i$ the value of $(ve_p)_i$ is calculated in $O(\log m)$. Therefore, the expected query time is $O(\frac{1}{\delta^2} m^{\beta/2} \log m) + O(m^{\beta/2} \log m)$. According to [2], $ve_p$ is a 2 approximation answer for VCP, therefore, $ve'_p$ is a $2 + 2\delta$ approximation answer for VCP as well. Theorem 2.2 is derived.

## 3.5 Second algorithm to approximate the number of visible end-points

Now, we introduce another method to approximate the number of visible end-points, $ve_p$. This will be used in the next section to produce a $(1.5 + 3\delta)$-approximation for $m_p$. First, we state the following theorem.

**Theorem 3.1.** *[15] Given a set of $n$ disjoint line segments $S$, we can preprocess the segments using $O(n^2 \log n)$ time, and $O(n^2)$ space such that for a given query point $p$ and a direction $\vec{d}$, the first segment in $S$ intersected by ray shot from $p$ in the direction $\vec{d}$, is determined in $O(\log n)$ time.*

So, by Theorem 3.1 we can answer if an end-point is visible from a query point $p$ in $O(\log n)$ time. Similar to the previous algorithm, first we run the algorithm of [18] for $\frac{1}{\delta^2}\sqrt{n} \log n$ steps, if it does not terminate then $\frac{1}{\delta^2}\sqrt{n} \log n < m_p \leq ve_p$.
Now, we choose a random sample from $2n$ end-points by choosing each end-point randomly and independently with probability $\frac{1}{n^{\frac{2}{3}}}$, to produce a sample of expected size $2n^{\frac{1}{3}}$. We count the number of visible end-points in this sample and multiply it with $n^{\frac{2}{3}}$ to report $(ve_p)_1$. Note that $E((ve_p)_1) = ve_p$. To compute $Var((ve_p)_1)$, we use the fact that $(ve_p)_1$ is $n^{\frac{2}{3}}(X_1 + \cdots + X_{ve_p})$ where $X_i$'s are independent Bernoulli random variables which are 1 with probability $n^{-\frac{2}{3}}$ and 0 with probability $1 - n^{-\frac{2}{3}}$. So, $Var(X_i) = n^{-\frac{2}{3}}(1 - n^{-\frac{2}{3}})$ and $Var((ve_p)_1) = n^{\frac{4}{3}} ve_p n^{-\frac{2}{3}}(1 - n^{-\frac{2}{3}}) < n^{\frac{2}{3}} ve_p$.

We use the following well known lemma:

**Lemma 3.6.** *(Chebyshev's Lemma) Given a sequence $X_1, \dots, X_N$ of i.i.d. random variables with $E(X_i) = \mu$ and $\delta > 0$, we have:*

$$P(|\frac{X_1 + \cdots + X_N}{N} - \mu| > \delta) \leq \frac{Var(X_1)}{N\delta^2}.$$

We repeat the process of random sampling for $n^{\frac{1}{6}}$ times and report $ve'_p = \frac{(ve_p)_1 + \cdots + (ve_p)_{n^{\frac{1}{6}}}}{n^{\frac{1}{6}}}$. We show that $ve'_p$ is a $1 + \delta$ approximation of $ve_p$ with probability at least $1 - \frac{1}{\log n}$. In fact:

$$P(|ve'_p - ve_p| > \delta ve_p) \leq \frac{n^{\frac{2}{3}} ve_p}{n^{\frac{1}{6}} ve_p^2 \delta^2} \leq \frac{1}{\log n}.$$

Where we have used the fact that $ve_p > \frac{1}{\delta^2}\sqrt{n}\log n$ for the last inequality. This shows with the probability at least $1 - \frac{1}{\log n}$ we have:

$$(1 - \delta) ve_p \leq ve'_p \leq (1 + \delta) ve_p.$$

For this algorithm according to Theorem 3.1, the preprocessing time and space needed is $O(n^2 \log n)$ and $O(n^2)$ respectively, and the expected query time is $O(n^{\frac{1}{6}} n^{\frac{1}{3}} \log n) = O(\sqrt{n} \log n)$.

## 4 An algorithm to approximate $C(G(p))$ and the proof of Theorem 2.3

In this section, we explain an algorithm to estimate the number of connected components of $G(p)$. For simplicity, by a component, we mean a connected component of $G(p)$.

Let $c$ be a component such that $p$ is not inside any of its faces. Without loss of generality, we can assume that $p$ lies below $c$. It is easy to see that there exist rays emanating from $p$ that do not intersect any segments corresponding to the vertices of $c$. We start sweeping one of these rays in a clockwise direction. Let $L(c)$ (left end-point of $c$) be the first end-point of a segment of $c$ and $R(c)$ (right end-point of $c$) be the last end-point of a segment of $c$ that intersect this ray (Fig. 3). This way, every component $c$ has $L(c)$ and $R(c)$ except for the component that contains $p$ in one of its faces. Also, note that $L(c)$ and $R(c)$ do not depend on the choice of the starting ray and $L(c)(R(c))$ is always a left(right) end-point of a segment $s$, $l(s)(r(s))$, that corresponds to a vertex of $c$. For example in Fig. 3, $c_1 = \{s_1, s_2, s_6\}$ is a component with three vertices and $L(c_1) = l(s_2)$, $R(c_1) = r(s_6)$ and they both project to the bounding box.

Now, we introduce the following lemma.

**Lemma 4.1.** *For each component $c$, except for the one with a face containing $p$, the projections of $L(c)$ and $R(c)$ are either on the same segment or both are on the bounding box.*

*Proof.* Let $c$ be a component with left and right end-points $L(c)$ and $R(c)$. Assume that one of these end-points, say $L(c)$ projects on a segment $s$. We claim that the right end-point of $s$ is to the right of $R(c)$. Otherwise, this end-point is covered by another segment $t$, whose left end-point should be to the left of $R(c)$, and is to the right of $L(c)$, since otherwise $L(c)$ can not project on $s$. Therefore the left end-point of $t$ is also not visible from $p$ and is covered by another segment $t_1$. Similarly, the left end-point of $t_1$ is between $L(c)$ and $R(c)$ (with respect to the rotating ray from $p$) and hence, it is not visible from

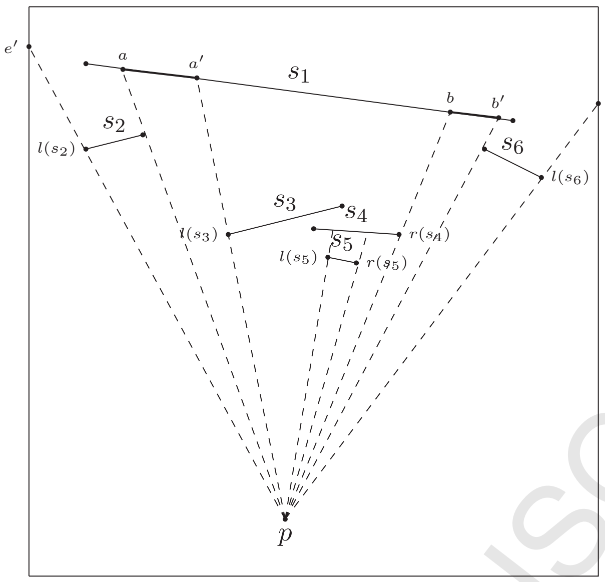


Figure 3: In this figure, $\overline{aa'}$ and $\overline{bb'}$ are the visible subsegments of $s_1$. The bounding box has one visible part from $e$ to $e'$. Here, $G(p)$ has three components; $c_1 = \{s_1, s_2, s_6\}$, $c_2 = \{s_3, s_4\}$, and $c_3 = \{s_5\}$. Also, $L(c_1) = l(s_2)$, $L(c_2) = l(s_3)$, and $L(c_3) = l(s_5)$. $R(c_1) = r(s_6)$, $R(c_2) = r(s_4)$, and $R(c_3) = r(s_5)$.

$p$. So, it should be covered by another segment $t_2$ and hence the process produces an infinite sequence of segments, leading to a contradiction. Since, we showed that the right end-point of $s$ is to the right of $R(c)$, $R(c)$ can not project on the bounding box and assume it projects on a segment $t$. We want to show that $t = s$. An argument identical to the one given above, shows that the left end-point of $t$ is to the left of $L(c)$ or the point of $s$ that $L(c)$ projects to. If $s$ and $t$ are different, the fact that the left end-point of $t$ is to the left of $pr(L(c))$ on $s$ and the right end-point of $s$ is to the right of $pr(R(c))$ on $t$ implies that they intersect which is clearly a contradiction. Notice that this argument also shows that if one of the two end-points $L(c)$ or $R(c)$ projects on the bounding box, the other one also projects on the bounding box. Since, we showed that if one of them project on a segment, the other one should project on the same segment. ■

Since the left and right end-point of a component are visible, every component has at least one visible segment, so $C(G(p)) \leq m_p$. If $C_1$ is the number of components with only one visible segment and $C_2$ is the number of components with more than one visible segment then, since $C(G(p)) = C_1 + C_2$ and $C_2 \leq \frac{m_p}{2}$, we have:

$$C(G(p)) - \frac{m_p}{2} \leq C_1 \leq C(G(p)).$$

For example in Fig. 3, $\{s_1, s_2, s_6\}$ is a component with three visible segments and $\{s_5\}$ is a component with only one visible segment. Using Lemma 2.2 and ignoring the additive constant one, we get

$$m_p \leq ve_p - C_1 \leq 1.5 m_p.$$

Hence, if we have a $(1 + \delta)$ approximation for the values of $ve_p$ and $C_1$, we will get a $(1.5 + \delta)$ approximation for $m_p$. In the next subsection we propose an algorithm to approximate the value of $C_1$ that uses random sampling of the segments.

## 4.1 Algorithm

In this subsection, we propose an algorithm to give a $(1.5+3\delta)$-approximation solution for $m_p$. To start, we run the algorithm of [18] for $\frac{1}{\delta^2}\sqrt{n}\log n$ steps, if it terminates, then we have an exact answer for $m_p$. Otherwise, we run the following algorithm.

Our goal is to estimate $C_1$, the number of components with only one visible segment. Any component $c$ with only one visible segment, corresponds to a segment that both of its end-points are visible and their projections are on the same segment or on the bounding box. This follows from Lemma 4.1. In fact, the two end-points of the only visible segment of $c$ are the left and right end-points of $c$. To estimate $C_1$, we need to estimate the number of such segments. Let $s_1, \ldots, s_{C_1}$ be these segments. We choose a random sample of segments by choosing any segment with probability $\frac{1}{n^{2/3}}$. We use the algorithm of Theorem 3.1 to answer if an end-point $a$ is visible from a query point $p$ or not and also to find the projection of $a$ by shooting a ray from $a$ in the direction of $\vec{pa}$.

For each segment in our sample, using $O(\log n)$ time, we can decide if both of its end-points are visible and if both of them project onto the same segment or the bounding box. This way we can count the number of components with only one visible segment in that sample, say $C_{1,1}$ in $O(n^{\frac{1}{3}}\log n)$ expected time. We repeat taking these samples for $n^{\frac{1}{6}}$ times, and report $C_1' = n^{\frac{2}{3}}\frac{C_{1,1}+\cdots+C_{1,n^{\frac{1}{6}}}}{n^{\frac{1}{6}}}$ as the approximated value of $C_1$. Then, since $E(C_{1,i}) = \frac{1}{n^{\frac{2}{3}}}C_1$ and $Var(C_{1,i}) = n^{\frac{4}{3}}C_1 n^{-\frac{2}{3}}(1-n^{-\frac{2}{3}}) \le n^{\frac{2}{3}}C_1$, by Chebyshev's lemma:

$$P(|C_1' - C_1| > \delta m_p) \le \frac{n^{\frac{2}{3}}C_1}{n^{\frac{1}{6}}m_p^2\delta^2} \le \frac{1}{\log n}$$

where for the last inequality, we used the fact that $m_p > \frac{1}{\delta^2}\sqrt{n}\log n$ and $C_1 \le m_p$. Therefore, with the probability at least $1 - \frac{1}{\log n}$, we have:

$$C_1 - \delta m_p \le C_1' \le C_1 + \delta m_p.$$

Now, by the algorithm of Subsection 3.5, we get $ve_p'$ for the visible end-points such that $(1-\delta)ve_p \le ve_p' \le (1+\delta)ve_p$. Therefore:

$$(1-\delta)ve_p - C_1 - \delta m_p \le ve_p' - C_1' \le (1+\delta)ve_p - C_1 + \delta m_p.$$

If we use the facts that $m_p \le ve_p - C_1 \le 1.5m_p$ and $ve_p \le 2m_p$, we can derive:

$$(1-3\delta)m_p \le ve_p' - C_1' \le (1.5+3\delta)m_p.$$

## 4.2 Analysis of time and space complexity

In the first phase, we run the algorithm of [18] for $O(\frac{1}{\delta^2}\sqrt{n}\log n)$ steps. In the second phase, the expected number of segments in each sample is $n^{\frac{1}{3}}$ and the number of samples is $n^{\frac{1}{6}}$. In $O(\log n)$ we can check whether a segment is the only visible segment of a component. So, the overall expected query time is $O(\frac{1}{\delta^2}\sqrt{n}\log n)$. The preprocessing time and space of the algorithm of [18] are $O(n^2\log n)$ and $O(n^2)$ and the space and preprocessing time of Theorem 3.1 are $O(n^2\log n)$ and $O(n^2)$. So, the overall preprocessing time and space are $O(n^2\log n)$ and $O(n^2)$.

So, if we return $ve_p' - C_1'$, we have a $(1.5+3\delta)$-approximation for $m_p$. Therefore, Theorem 2.3 is proved.

## 5 Conclusion

In this paper, two randomized algorithms are proposed to compute an approximation answer for VCP. The main ideas of these algorithms that reduce the complexity of previous methods are random sampling and breaking the query into two steps. The time and space complexity of the first algorithm depend on the size of $EVG(S)$. In the second algorithm, a planar graph is associated to each query point $p$. It is proven that the answer is equal to $ve_p - C(G(p))$ or $ve_p - C(G(p)) + 1$, where $ve_p$ is the number of visible end-points and $C(G(p))$ is the number of connected components in this planar graph. To improve the running time of our algorithm instead of finding the exact values of $ve_p$ and $C(G(p))$, we approximate these values. It is possible to compute the exact value of $ve_p$ but, computing the exact value of $C(G(p))$ with a tradeoff between the query time and the space is a challenging problem.

## References

[1] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, Advances in Discrete and Computational Geometry, volume 223 of Contemporary Mathematics, pages 1–56. American Mathematical Society Press, 1999.

[2] Alipour, S., Zarei, A.: Visibility Testing and Counting. FAW-AAIM 2011, Jinhua, China, LNCS (Volume 6681) by Springer-Verlag, 343-351 (2011)

[3] Alipour, S., Ghodsi, M., Jafari, A.: An improved Constant-Factor Approximation Algorithm for Planar Visibility Counting Problem. International Computing and Combinatorics Conference, Springer International Publishing, 209–221 (2016)

[4] Aronov, B., Guibas, L. J., Teichmann M. and Zhang L.: Visibility queries and maintenance in simple polygons. Discrete and Computational Geometry. 27, 461–483 (2002)

[5] Asano, T.: An efficient algorithm for finding the visibility polygon for a polygonal region with holes. IEICE Transactions. 557–589(1985)

[6] Bose, P., Lubiw, A. and Munro, J. I.: Efficient visibility queries in simple polygons. Computational Geometry Theory and Applications. 23(7), 313–335(2002)

[7] Bondy, J. A., Murty, U. S. R.: Graph theory with applications (Vol. 290). London: Macmillan (1976)

[8] Fischer, M., Hilbig, M., Jahn, C., Meyer auf der Heide F. and Ziegler M.: Planar visibility counting. CoRR, abs/0810.0052. (2008)

[9] Fischer, M., Hilbig, M., Jahn, C., Meyer auf der Heide F. and Ziegler M.: Planar visibility counting. In Proceedings of the 25th European Workshop on Computational Geometry(EuroCG 2009).203–206(2009)

[10] Ghosh, S. K. and Mount, D.: An output sensitive algorithm for computing visibility graphs. SIAM Journal on Computing. 20, 888–910 (1991)

[11] Ghosh, S. K.: Visibility algorithms in the plane. Cambridge university press. (2007)

[12] Gudmundsson, J., Morin, P.: Planar visibility: testing and counting. Annual Symposium on Computational Geometry. 77–86 (2010)

[13] Kirkpatrick, D.:Optimal search in planar subdivisions. SIAM Journal on Computing.12(1), 28–35(1983)

[14] Nouri, M. and Ghodsi, M.: Space/query-time tradeoff for computing the visibility polygon. Computational Geometry. 46(3), 371–381 (2013)

[15] Pocchiola, M.: Graphics in Flatland revisited. In Scandinavian Workshop on Algorithm Theory(1990) pp. 85–96. Springer Berlin Heidelberg. Chicago (1990)

[16] Pocchiola, M. and Vegter, G.: The visibility complex. International Journal of Computational Geometry and Applications. 6(3), 279–308 (1996)

[17] Suri, S. and O'Rourke, J.: Worst-case optimal algorithms for constructing visibility polygons with holes. In Proceedings of the Second Annual Symposium on Computational Geometry (SCG 86), 14–23(1986)

[18] Vegter, G.: The visibility diagram: A data structure for visibility problems and motion planning. In: Gilbert, J.R., Karlsson, R. (eds.) SWAT(1990). LNCS, 447, pp. 97–110. Springer, Heidelberg (1990)

[19] Zarei, A. and Ghodsi, M.: Efficient computation of query point visibility in polygons with holes. In Proceedings of the 21st Annual ACM Symposium on Computational Geometry. (SCG 2005). (2005).