Constant Approximation Algorithms for Guarding Simple Polygons using Vertex Guards

Pritam Bhattacharya*[1], Subir Kumar Ghosh[2], and Sudebkumar Prasant Pal[1]

[1]Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur, West Bengal - 721302, India.
[2]Department of Computer Science, RKM Vivekananda Educational and Research Institute, Belur, West Bengal - 711202, India.

Abstract

The art gallery problem enquires about the least number of guards sufficient to ensure that an art gallery, represented by a polygon P, is fully guarded. Most standard versions of this problem are known to be NP-hard. In 1987, Ghosh provided a deterministic $\mathcal{O}(\log n)$-approximation algorithm for the case of vertex guards and edge guards in simple polygons. In the same paper, Ghosh also conjectured the existence of constant ratio approximation algorithms for these problems. We present here three polynomial-time algorithms with a constant approximation ratio for guarding an n-sided simple polygon P using vertex guards. (i) The first algorithm, that has an approximation ratio of 18, guards all vertices of P in $\mathcal{O}(n^4)$ time. (ii) The second algorithm, that has the same approximation ratio of 18, guards the entire boundary of P in $\mathcal{O}(n^5)$ time. (iii) The third algorithm, that has an approximation ratio of 27, guards all interior and boundary points of P in $\mathcal{O}(n^5)$ time. The significance of our results lies in the fact that these results settle the conjecture by Ghosh regarding the existence of constant-factor approximation algorithms for this problem, which has been open since 1987 despite several attempts by researchers. Our approximation algorithms exploit several deep visibility structures of simple polygons which are interesting in their own right.

1 Introduction

1.1 The art gallery problem and its variants

The art gallery problem enquires about the least number of guards sufficient to ensure that an art gallery (represented by a polygon P) is fully guarded, assuming that a guard's field of view covers 360° as well as unbounded distance. This problem was first posed by Victor Klee in a conference in 1973, and has become a well investigated problem in computational geometry.

A *polygon* P is defined to be a closed region in the plane bounded by a finite set of line segments, called edges of P, such that between any two points of P, there exists a path which does not intersect any edge of P. If the boundary of a polygon P consists of two or more cycles, then P is called a *polygon with holes*. Otherwise, P is called a *simple polygon* or a *polygon without holes*.

An art gallery can be viewed as an n-sided polygon P (with or without holes) and guards as points inside P. Any point $z \in P$ is said to be *visible* from a guard g if the line segment zg does not intersect the exterior of P. In general, guards may be placed anywhere inside P. If the guards are allowed to be placed only on vertices of P, they are called *vertex guards*. If there is no such restriction, then they are called *point guards*. Point and vertex guards together are also referred to as *stationary guards*. If guards are allowed to patrol along a line segment inside P, they are called *mobile guards*. If they are allowed to patrol only along the edges of P, they are called *edge guards* [12, 23].

*Email: pritam.bhattacharya@cse.iitkgp.ernet.in, lord.pritomose@gmail.com

In 1975, Chvátal [5] showed that $\lfloor \frac{n}{3} \rfloor$ stationary guards are sufficient and sometimes necessary for guarding a simple polygon. In 1978, Fisk [10] presented a simpler and more elegant proof of this result. For a simple orthogonal polygon, whose edges are either horizontal or vertical, Kahn et al. [17] and also O'Rourke [22] showed that $\lfloor \frac{n}{4} \rfloor$ stationary guards are sufficient and sometimes necessary.

1.2 Related hardness and approximation results

The decision version of the art gallery problem is to determine, given a polygon P and a number k as input, whether the polygon P can be guarded with k or fewer guards. This problem was first shown to be NP-complete for polygons with holes by O'Rourke and Supowit [24]. This problem was also shown to be NP-complete for simple polygons for guarding using vertex guards by Lee and Lin [21]. Their proof was generalized to work for point guards by Aggarwal [1]. The problem is NP-hard even for simple orthogonal polygons as shown by Katz and Roisman [18] and Schuchardt and Hecker [25]. Each one of these hardness results hold irrespective of whether we are dealing with vertex guards, edge guards, or point guards.

In 1987, Ghosh [11, 13] provided a deterministic $\mathcal{O}(\log n)$-approximation algorithm for the case of vertex and edge guards by discretizing the input polygon P and treating it as an instance of the Set Cover problem. As pointed out by King and Kirkpatrick [19], newer methods for improving the approximation ratio of the Set Cover problem itself have been developed in the time after Ghosh's algorithm was published. By applying these methods, the approximation ratio of Ghosh's algorithm becomes $\mathcal{O}(\log OPT)$ for guarding simple polygons and $\mathcal{O}(\log h \log OPT)$ for guarding a polygon with h holes, where OPT denotes the size of the smallest guard set for P. Deshpande et al. [6] obtained an approximation factor of $\mathcal{O}(\log OPT)$ for point guards or perimeter guards by developing a sophisticated discretization method that runs in pseudo-polynomial time. Efrat and Har-Peled [7] provided a randomized algorithm with the same approximation ratio that runs in fully polynomial expected time. Recently, Bonnet and Miltzow [4] obtained an approximation factor of $\mathcal{O}(\log OPT)$ for the point guard problem assuming integer coordinates and a specific general position. For guarding simple polygons using perimeter guards, King and Kirkpatrick [19] designed a deterministic $\mathcal{O}(\log \log OPT)$-approximation algorithm in 2011.

In 1998, Eidenbenz, Stamm and Widmayer [8, 9] proved that the problem is APX-complete, implying that an approximation ratio better than a fixed constant cannot be achieved unless NP = P. They also proved that if the input polygon is allowed to contain holes, then there cannot exist a polynomial time algorithm for the problem with an approximation ratio better than $((1-\epsilon)/12)\ln n$ for any $\epsilon > 0$, unless NP $\subseteq$ TIME$(n^{\mathcal{O}(\log \log n)})$. Extending their method, Bhattacharya, Ghosh and Roy [2] recently proved that, even for the special subclass of polygons with holes that are weakly visible from an edge, there cannot exist a polynomial time algorithm for the problem with an approximation ratio better than $((1-\epsilon)/12)\ln n$ for any $\epsilon > 0$, unless NP = P. These inapproximability results establish that the approximation ratio of $\mathcal{O}(\log n)$ obtained by Ghosh in 1987 is in fact the best possible for the case of polygons with holes. However, for simple polygons, the existence of a constant factor approximation algorithm for vertex and edge guards was conjectured by Ghosh [11, 14] in 1987.

Ghosh's conjecture has been shown to be true for vertex guarding in two special sub-classes of simple polygons, viz. monotone polygons and polygons weakly visible from an edge. In 2012, Krohn and Nilsson [20] presented an approximation algorithm that computes in polynomial time a guard set for a monotone polygon P, such that the size of the guard set is at most $30 \cdot OPT$. Recently, Bhattacharya, Ghosh and Roy [2, 3] presented a 6-approximation algorithm that runs in $\mathcal{O}(n^2)$ time for vertex guarding polygons that are weakly visible from an edge.

1.3 Our contributions

In this paper, we present three polynomial-time algorithms with a constant approximation ratio for guarding an n-sided simple polygon P using vertex guards. The first algorithm, that has an approximation ratio of 18, guards all vertices of P in $\mathcal{O}(n^4)$ time. The second algorithm, that has the same approximation ratio of 18, guards the entire boundary of P in $\mathcal{O}(n^5)$ time. The third algorithm, that has an approximation ratio of 27, guards all interior and boundary points of P in $\mathcal{O}(n^5)$ time. The significance of our results lies in the fact that these results settle the *long-standing conjecture by Ghosh* [11] regarding the existence of constant-factor approximation algorithms for this problem, which has been open since 1987 despite several attempts by researchers.

In each of our algorithms, P is first partitioned into a hierarchy of *weak visibility polygons* according to the *link distance* from a starting vertex (see Figure 2). This partitioning is very similar to the *window partitioning* given by Suri [26, 27] in the context of computing minimum link paths. Then, starting with the farthest level in the hierarchy (i.e. the set of weak visibility polygons that are at the maximum link distance from the starting vertex), the entire hierarchy is traversed backward level by level, and at each level, vertex guards (of two types, viz. *inside* and *outside*) are placed for guarding every weak visibility polygon at that level of P. At every level, a novel procedure is used that has been developed for placing guards in (i) a simple polygon that is weakly visible from an internal chord, or (ii) a union of overlapping polygons that are weakly visible from multiple disjoint internal chords. Note that these chords are actually the constructed edges introduced during the hierarchical partitioning of P.

Due to partitioning according to link distances, guards can only see points within the adjacent weak visibility polygons in the hierarchical partitioning of P. This property locally restricts the visibility of the chosen guards, and thereby ensures that the approximation bound on the number of vertex guards placed by our algorithms at any level leads directly to overall approximation bounds for guarding P. Thus, a constant factor approximation bound on the overall number of guards placed by our algorithms is a direct consequence of choosing vertex guards in a judicious manner for guarding each collection of overlapping weak visibility polygons obtained from the hierarchical partitioning of P. Our algorithms exploit several deep visibility structures of simple polygons which are interesting in their own right.

1.4 Organization of the paper

In Section 2, we introduce some preliminary definitions and notations that are used throughout the rest of the paper. In Section 3, we present the hierarchical partitioning of a simple polygon P into weak visibility polygons. Next, in Section 4, we describe how the algorithm traverses the hierarchy of visibility polygons, starting from the farthest level, and uses the procedures from Section 5 at each level as a sub-routine for guarding P. In Section 5, we present a novel procedure for placing vertex guards necessary for guarding a simple polygon Q that is weakly visible from a single internal chord uv or from multiple disjoint chords. In Section 6, we establish the overall approximation ratios for the three approximation algorithms. Finally, in Section 7, we conclude the paper with a few remarks.

2 Preliminary definitions and notations

Let P be a simple polygon. Assume that the vertices of P are labelled $v_1, v_2, \ldots, v_n$ in clockwise order. Let $\mathcal{V}(P)$ denote the set of all vertices. Let $bd_c(p, q)$ (or $bd_{cc}(p, q)$) denote the clockwise (respectively, counterclockwise) boundary of P from a vertex p to another vertex q. Note that by definition, $bd_c(p, q) = bd_{cc}(q, p)$. Also, we denote the entire boundary of P by $bd(P)$. So, $bd(P) = bd_c(p, p) = bd_{cc}(p, p)$ for any chosen vertex p belonging to P.

The *visibility polygon* of P from a point z, denoted as $\mathcal{VP}(z)$, is defined to be the set of all points of P that are visible from z. In other words, $\mathcal{VP}(z) = \{q \in P : q \text{ is visible from } z\}$. Observe that the boundary of $\mathcal{VP}(z)$ consists of polygonal edges and non-polygonal edges. We refer to the non-polygonal edges as *constructed edges*. Note that one point of a constructed edge is a vertex (say, v_i) of P, while the other point (say, u_i) lies on $bd(P)$. Moreover, the points z, v_i and u_i are collinear (see Figure 1).

Let bc be an internal chord or an edge of P. A point q of P is said to be *weakly visible* from bc if there exists a point $z \in bc$ such that q is visible from z. The set of all such points of P is said to be the *weak visibility polygon* of P from bc, and denoted as $\mathcal{VP}(bc)$. If $\mathcal{VP}(bc) = P$, then P is said to be *weakly visible from* bc. Like $\mathcal{VP}(z)$, the boundary of $\mathcal{VP}(bc)$ also consists of polygonal edges and constructed edges v_iu_i (see Figure 1). If z (or bc) does not belong to $bd_c(v_iu_i)$, then v_iu_i is called a *left constructed edge* of $\mathcal{VP}(z)$ (respectively, $\mathcal{VP}(bc)$). Otherwise, v_iu_i is called a *right constructed edge*.

Let v_iu_i be a constructed edge of $\mathcal{VP}(bc)$ (or $\mathcal{VP}(z)$). Observe that v_iu_i divides P into two subpolygons. One of the subpolygons is bounded by $bd_c(v_i, u_i)$ and v_iu_i, whereas the other one is bounded by $bd_{cc}(v_i, u_i)$ and v_iu_i. Out of these two, the subpolygon that does not contain bc (respectively, z) is referred to as the *pocket* of v_iu_i, and is denoted by $P(v_iu_i)$ (see Figure 1). If v_iu_i is a left (or right) constructed edge, then $P(v_iu_i)$ is called a *left pocket* (respectively, *right pocket*).

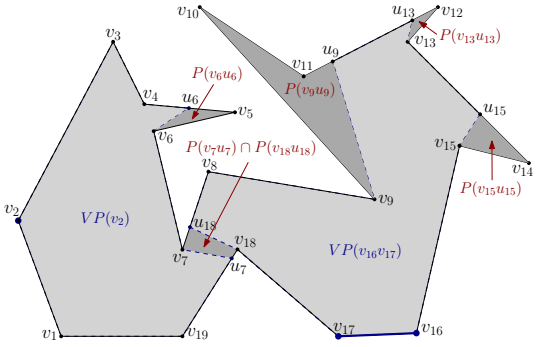

Figure 1: Figure showing visibility polygon $\mathcal{VP}(v_2)$ and weak visibility polygon $\mathcal{VP}(v_{16}v_{17})$, along with several pockets created by constructed edges belonging to both.

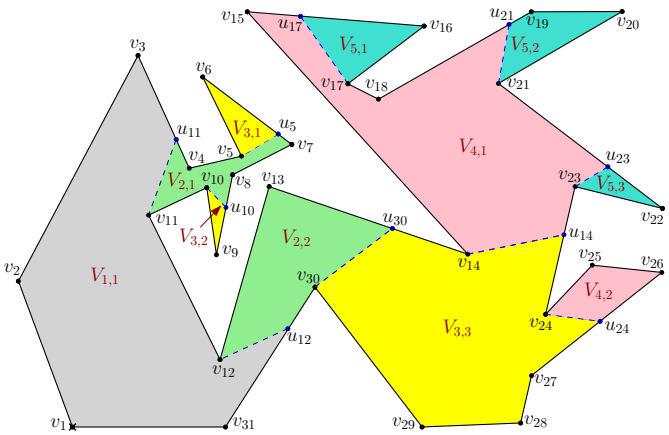

Figure 2: Figure showing the partitioning of a simple polygon into visibility windows.

Let $SP(s, t)$ define the Euclidean shortest path from a point s to another point t within P. The *shortest path tree* of P rooted at any point s of P, denoted by $SPT(s)$, is the union of Euclidean shortest paths from s to all vertices of P. This union of paths is a planar tree, rooted at s, which has n nodes, namely the vertices of P. For every vertex x of P, let $p(s, x)$ denote the parent of x in $SPT(s)$.

A *link path* between two points s and t in P is a path inside P that connects s and t by a chain of line segments called *links*. A *minimum link path* between s and t is a link path connecting s and t that has the minimum number of links. Observe that there may be several different minimum link paths between s and t. The *link distance* between any two points of P is defined to be the number of links in a minimum link path between them.

3 Partitioning a simple polygon into weak visibility polygons

The partitioning algorithm partitions P into regions according to their link distance from v_1. The algorithm starts by computing $\mathcal{VP}(v_1)$, which is the set of all points of P whose link distance from v_1 is 1. Let us denote $\mathcal{VP}(v_1)$ as $V_{1,1}$. Then the algorithm computes the weak visibility polygons from every constructed edge of $V_{1,1}$. Let $v_{k(1)}u_{k(1)}, v_{k(2)}u_{k(2)}, \ldots, v_{k(c)}u_{k(c)}$ denote the constructed edges of $V_{1,1}$ along $bd(P)$ in clockwise order from v_1, where c is the number of constructed edges in $V_{1,1}$. Then the algorithm removes $V_{1,1}$ from P. It can be seen that the remaining polygon $P \setminus V_{1,1}$ consists of c disjoint polygons $P(v_{k(1)}u_{k(1)}), P(v_{k(2)}u_{k(2)}), \ldots, P(v_{k(c)}u_{k(c)})$. For each $j \in \{1, 2, \ldots, c\}$, the weak visibility polygon $\mathcal{VP}(v_{k(j)}u_{k(j)})$ is computed inside the pocket $P(v_{k(j)}u_{k(j)})$, and it is denoted as $V_{2,j}$. Let $W_1 = \{V_{1,1}\}$ and $W_2 = \bigcup_{j=1}^{c}\{V_{2,j}\}$. Observe that W_2 is the set of all the disjoint regions of P, such that every point of each disjoint region in W_2 is at link distance two from v_1.

Repeating the same process, the algorithm computes $W_3, W_4, \ldots, W_d$, where d denotes the maximum link distance of any point of P from v_1. Note that it is not possible for any visibility polygon belonging to $W_d = \bigcup_{j=1}^{c} V_{d,j}$ to have any constructed edge. Therefore, no further visibility polygon is computed. Hence, $P = W_1 \cup W_2 \cup \ldots W_d = V_{1,1} \cup V_{2,1} \cup V_{2,2} \cup \ldots \cup V_{d,1} \cup V_{d,2} \cup \ldots$. Thus, the algorithm returns the set $W = \bigcup_{i=1}^{d} W_i$, which is a partition of P. We present the pseudocode for the entire partitioning algorithm below as Algorithm 3.1.

Algorithm 3.1 An algorithm for partitioning P into visibility polygons

```
1: Compute VP(v_1)
2: V_{1,1} ← VP(v_1), W_1 ← {V_{1,1}}
3: C ← ⋃_{s∈W_1}(constructed edges of s), c ← |C|
4: W ← W_1, i ← 1
5: while c > 0 do
6:     i ← i + 1, W_i ← ∅
7:     for j = 1 to c do
8:         Compute VP(v_{k(j)}u_{k(j)}), V_{i,j} ← VP(v_{k(j)}u_{k(j)})
9:         W_i ← W_i ∪ {V_{i,j}}
10:    end for
11:    W ← W ∪ W_i
12:    C ← ⋃_{s∈W_i}(constructed edges of s), c ← |C|
13: end while
14: return W
```

Figure 2 shows the outcome of running Algorithm 3.1 on a simple polygon P having 31 vertices, where the maximum link distance of any point of P from v_1 is 5. The algorithm returns the partition $W = \{V_{1,1}, V_{2,1}, V_{2,2}, V_{3,1}, V_{3,2}, V_{3,3,}, V_{4,1}, V_{4,2}, V_{5,1}, V_{5,2}, V_{5,3}\}$.

It can be seen that Algorithm 3.1, as stated above, requires $\mathcal{O}(n^2)$ time, since the visibility polygons are computed separately. However, the running time can be improved to $\mathcal{O}(n)$ by using the partitioning method given by Suri [26, 27] in the context of computing minimum link paths. Using the algorithm of Hershberger [16] for computing visibility graphs of P, Suri's algorithm computes weak visibility polygons from selected constructed edges. The same method can be used to compute weak visibility polygons from all constructed edges of visibility polygons in W in $\mathcal{O}(n)$ time. The *visibility graph* of P is a graph which has a node corresponding to every vertex of P and there is an edge between a pair of nodes if and only if the corresponding pair of vertices are visible from each other in P. We summarize the result in the following theorem.

Theorem 1. *A simple polygon P can be partitioned into visibility polygons according to their link distance from any vertex in $\mathcal{O}(n)$ time.*

4 Traversing the hierarchy of visibility polygons

Our algorithm for placement of vertex guards uses the hierarchy of visibility polygons W, as computed in Section 3. Let $S_d, S_{d-1}, \ldots, S_2, S_1$ be the set of vertex guards chosen for guarding vertices of visibility polygons in $W_d, W_{d-1}, \ldots, W_2, W_1$ respectively. Since $W_1 = \{V_{1,1}\}$ and $V_{1,1} = \mathcal{VP}(v_1)$, we have $S_1 = \{v_1\}$. So the algorithm essentially has to decide guards in $S_d, S_{d-1}, \ldots, S_2$. We have the following observation.

Lemma 2. *For every $2 \leq i < d$, every vertex guard in S_i belongs to some visibility polygon in $W_{i+1} \cup W_i \cup W_{i-1}$, and every vertex guard in S_d belongs to some visibility polygon in $W_d \cup W_{d-1}$.*

Proof. Consider any vertex guard $g \in S_i$, where $2 \leq i < d$. Now g can guard only vertices in $\mathcal{VP}(g)$, and every vertex in $\mathcal{VP}(g)$ must be at a link distance of 1 from g. Let U denote the set of vertices in $\mathcal{VP}(g)$ that also belong to any $V_{i,j} \in W_i$. The inclusion of g in S_i guarantees that U is not empty, since there exists at least one vertex $y \in U$ that is guarded by g. Now, if we consider any such $y \in U$, then the link distance between g and y must be 1, and also the link distance of y from v_1 must be i. Therefore, the link distance of g from v_1 can only be $i-1$, i, or $i+1$, and hence g must belong to some visibility polygon in $W_{i+1} \cup W_i \cup W_{i-1}$. Using the same argument, for any vertex guard $g \in S_d$, g must belong to some visibility polygon in $W_d \cup W_{d-1}$ (rather than $W_{d+1} \cup W_d \cup W_{d-1}$), since the level W_{d+1} does not exist in the hierarchy W. □

As can be seen from the proof of Lemma 2, the placement of guards is locally restricted to visibility polygons belonging to adjacent levels in the partition hierarchy W. We formalize this intuition by introducing the notion of the *partition tree* of P, which is a *dual graph* denoted by T. Each visibility polygon $V_{i,j} \in W$ is represented as a vertex of T (also denoted by $V_{i,j}$), and two vertices of T are connected by an edge in T if and only if the corresponding visibility polygons share a constructed edge. Treating $V_{1,1}$ as the root of T, the standard parent-child-sibling relationships can be imposed between the visibility polygons in W.

Our algorithm starts off by guarding all vertices belonging to the visibility polygons in $W_d = \{V_{d,1}, V_{d,2}, \ldots\}$, which are effectively the nodes of T furthest from the root $V_{1,1}$. The algorithm scans $V_{d,1}$,$V_{d,2}$,... separately for identifying the respective inward and outward guards in S_d. We know from Lemma 2 that every vertex guard in S_d belongs to some visibility polygon in $W_d \cup W_{d-1}$. Consider a particular $V_{d,k} \in W_d$, and let $V_{d-1,j} \in W_{d-1}$ be the parent of $V_{d,k}$ in T. Consider the constructed edge $v_k u_k$ between $V_{d,k}$ and $V_{d-1,j}$. For guarding the vertices of $V_{d,k} = \mathcal{VP}(v_k u_k) \setminus V_{d-1,j}$, it is enough to focus on the subpolygon Q consisting of $V_{d,k}$ itself and the portion of $V_{d-1,j}$ that is weakly visible from $v_k u_k$. So, the subproblem of guarding $V_{d,k}$ (or any other visibility polygon belonging to W_d) essentially reduces to placing vertex guards in a polygon containing a weak visibility chord vu (corresponding to $v_k u_k$ in the original subproblem) in order to guard only the vertices lying on one side of uv; however, vertex guards can be chosen freely from either side of the chord uv. We discuss the method for the placement of guards in this reduced problem in Section 5.

Instead of guarding each weak visibility polygon Q separately, common vertex guards can be placed by traversing the boundary of overlapping weak visibility polygons. Let us explain by considering any $V_{d-1,j} \in W_{d-1}$. Let us denote the constructed edges that are shared between $V_{d-1,j}$ and the m children of $V_{d-1,j}$ as $v_{j(1)}u_{j(1)}, v_{j(2)}u_{j(2)}, \ldots, v_{j(m)}u_{j(m)}$ respectively. Using all these constructed edges, let us construct the weak visibility polygons $\mathcal{VP}(v_{j(1)}u_{j(1)})$, $\mathcal{VP}(v_{j(2)}u_{j(2)})$, $\ldots$, $\mathcal{VP}(v_{j(m)}u_{j(m)})$. Observe that each such weak visibility polygon is divided into two portions by the corresponding constructed edge; one of the portions forms a child of $V_{d-1,j}$ belonging to W_d, whereas the other portion is a subregion of $V_{d-1,j}$ itself. Moreover, for several of the weak visibility polygons among $\mathcal{VP}(v_{j(1)}u_{j(1)}), \mathcal{VP}(v_{j(2)}u_{j(2)}), \ldots, \mathcal{VP}(v_{j(m)}u_{j(m)})$, the second portions may have overlapping subregions in $V_{d-1,j}$. Thus, there may exist vertex guards in these overlapping subregions that can see portions of several of the children of $V_{d-1,j}$. Therefore,

for guarding vertices of polygons from W_d, let us extend the definition of Q to be the union of all the overlapping weak visibility polygons defined by the constructed edges corresponding to the children of each $V_{d-1,j}$. For instance, consider the constructed edges $v_{17}u_{17}$, $v_{21}u_{21}$ and $v_{23}u_{23}$ on the boundary of $V_{4,1}$ in Figure 2; for guarding the corresponding children $V_{5,1}$, $V_{5,2}$ and $V_{5,3}$ respectively, we define Q as $\mathcal{VP}(v_{17}u_{17}) \cup \mathcal{VP}(v_{21}u_{21}) \cup \mathcal{VP}(v_{23}u_{23})$ and traverse Q.

After having successively computed S_d for guarding vertices belonging to visibility polygons in $W_d = \{V_{d,1}, V_{d,2}, \dots\}$, the algorithm next computes S_{d-1} for guarding vertices belonging to visibility polygons in $W_{d-1} = \{V_{d-1,1}, V_{d-1,2}, \dots\}$. Since all vertices belonging to visibility polygons in W_d are already marked by guards chosen belonging to S_d, all remaining unmarked vertices of P can have link distance at most $d-1$ from v_1. So, any weak visibility polygon $V_{d-1,k} \in W_{d-1}$ can now be treated as a weak visibility polygon that is the farthest link distance from v_1. Therefore, the guards of S_{d-1} are chosen in a similar way as those of S_d. It can be seen that this same method can be used for computing S_i for every $i < d$. Thus, in successive phases, our algorithm computes the guard sets $S_d, S_{d-1}, S_{d-2}, \dots, S_2$ for guarding vertices belonging to visibility polygons in $W_d, W_{d-1}, W_{d-2}, \dots, W_2$ respectively, until it finally terminates after placing a single guard at v_1 for guarding vertices of $V_{1,1} \in W_1$. The final guard set $S = S_d \cup S_{d-1} \cup S_{d-2} \cup \dots \cup S_2 \cup S_1$ returned by the algorithm guards all vertices of P. The pseudocode for the entire algorithmic framework is provided below.

Algorithm 4.1 Algorithm for computing a guard set S from the partition tree T rooted at v_1

```
1:  Initialize all vertices of P as unmarked
2:  d ← number of levels in the partition tree T
3:  for each i ∈ {d, d−1, ..., 3, 2} do          ▷ Traverse starting from the deepest level of T
4:      S_i ← ∅
5:      c_i ← |W_i|                              ▷ c_i denotes the number of nodes at the ith level of T
6:      for each j ∈ {1, 2, ..., c_i} do
7:          Include new guards in S_i for guarding all unmarked vertices of V_{i,j}
8:          Mark all vertices of P that are visible from the new guards added to S_i
9:      end for
10: end for
11: S_1 ← {v_1}
12: return S = S_d ∪ S_{d−1} ∪ S_{d−2} ∪ ··· ∪ S_2 ∪ S_1
```

The procedure for placing new guards in S_i for guarding a particular $V_{i,j}$, as mentioned in line 7 of Algorithm 4.1, is presented in detail in Section 5.

5 Placement of Vertex Guards in a Weak Visibility Polygon

Let Q be a simple polygon that is weakly visible from an internal chord uv, i.e. we have $\mathcal{VP}(uv) = Q$. Observe that the chord uv splits Q into two sub-polygons Q_U and Q_L as follows. The sub-polygon bounded by $bd_c(u, v)$ and uv, is denoted as Q_U, and the sub-polygon bounded by $bd_{cc}(u, v)$ and uv, is denoted as Q_L. As a first step, our algorithm (see Algorithm 5.2) places a set of vertex guards, denoted by S, for guarding *only* the vertices belonging to Q_U, though S is allowed to contain vertex guards from both Q_U and Q_L.

Let G_{opt} be a set of optimal vertex guards for guarding all points of Q_U, including interior points. Let G^U_{opt} and G^L_{opt} be the subset of guards in G_{opt} that belong to Q_U (i.e. lie on $bd_c(u, v)$) and Q_L (i.e. lie on $bd_{cc}(u, v)$) respectively. Since G^U_{opt} and G^L_{opt} form a partition of G_{opt}, $|G^U_{opt}| + |G^L_{opt}| = |G_{opt}|$. Moreover, let us denote the guards belonging to G^L_{opt} as $g^L_1, g^L_2, \dots$ such that $g^L_1 \prec g^L_2 \prec \dots$ in counter-clockwise order on $bd_{cc}(u, v)$. Similarly, let us denote the guards belonging to G^U_{opt} as $g^U_1, g^U_2, \dots$ such that $g^U_1 \prec g^U_2 \prec \dots$ in clockwise order on $bd_c(u, v)$.

5.1 Concept of Inside and Outside Guards

Suppose we wish to guard an arbitrary vertex z of Q_U. Then, a guard must be placed at a vertex of Q belonging to $\mathcal{VP}(z)$. Henceforth, let $\mathcal{VVP}(z)$ denote the set of all polygonal vertices of $\mathcal{VP}(z)$. Further, let us define the *inward visible vertices* and the *outward visible vertices* of z, denoted by $\mathcal{VVP}^+(z)$ and $\mathcal{VVP}^-(z)$ respectively, as follows.

$$\mathcal{VVP}^+(z) = \{x \in \mathcal{VVP}(z) : \text{the segment } zx \text{ does not intersect } uv\}$$
$$\mathcal{VVP}^-(z) = \{x \in \mathcal{VVP}(z) : \text{the segment } zx \text{ intersects } uv\}$$

We shall henceforth refer to the vertex guards belonging to $\mathcal{VVP}^+(z)$ and $\mathcal{VVP}^-(z)$ as *inside guards* and *outside guards* for z respectively.

Consider the weakly visible polygon in Figure 3, where the three vertices z_1, z_2 and z_3 of Q_U are such that their *outward* visible vertices are pairwise disjoint, i.e. $\mathcal{VVP}^-(z_1) \cap \mathcal{VVP}^-(z_2) = \emptyset$, $\mathcal{VVP}^-(z_2) \cap \mathcal{VVP}^-(z_3) = \emptyset$, and $\mathcal{VVP}^-(z_1) \cap \mathcal{VVP}^-(z_3) = \emptyset$. Therefore, if our algorithm chooses only outside guards, then three separate guards are required for guarding z_1, z_2 and z_3. However, an optimal solution may place a single guard on any one of the five vertices of Q_U for guarding z_1, z_2 and z_3.

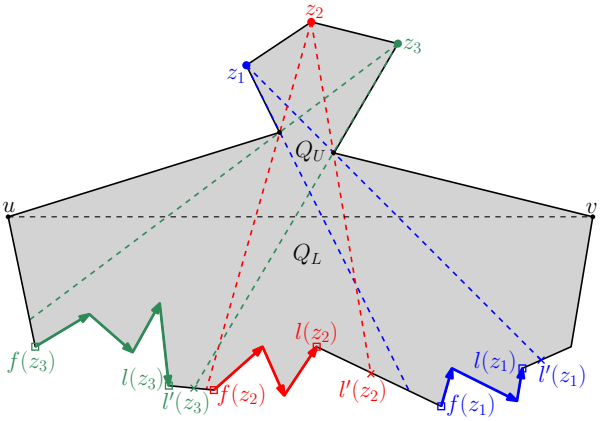

Figure 3: Example showing the need for placement of inside guards. Note that $\mathcal{VVP}^-(z_1) \cap \mathcal{VVP}^-(z_2) = \emptyset$, $\mathcal{VVP}^-(z_2) \cap \mathcal{VVP}^-(z_3) = \emptyset$, and $\mathcal{VVP}^-(z_1) \cap \mathcal{VVP}^-(z_3) = \emptyset$.

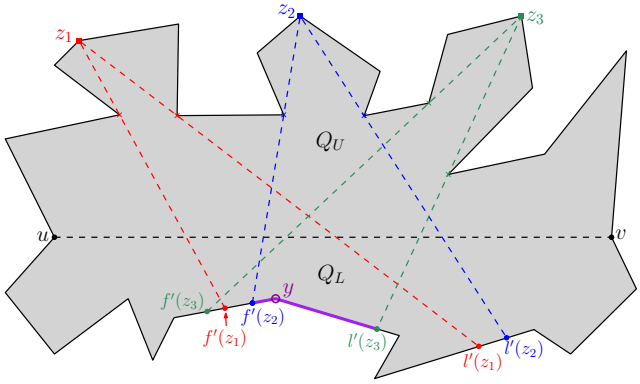

Figure 4: Example showing the need for placement of outside guards. Note that $\mathcal{VVP}^+(z_1) \cap \mathcal{VVP}^+(z_2) = \emptyset$, $\mathcal{VVP}^+(z_2) \cap \mathcal{VVP}^+(z_3) = \emptyset$, and $\mathcal{VVP}^+(z_1) \cap \mathcal{VVP}^+(z_3) = \emptyset$.

On the other hand, in Figure 4, the three vertices z_1, z_2 and z_3 of Q_U are such that their *inward* visible vertices are pairwise disjoint, i.e. $\mathcal{VVP}^+(z_1) \cap \mathcal{VVP}^+(z_2) = \emptyset$, $\mathcal{VVP}^+(z_2) \cap \mathcal{VVP}^+(z_3) = \emptyset$, and $\mathcal{VVP}^+(z_1) \cap \mathcal{VVP}^+(z_3) = \emptyset$. Therefore, if our algorithm chooses only inside guards, then three separate guards are required for guarding z_1, z_2 and z_3. However, the outward visible vertices of z_1, z_2 and z_3 overlap, and moreover there exists a vertex Y of Q_L such that $G \in \mathcal{VVP}^-(z_1) \cap \mathcal{VVP}^-(z_2) \cap \mathcal{VVP}^-(z_3)$. Thus, an optimal solution may choose y as an outside guard for guarding z_1, z_2 and z_3 together.

The above discussion suggests that it is better to choose guards from both $\mathcal{VVP}^+(z)$ and $\mathcal{VVP}^-(z)$ for guarding the same vertex z of Q_U, in order to prevent computing a guard S that is arbitrarily large compared to G_{opt}. Therefore, a special subset of vertices of Q_U is selected by our algorithm, such that both inside and outside guards are placed with the explicit intent of guarding these vertices only, and yet all the remaining vertices of Q_U become guarded as an indirect consequence. We henceforth refer to this subset of special vertices as *primary vertices*, and denote it by Z. Further, let $Z^U \subseteq Z$ be such that each primary vertex in Z^U is visible from at least one guard in G^U_{opt}. Similarly, let $Z^L \subseteq Z$ be such that each primary vertex in Z^L is visible from at least one guard in G^L_{opt}. Since any $z \in Z$ must be visible from at least

one guard of G^U_{opt} or G^L_{opt}, we have $Z = Z^U \cup Z^L$, and so we have:

$$|Z| \leq |Z^U| + |Z^L| \tag{1}$$

The general strategy for placement of guards by our algorithm for guarding only the vertices of Q_U aims to establish a constant-factor approximation bound on S by separately proving the following three bounds.

$$|S| \leq c \cdot |Z| \tag{2}$$

$$|Z^L| \leq c_1 \cdot |G^L_{opt}| \tag{3}$$

$$|Z^U| \leq c_2 \cdot |G^U_{opt}| \tag{4}$$

The above inequalities (1), (2), (3) and (4) together imply the following conclusion.

$$|Z| \leq |Z^L| + |Z^U| \leq c_1 \cdot |G^L_{opt}| + c_2 \cdot |G^U_{opt}| \leq \max(c_1, c_2) \cdot |G_{opt}| \tag{5}$$

5.2 Selection of Primary Vertices

Observe that, for any vertex $z_k \in Z$, both $\mathcal{VVP}^+(z_k)$ and $\mathcal{VVP}^-(z_k)$ may be considered to be ordered sets by taking into account the natural ordering of the visible vertices of Q in clockwise order along $bd_c(u, v)$ and in counter-clockwise order along $bd_{cc}(u, v)$ respectively. Let us denote the *first visible vertex* and the *last visible vertex* belonging to the ordered set $\mathcal{VVP}^-(z_k)$ by $f(z_k)$ and $l(z_k)$ respectively (see Figure 3). Also, we denote by $l'(z_k)$ (similarly, $f'(z_k)$) the *last visible point* (similarly, *first visible point*) of $\mathcal{VVP}^-(z_k)$, which is obtained by extending the ray $\overrightarrow{z_k p(v, z_k)}$ (respectively, $\overrightarrow{z_k p(u, z_k)}$) till it touches $bd_{cc}(u, v)$. Observe that $bd_{cc}(f'(z_k), l'(z_k)) \cap \mathcal{VP}(z_k)$ is the portion of the boundary $bd_{cc}(u, v)$ that is visible from z_k. Note that all vertices of $bd_{cc}(f'(z_k), l'(z_k))$ may not be visible from z_k, since some of them may lie inside left or right pockets of $\mathcal{VP}(z_k)$. Similarly, observe that $bd_c(p(u, z_k), p(v, z_k))$ is the portion of the boundary $bd_c(u, v)$ that is visible from z_k. Note that all vertices of $bd_{cc}(p(u, z_k), p(v, z_k))$ may not be visible from z_k, since some of them may lie inside left or right pockets of $\mathcal{VP}(z_k)$.

Let us discuss how primary vertices are selected by our algorithm. Initially, since all vertices of Q_U are unguarded, they are considered to be *unmarked*. As vertex guards are placed over successive iterations, vertices of Q_U are *marked* as soon as they become visible from some guard placed so far. At the k-th iteration, the next primary vertex $z_k \in Z$ chosen by our algorithm is an unmarked vertex such that $l'(z_k)$ precedes $l'(x)$ (henceforth denoted notationally as $l'(z_k) \prec l'(x)$) for any other unmarked vertex x of Q_U. An immediate consequence of such a choice for the primary vertex z_k is that all vertices on $bd_c(z_k, p(v, z_k))$ must already be marked.

Lemma 3. *If z_k is chosen as the next primary vertex by Algorithm 5.1, then no unmarked vertices exist on $bd_c(z_k, p(v, z_k))$.*

Proof. Let us assume, on the contrary, that there exists a vertex q on $bd_c(z_k, p(v, z_k))$ that is yet to be marked. Observe that the ray $\overrightarrow{qp(v, q)}$ must intersect the ray $\overrightarrow{z_k p(v, z_k)}$ at a point between z_k and $p(v, z_k)$, which implies that $l'(q) \prec l'(z_k)$ on $bd_{cc}(u, v)$. So, in the current iteration, q rather than z_k is chosen as the next primary vertex, which is a contradiction. □

In Section 5.3, we consider guarding vertices of Q_U in a special scenario where G_{opt} uses vertex guards only from Q_L. In Section 5.4, we consider guarding vertices of Q_U in the general scenario where the guards of G_{opt} are not restricted to Q_L. In Section 5.5, we enhance the procedure for guarding vertices to ensure that all interior points of Q_U are guarded as well.

5.3 Placement of guards in a special scenario

Let us consider a special situation where $G^L_{opt} = G_{opt}$ and $G^U_{opt} = \emptyset$, i.e. G_{opt} uses only vertex guards from Q_L. In such a scenario, for every vertex $z_k \in Z$, $\mathcal{VVP}^-(z_k)$ must contain a guard from G_{opt}. So, a natural idea is to place outside guards in a greedy manner so that they lie in the common intersection of outward visible vertices of as many vertices of Q_U as possible. For any primary vertex z_k, let us denote by $\mathcal{OVV}^-(z_k)$ the set of unmarked vertices of Q_U whose outward visible vertices overlap with those of z_k. In other words,

$$\mathcal{OVV}^-(z_k) = \{x \in \mathcal{V}(Q_U) : \ x \text{ is unmarked, and } \mathcal{VVP}^-(z_k) \cap \mathcal{VVP}^-(x) \neq \emptyset\}$$

So, each vertex of Q_U belonging to $\mathcal{OVV}^-(z_k)$ is visible from at least one vertex of $\mathcal{VVP}^-(z_k)$. Further, $\mathcal{OVV}^-(z_k)$ can be considered to be an ordered set, where for any pair of elements $x_1, x_2 \in \mathcal{OVV}^-(z_k)$, we define $x_1 \prec x_2$ if and only if $l'(x_1)$ precedes $l'(x_2)$ in counter-clockwise order on $bd_{cc}(u, v)$. For the current primary vertex z_k, let us assume without loss of generality that $\mathcal{OVV}^-(z_k) = \{x^k_1, x^k_2, x^k_3, \dots x^k_{m(k)}\}$ such that $l'(x^k_1) \prec l'(x^k_2) \prec \cdots \prec l'(x^k_{m(k)})$ in counter-clockwise order on $bd_{cc}(u, v)$.

Let us partition the vertices belonging to $\mathcal{OVV}^-(z_k)$ into 3 sets, viz. A^k, B^k and C^k in the following manner. Consider any vertex $x^k_i \in \mathcal{OVV}^-(z_k)$, such that $\mathcal{VVP}^-(x^k_i)$ creates a constructed edge $t(x^k_i)t'(x^k_i)$, where $t(x^k_i) \in \mathcal{V}(Q_L)$ is a polygonal vertex and $t'(x^k_i)$ is the point where $\overrightarrow{x^k_i t(x^k_i)}$ first intersects $bd_{cc}(u, v)$. If $t(x^k_i)$ lies on $bd_{cc}(f'(z_k), l'(z_k))$ and $t'(x^k_i)$ lies on $bd_{cc}(l'(z_k), v)$, i.e. if $f'(z_k) \prec t(x^k_i) \prec l'(z_k)$ and $l'(z_k) \prec t'(x^k_i) \prec v$, then x^k_i is included in A^k. For example, in Figure 6, $x^k_1 \in A^k$ due to the constructed edge $t(x^k_1)t'(x^k_1)$. If $t(x^k_i)$ lies on $bd_{cc}(l'(z_k), v)$ and $t'(x^k_i)$ lies on $bd_{cc}(f'(z_k), l'(z_k))$, i.e. if $l'(z_k) \prec t(x^k_i) \prec v$ and $f'(z_k) \prec t'(x^k_i) \prec l'(z_k)$, then x^k_i is included in C^k. For example, in Figure 6, $x^k_3 \in C^k$ due to the constructed edge $t(x^k_3)t'(x^k_3)$. Finally, we define B^k to consist of all the remaining vertices of $\mathcal{OVV}^-(z_k)$ apart from those included in A^k or C^k already, i.e. $B^k = \mathcal{OVV}^-(z_k) \setminus (A^k \cup C^k)$. Observe that, all vertices of A^k must lie on $bd_c(u, z_k)$, whereas all vertices of C^k must lie on $bd_c(z_k, v)$.

Lemma 4. *The vertex $l(z_k)$ sees all vertices belonging to B^k.*

Proof. We know that, due to the choice of the primary vertex z_k, $f'(x) \prec l(z_k) \prec l'(x)$ for every vertex $x \in \mathcal{OVV}^-(z_k)$. Therefore, if x is not visible from $l(z_k)$, then there must exist a constructed edge $t(x)t'(x)$ of $\mathcal{VVP}^-(z_k)$ such that (i) either $t(x) \in \mathcal{VVP}^-(z_k)$, in which case x must belong to A^k, or (ii) $t(x)$ lies on $bd_{cc}(l'(z_k), v)$, in which case x must belong to C^k. So, if $x \in B^k$, x must be visible from $l(z_k)$. In other words, $l(z_k)$ sees all vertices belonging to B^k. □

Corollary 5. *The shortest paths from $l(z_k)$ to any vertex of A^k makes only left turns, whereas the shortest paths from $l(z_k)$ to any vertex of C^k makes only right turns.*

Corollary 6. *If $bd_{cc}(f(z_k), l(z_k))$ is convex, then $A^k = C^k = \emptyset$ and $B^k = \mathcal{OVV}^-(z_k)$, which implies that all vertices of $\mathcal{OVV}^-(z_k)$ are visible from $l(z_k)$.*

For any $X \subseteq \mathcal{OVV}^-(z_k)$, we define $\mathcal{CI}(X) = \bigcap_{x \in X} \mathcal{VVP}^-(x)$, which implies that, for each vertex $y \in \mathcal{CI}(X)$, all the vertices belonging to X are visible from y, and hence can be guarded by placing a vertex guard at y. By definition, for any $X \subseteq \mathcal{OVV}^-(z_k)$, the vertices belonging to $\mathcal{CI}(X)$ lie on $bd_{cc}(u, v)$.

Lemma 7. *For every $k \in \{1, 2, \dots, |Z|\}$, $\mathcal{CI}(B^k \cup \{x_k\}) \neq \emptyset$.*

Proof. It follows directly from Lemma 4 that $l(z_k) \in \mathcal{VVP}^-(x)$ for every $x \in B^k$. Also, we know that $l(z_k) \in \mathcal{VVP}^-(z_k)$. Thus, $l(z_k) \in \bigcap_{x \in B^k \cup \{z_k\}} \mathcal{VVP}^-(x)$, i.e. $l(z_k) \in \mathcal{CI}(B^k \cup \{z_k\})$. □

Depending on the vertices in A^k, B^k and C^k, we have the following exhaustive cases because $\mathcal{CI}(B^k \cup \{x_k\}) \neq \emptyset$ by Lemma 4.

Case 1 - $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) \neq \emptyset$ (see Figure 6)

Case 2 - $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$ and $\mathcal{CI}(B^k \cup \{z_k\}) \neq \emptyset$

Case 2a - $\mathcal{CI}(A^k) \neq \emptyset$ and $\mathcal{CI}(C^k) \neq \emptyset$ (see Figure 7)

Case 2b - $\mathcal{CI}(A^k) = \emptyset$ and $\mathcal{CI}(C^k) \neq \emptyset$ (see Figure 8)

Case 2c - $\mathcal{CI}(A^k) \neq \emptyset$ and $\mathcal{CI}(C^k) = \emptyset$ (see Figure 9)

Case 2d - $\mathcal{CI}(A^k) = \emptyset$ and $\mathcal{CI}(C^k) = \emptyset$ (see Figure 10)

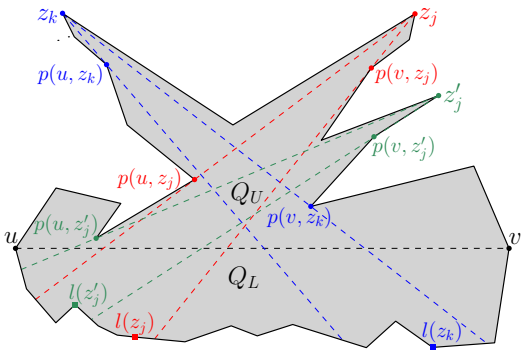

Figure 5: The choice of the next primary vertex.

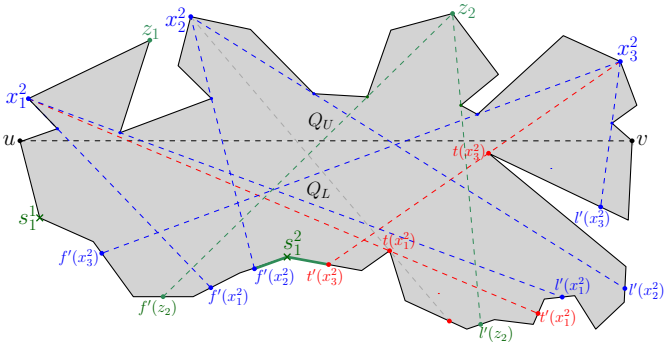

Figure 6: Figure for Case 1, where $A^2 = \{x_1^2\}$, $B^2 \supseteq \{x_2^2\}$, $C^2 = \{x_3^2\}$, and $s_1^2 \in \mathcal{CI}(A^2 \cup B^2 \cup C^2)$.

Consider Case 1, where $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) \neq \emptyset$ (see Figure 6). Here, the algorithm places a vertex guard s_1^k at any vertex belonging to $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\})$. So, every vertex in $\mathcal{OVV}^-(z_k)$ is visible from s_1^k and are hence marked after the placement of the guard at s_3^k.

Lemma 8. *If $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) \neq \emptyset$, then all vertices belonging to $\mathcal{OVV}^-(z_k)$ are visible from the vertex guard placed at s_3^k. Therefore, no vertex $x_i^k \in \mathcal{OVV}^-(z_k)$ can be a primary vertex z_j for any $j > k$.*

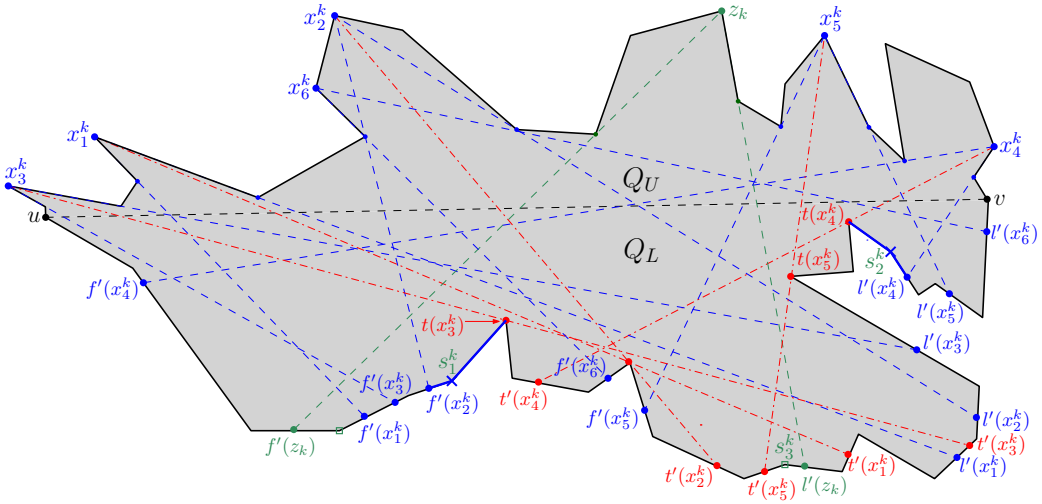

Figure 7: Figure for Case 2a, where $A^k = \{x_1^k, x_3^k\}$, $B^k = \{x_2^k, x_6^k\}$, $C^k = \{x_4^k, x_5^k\}$, $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$, $s_1^k \in \mathcal{CI}(A^k)$, $s_2^k \in \mathcal{CI}(C^k)$, and $s_3^k \in \mathcal{CI}(B^k \cup \{z_k\})$.

Now consider Case 2a, where $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$, $\mathcal{CI}(A^k) \neq \emptyset$, and $\mathcal{CI}(C^k) \neq \emptyset$ (see Figure 7). Here, the algorithm places three vertex guards, viz. $s_1^k \in \mathcal{CI}(A^k)$, $s_2^k \in \mathcal{CI}(C^k)$, and

$s_3^k \in \mathcal{CI}(B^k \cup \{z_k\})$. So, the three vertex guards placed by the algorithm together see all the vertices of $\mathcal{OVV}^-(z_k)$, and of course z_k itself. It is important to note that s_1^k or s_2^k may not belong to $\mathcal{VVP}^-(z_k)$, but s_3^k must belong to $\mathcal{VVP}^-(z_k)$.

Lemma 9. *If $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$, $\mathcal{CI}(A^k) \neq \emptyset$, and $\mathcal{CI}(C^k) \neq \emptyset$, then all vertices belonging to $\mathcal{OVV}^-(z_k)$ are visible from at least one of the three vertex guards placed at s_1^k, s_2^k, and s_3^k. Therefore, no vertex $x_i^k \in \mathcal{OVV}^-(z_k)$ can be a primary vertex z_j for any $j > k$.*

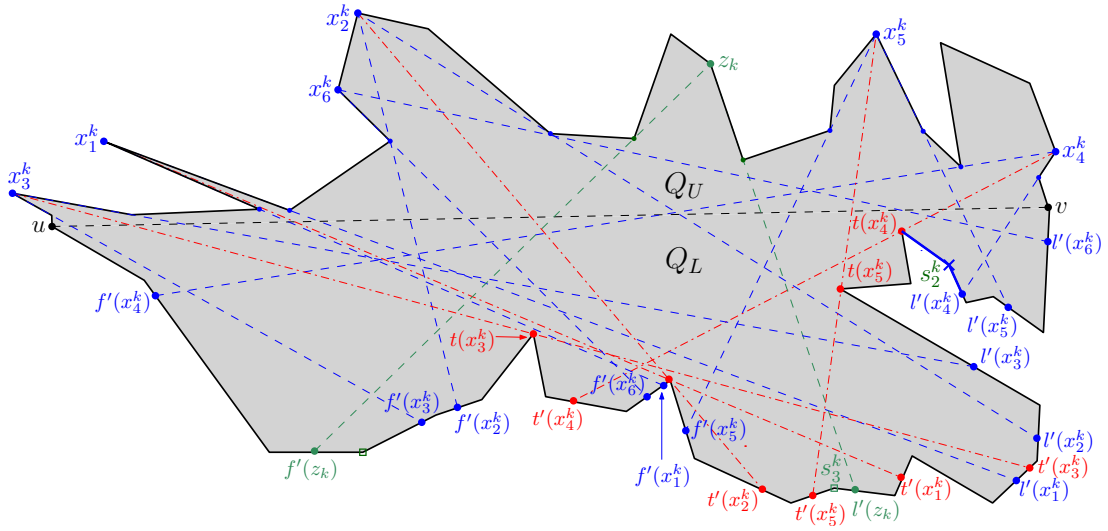

Figure 8: Figure for Case 2b, where $A^k = \{x_1^k, x_3^k\}$, $B^k = \{x_2^k, x_6^k\}$, $C^k = \{x_4^k, x_5^k\}$, $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$, $\mathcal{CI}(A^k) = \emptyset$, $s_2^k \in \mathcal{CI}(C^k)$, and $s_3^k \in \mathcal{CI}(B^k \cup \{z_k\})$.

Consider Case 2b, where $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$, $\mathcal{CI}(A^k) = \emptyset$, and $\mathcal{CI}(C^k) \neq \emptyset$ (see Figure 8). Observe that a vertex of $\mathcal{CI}(C^k)$ may not lie within $\mathcal{VVP}^-(z_k)$, but rather lie on $bd_{cc}(l'(z_k, v)$. If $\mathcal{CI}(B \cup \{z_k\} \cup C) \neq \emptyset$, then the algorithm places a single vertex guard at $s_2^k \in \mathcal{CI}(B \cup \{z_k\} \cup C)$. Otherwise, it places two vertex guards, one at a vertex $s_2^k \in \mathcal{CI}(C)$, and another one at a vertex $s_3^k \in \mathcal{CI}(B \cup \{z_k\})$. Note that s_2^k or s_3^k may see some of the vertices of A^k, which may get marked as a consequence. However, as a worst case, we assume that none of the vertices of A^k are marked due to the placement of s_2^k and s_3^k.

Also, a third vertex guard s_1^k is chosen to guard a subset of A^k, since $\mathcal{CI}(A^k) = \emptyset$. In order to choose s_1^k, $\mathcal{VVP}^-(z_k)$ is traversed counter-clockwise, starting from $f(z_k)$, till a vertex y is encountered such that there exists a vertex $x_i^k \in A^k$ which is visible from y but not from any subsequent vertex of $\mathcal{VVP}^-(z_k)$. So, this vertex $y = t(x_i^k)$ is chosen as the vertex guard s_1^k. It can be seen that once a guard is placed at $s_1^k = t(x_i^k)$, then some of the vertices in A^k are visible from x_i^k, and hence marked. Let us denote the remaining vertices of A^k that are still unmarked as A_1^k.

Lemma 10. *If $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$, $\mathcal{CI}(A^k) = \emptyset$, and $\mathcal{CI}(C^k) \neq \emptyset$, then all vertices belonging to $\mathcal{OVV}^-(z_k) \setminus A_1^k$ are visible from one of the vertex guards placed at s_1^k, s_2^k, and s_3^k. Therefore, no vertex $x_i^k \in \mathcal{OVV}^-(z_k)$ can be a primary vertex z_j for any $j > k$ unless $x_i^k \in A_1^k$.*

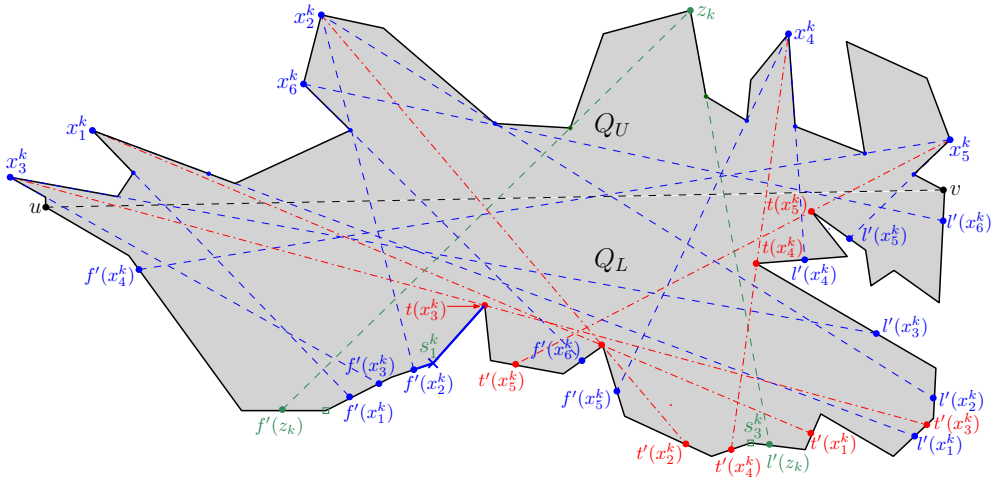

Figure 9: Figure for Case 2c, where $A^k = \{x_1^k, x_3^k\}$, $B^k = \{x_2^k, x_6^k\}$, $C^k = \{x_4^k, x_5^k\}$, $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$, $s_1^k \in \mathcal{CI}(A^k)$, $\mathcal{CI}(C^k) = \emptyset$, and $s_3^k \in \mathcal{CI}(B^k \cup \{z_k\})$.

Consider Case 2c, where $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$, $\mathcal{CI}(A^k) \neq \emptyset$, and $\mathcal{CI}(C^k) = \emptyset$ (see Figure 9). Observe that a vertex of $\mathcal{CI}(A^k)$ may not lie within $\mathcal{VVP}^-(z_k)$, but rather lie on $bd_{cc}(l'(z_k, v)$. If $\mathcal{CI}(A \cup B \cup \{z_k\}) \neq \emptyset$, then the algorithm places a single vertex guard at $s_1^k \in \mathcal{CI}(A \cup B \cup \{z_k\})$. Otherwise, it places two vertex guards, one at a vertex $s_1^k \in \mathcal{CI}(A)$, and another one at a vertex $s_3^k \in \mathcal{CI}(B \cup \{z_k\})$. Note that s_1^k or s_3^k may see some of the vertices of C^k, which may get marked as a consequence. However, as a worst case, we assume that none of the vertices of C^k are marked due to the placement of s_1^k and s_3^k.

Also, a third vertex guard s_2^k is chosen to guard a subset of C^k, since $\mathcal{CI}(C^k) = \emptyset$. In order to choose s_2^k, $\mathcal{VVP}^-(z_k)$ is traversed counter-clockwise, starting from $f(z_k)$, till a vertex y is encountered such that there exists a vertex $x_i^k \in C^k$ which is visible from y but not from any subsequent vertex of $\mathcal{VVP}^-(z_k)$. So, this vertex y, which should essentially be the vertex of $\mathcal{VVP}^-(z_k)$ immediately preceding $t'(x_i^k)$, is chosen as the vertex guard s_2^k. It can be seen that once a guard is placed at s_2^k, then a subset of the vertices in C^k are visible from x_i^k, and hence marked. Let us denote the remaining vertices of C^k that are still unmarked as C_1^k.

Lemma 11. *If $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$, $\mathcal{CI}(A^k) \neq \emptyset$, and $\mathcal{CI}(C^k) = \emptyset$, then all vertices belonging to $\mathcal{OVV}^-(z_k) \setminus C_1^k$ are visible from one of the vertex guards placed at s_1^k, s_2^k, and s_3^k. Therefore, a vertex $x_i^k \in \mathcal{OVV}^-(z_k)$ cannot be a primary vertex z_j for any $j > k$ unless $x_i^k \in C_1^k$.*

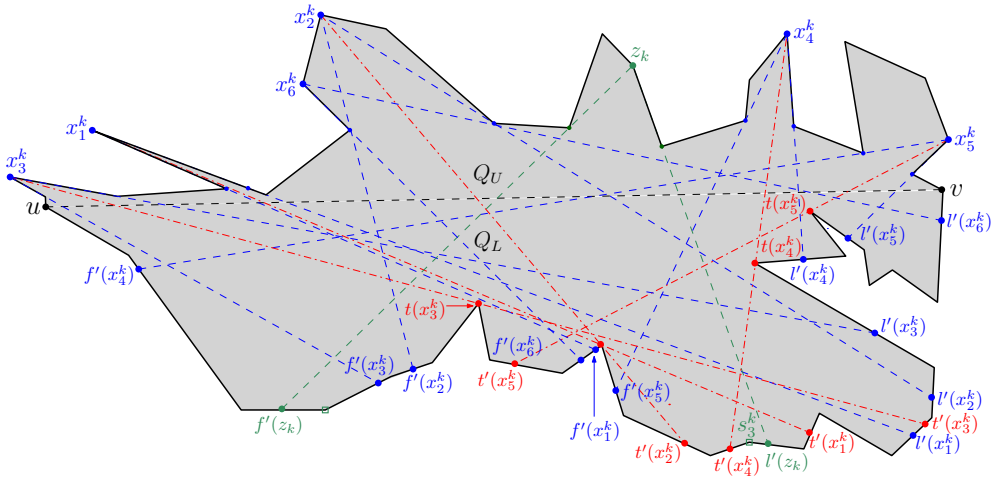

Figure 10: Figure for Case 2d, where $A^k = \{x_1^k, x_3^k\}$, $B^k = \{x_2^k, x_6^k\}$, $C^k = \{x_4^k, x_5^k\}$, $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$, $\mathcal{CI}(A^k) = \emptyset$ and $\mathcal{CI}(C^k) = \emptyset$, and $s_3^k \in \mathcal{CI}(B^k \cup \{z_k\})$.

Consider Case 2d, where $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$, $\mathcal{CI}(A^k) = \emptyset$, and $\mathcal{CI}(C^k) = \emptyset$ (see Figure 10). The algorithm first places a guard at a vertex $s_3^k \in \mathcal{CI}(B \cup \{z_k\})$. Note that s_3^k may see some of the vertices of A^k and C^k, which may get marked as a consequence. However, as a worst case, we assume that none of the vertices of A^k or C^k are marked due to the placement of s_3^k.

Following the same procedure as in Case 2c, another vertex guard s_2^k is chosen from $\mathcal{CI}(C^k)$. Similarly, following the same procedure as in Case 2b, another vertex guard s_1^k is chosen from $\mathcal{CI}(A^k)$. It can be seen that once guards are placed at s_1^k and s_2^k, then a subset of the vertices in both A^k and C^k are visible from them, and hence marked. So let us denote the remaining vertices of A^k and C^k that are still unmarked as A_1^k and C_1^k respectively.

Lemma 12. *If $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) = \emptyset$, $\mathcal{CI}(A^k) = \emptyset$, and $\mathcal{CI}(C^k) = \emptyset$, then all vertices belonging to $\mathcal{OVV}^-(z_k) \setminus (A_1^k \cup C_1^k)$ are visible from one of the three vertex guards placed at s_1^k, s_2^k, and s_3^k. Therefore, no vertex $x_i^k \in \mathcal{OVV}^-(z_k)$ can be a primary vertex z_j for any $j > k$ unless $x_i^k \in (A_1^k \cup C_1^k)$.*

Lemma 13. *Let z_k and z_j be any two primary vertices, where $j > k$. If $z_j \notin \mathcal{OVV}^-(z_k)$, then G_{opt} must require two distinct vertex guards for guarding both z_k and z_j.*

Proof. Suppose that there exists a single guard $g \in G_{opt}$ that can see both z_k and z_j. This is only possible if $g \in (\mathcal{VVP}^-(z_k) \cap \mathcal{VVP}^-(z_j))$, which in turn means that $(\mathcal{VVP}^-(z_k) \cap \mathcal{VVP}^-(z_j)) \neq \emptyset$. Therefore, $z_j \in \mathcal{OVV}^-(z_k)$ by the definition of $\mathcal{OVV}^-(z_k)$, a contradiction. □

Lemma 14. *Let z_k and z_j be any two primary vertices, where $j > k$. Assume that $A_1^k \neq \emptyset$ and $C_1^k \neq \emptyset$. If $z_j \notin (A_1^k \cup C_1^k)$, then G_{opt} must require two distinct vertex guards for guarding both z_k and z_j.*

Proof. After the placement of guards for z_k, the only vertices of $\mathcal{OVV}^-(z_k)$ that are still unmarked belong to A_1^k or C_1^k. Since z_j is unmarked when it is chosen as a primary vertex, $z_j \notin (A_1^k \cup C_1^k)$ implies that $z_j \notin \mathcal{OVV}^-(z_k)$. So, it follows directly from Lemma 13 that G_{opt} must require two distinct vertex guards for guarding both z_k and z_j. □

Lemma 15. *For every $k \in \{1, 2, \ldots, |Z|-1\}$, assume that $z_j \notin (A_1^k \cup C_1^k)$ for any j, $k < j \leq |Z|$. Then, $|S| \leq 3 \cdot |G_{opt}|$.*

Proof. We know from Lemma 14, that for any arbitrary pair z_k and z_j, where $k, j \in \{1, 2, \ldots, |Z|\}$ and $j > k$, G_{opt} requires two distinct vertex guards for guarding both z_k and z_j. Thus, applying Lemma 14 repeatedly over all such possible pairs shows that G_{opt} requires as many guards as the number of primary vertices, i.e. $|Z| \leq |G_{opt}|$. Since at most three vertex guards s_1^k, s_2^k, and s_3^k are placed corresponding to each primary vertex $z_k \in Z$, $|S| \leq 3 \cdot |Z|$. So, combining the above two inequalities, we obtain $|S| \leq 3 \cdot |Z| \leq 3 \cdot |G_{opt}|$. □

Let us focus on the case where, for some $j > k$, we have $z_j \in A_1^k$ or $z_j \in C_1^k$. Let $a_1, a_2, \ldots, a_p$ be a maximum cardinality subset $L^k \subseteq A^k$ in clockwise order on $bd_c(u, z_k)$, such that $y_1 = t(a_1) \prec f'(a_2) \prec y_2 = t(a_2) \prec \cdots \prec f'(a_p) \prec y_p = t(a_p)$ on $bd_{cc}(f(z_k), l(z_k))$ (see Figure 11). Such a set L^k is called a *maximum disjoint subset* of A^k. Note that $y_1, y_2, \ldots, y_p \in \mathcal{VVP}^-(z_k)$. Also, let $w_p, w_{p-1}, \ldots, w_1$ be the vertices immediately succeeding $t'(a_p), t'(a_{p-1}), \ldots, t'(a_1)$ respectively on $bd_{cc}(l'(z_k), v)$, such that $w_p \prec w_{p-1} \prec \cdots \prec w_1$ on $bd_{cc}(l'(z_k), v)$. Then (y_1, w_1), (y_2, w_2), $\ldots$, (y_p, w_p) can be viewed as p *nested right pockets* of A^k, introduced by vertices of A^k. We have the following lemmas relating L^k with G_{opt}.

Lemma 16. *For any primary vertex z_k, if $CI(A^k) = \emptyset$ and A^k has introduced p nested right pockets by the vertices of L^k, then any outward vertex guard placed on $bd_{cc}(u, v)$ can see at most one vertex of L^k.*

Proof. Let a' and a'' be any two vertices of L^k. We assume without loss of generality that $f(a') \prec t(a') \prec f(a'') \prec t(a'') \prec t'(a'') \prec l(a'') \prec t'(a') \prec l(a')$ (see Figure 11). This implies that $\mathcal{VVP}^-(a') \cap \mathcal{VVP}^-(a'') = \emptyset$. Thus, no single outward vertex guard placed on $bd_{cc}(u, v)$ can see more than one vertex of L^k. □

Lemma 17. *For any primary vertex z_k, if $CI(A^k) = \emptyset$ and A^k has introduced p nested right pockets by the vertices of L^k, then any outward guard set requires at least p distinct vertex guards to guard all vertices of L^k.*

Proof. Follows directly from Lemma 16. □

Similarly, let $c_1, c_2, \ldots, c_p$ be a maximum disjoint subset $R^k \subseteq C^k$ in counter-clockwise order on $bd_c(z_k, v)$, such that $y_p = t(c_p) \prec l'(c_p) \prec y_{p-1} = t(c_{p-1}) \prec l'(c_{p-1}) \prec \cdots \prec y_2 = t(c_2) \prec l'(c_2) \prec y_1 = t(c_1)$ on $bd_{cc}(l(z_k), v)$ (see Figure 12). Note that $y_1, y_2, \ldots, y_p$ lie on $bd_{cc}(l(z_k), v)$. Also, let $w_1, w_2, \ldots, w_p$ be the vertices of $\mathcal{VVP}^-(z_k)$ that immediately precede $t'(c_1), t'(c_2), \ldots, t'(c_p)$ respectively, such that $w_1 \prec w_2 \prec \cdots \prec w_{p-1} \prec w_p$ on $bd_{cc}(f'(z_k), l'(z_k))$. Then $(w_1, y_1), (w_2, y_2), \ldots, (w_p, y_p)$ can be viewed as p *nested left pockets* of C^k, introduced by vertices of C^k. We have the following lemmas relating R^k with G_{opt}.

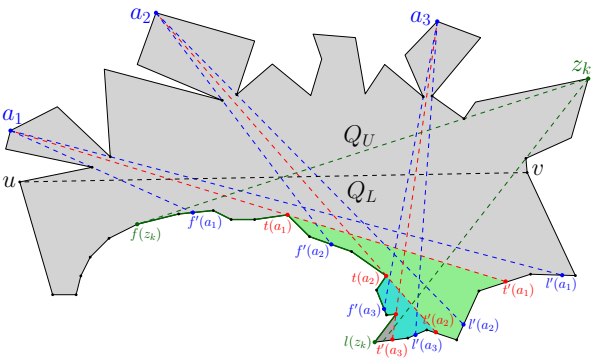

Figure 11: Figure showing a maximum disjoint subset $\{a_1, a_2, a_3\} = L^k \subseteq A^k$ such that $f'(a_1) \prec t(a_1) \prec f'(a_2) \prec t(a_2) \prec f'(a_3) \prec t(a_3) \prec t'(a_3) \prec l'(a_3) \prec t'(a_2) \prec l'(a_2) \prec t'(a_1) \prec l'(a_1)$.

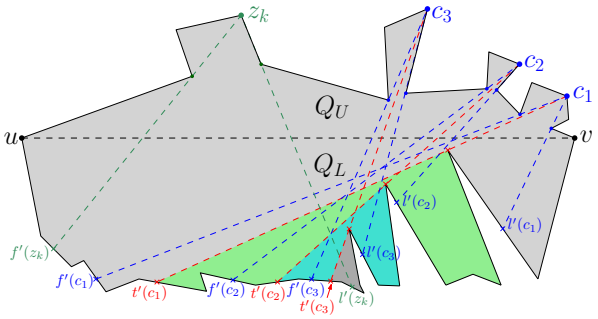

Figure 12: Figure showing a maximum disjoint subset $\{c_1, c_2, c_3\} = R^k \subseteq C^k$ such that $f'(c_1) \prec t'(c_1) \prec f'(c_2) \prec t'(c_2) \prec f'(c_3) \prec t'(c_3) \prec t(c_3) \prec l'(c_3) \prec t(c_2) \prec l'(c_2) \prec t(c_1) \prec l'(c_1)$.

Lemma 18. *For any primary vertex z_k, if $CI(C^k) = \emptyset$ and C^k has introduced q nested left pockets by the vertices of R^k, then any outward vertex guard placed on $bd_{cc}(u, v)$ can see at most one vertex of R^k.*

Proof. Let c' and c'' be any two vertices of R^k. We assume without loss of generality that $f(c') \prec t'(c') \prec f(c'') \prec t'(c'') \prec t(c'') \prec l(c'') \prec t(c') \prec l(c')$ (see Figure 12). This implies that $\mathcal{VVP}^-(c') \cap \mathcal{VVP}^-(c'') = \emptyset$. Thus, no single outward vertex guard placed on $bd_{cc}(u, v)$ can see more than one vertex of R^k. □

Lemma 19. *For any primary vertex z_k, if $CI(C^k) = \emptyset$ and C^k has introduced q nested left pockets by the vertices of R^k, then any outward guard set requires at least q distinct vertex guards to guard all vertices of R^k.*

Proof. Follows directly from Lemma 18. □

Lemma 20. *For any primary vertex z_k, let $CI(A^k) = \emptyset$. Assume that A^k has introduced p nested right pockets by the vertices of L^k. Then, the vertex guard s_1^k corresponding to z_k placed by our algorithm marks exactly one vertex of L^k.*

Proof. If $L^k = \{a_1, a_2, \ldots, a_p\}$ be such that $t(a_1) \prec t(a_2) \prec \cdots \prec t(a_p)$, then observe that the guard s_1^k is placed by Algorithm 5.1 (in line 25) on $t(a_1) \in \mathcal{VVP}^-(a_1)$, and hence $a_1 \in L^k$ is marked. However, by Lemma 16, we know that any outward vertex guard placed on $bd_{cc}(u, v)$ can see at most one vertex of L^k. Thus, s_1^k marks exactly one vertex of L^k. □

Lemma 21. *For any primary vertex z_k, let $CI(C^k) = \emptyset$. Assume that C^k has introduced q nested left pockets by the vertices of R^k. Then, the vertex guard s_2^k corresponding to z_k placed by our algorithm marks exactly one vertex of R^k.*

Proof. If $R^k = \{c_1, c_2, \ldots, c_q\}$ be such that $t'(c_1) \prec t'(c_2) \prec \cdots \prec t'(c_q)$, then observe that the guard s_2^k is placed by Algorithm 5.1 (in line 37) on a vertex $w_1 \in \mathcal{VVP}^-(c_1)$ that lies immediately before $t'(c_1)$ on $bd_{cc}(u, v)$, and hence $c_1 \in R^k$ is marked. However, by Lemma 18, we know that any outward vertex guard placed on $bd_{cc}(u, v)$ can see at most one vertex of R^k. Thus, s_2^k marks exactly one vertex of R^k. □

In the above lemmas, we have discussed the guarding of vertices in L^k and R^k. However, all vertices of A^k may not belong to L^k, and likewise, all vertices of C^k may not belong to R^k. We show that guarding the vertices of L^k (similarly, R^k), for all k, is enough for guarding the vertices of $A^k \setminus L^k$ (respectively, $C^k \setminus R^k$), without requiring any additional guards.

For a particular primary vertex $z_k \in Z$, let $L^k = \{a_1, a_2, \ldots, a_m\}$ be the maximum disjoint subset of A_k such that $t(a_1) \prec t(a_2) \prec \cdots \prec t(a_m)$ (see Figure 11). Consider any vertex $a' \in A^k \setminus L^k$ such that, for some $i \in \{2, 3, \ldots, m\}$, $f(a')$ lies between $t(a_{i-1})$ and $t(a_i)$ or $l(a')$ lies between $t'(a_i)$ and $t'(a_{i-i})$. Let $\overline{L}_i^k$ denote the set of all such vertices a' of $A^k \setminus L^k$. Observe that $A^k \setminus L^k = \bigcup_{2 \le i \le m} \overline{L}_i^k$. For any $a' \in \overline{L}_i^k \cup \{a_i\}$, where $2 \le i \le m$, $\mathcal{VVP}^-(a') \cap \mathcal{VVP}^-(a_i) \neq \emptyset$, and therefore only one vertex of $\overline{L}_i^k \cup \{a_i\}$ can belong to a maximum disjoint subset of A^k that creates nested right pockets. By Lemma 16, one vertex of $\overline{L}_i^k \cup \{a_i\}$ must be a primary vertex. We next show that two vertices of $\overline{L}_i^k \cup \{a_i\}$ cannot be chosen as primary vertices.

Lemma 22. *Let $a_i \in L^k \subseteq A^k$. If $a' \in \overline{L}_i^k \cup \{a_i\}$ is chosen as a primary vertex z_j, where $j > k$, then s_1^j and s_3^j see every other vertex $a'' \in \overline{L}_i^k \cup \{a_i\}$.*

Proof. Assume that $a \in \overline{L}_i^k \cup \{a_i\}$ is the vertex such that $t(a) \prec t(a'')$ for every other vertex $a'' \in \overline{L}_i^k \cup \{a_i\}$. Observe that, for any vertex $a'' \in \overline{L}_i^k \cup \{a_i\}$, $t(a) \in \mathcal{VVP}^-(a'')$ or $t'(a) \in \mathcal{VVP}^-(a'')$. If neither of the above is true for some $a'' \in \overline{L}_i^k \cup \{a_i\}$, then a larger disjoint set $L'_k = \{a, a''\} \cup L_k \setminus \{a_i\}$ can be constructed, which contradicts the fact that L^k is a maximum disjoint subset.

Now, if $t'(a) \prec l(a')$, then $l(a') \in \mathcal{VVP}^-(a'')$ for every $a'' \in \overline{L}_i^k \cup \{a_i\}$, and hence $s_3^j = l(z_j) = l(a')$ sees every other vertex $a'' \in \overline{L}_i^k \cup \{a_i\}$. Otherwise, $l(a') \prec t'(a)$ (see Figure 13 where $a = a_i$). For every vertex $a^* \in \overline{L}_i^k \cup \{a_i\}$ such that $t'(a^*) \prec l(a')$, a^* must be visible from $s_3^j = l(z_j) = l(a')$ as $t'(a^*) \prec l(a') \prec l(a^*)$. For every other vertex $a'' \in \overline{L}_i^k \cup \{a_i\}$, we have $t'(a') \prec l(a') \prec t'(a'')$. This implies that $f(a'') \prec t(a')$, because otherwise $t(a) \notin \mathcal{VVP}^-(a'')$ and $t'(a) \notin \mathcal{VVP}^-(a'')$, which we have shown to be impossible. Thus, a'' is seen from $s_1^j = t(a')$ placed due to $z_j = a'$. □

Lemma 23. *For every $z_k \in Z$, the guards placed by our algorithm for guarding all vertices of L^k (similarly, R^k) see all vertices of A^k (respectively, C^k), without the requirement of any additional guard.*

Proof. Follows directly from Lemma 22. □

Lemma 24. *For any primary vertex z_k, let $CI(A^k) = \emptyset$. Assume that A^k has introduced p nested right pockets by the vertices of L^k. Then, the vertex guard s_1^k corresponding to z_k placed by Algorithm 5.1 marks exactly one vertex of $L = \bigcup_{k \in \{1,2,\ldots,|Z|\}} L^k$.*

Lemma 25. *For any primary vertex z_k, let $CI(C^k) = \emptyset$. Assume that C^k has introduced q nested left pockets by the vertices of R^k. Then, the vertex guard s_2^k corresponding to z_k placed by Algorithm 5.1 marks exactly one vertex of $R = \bigcup_{k\in\{1,2,\ldots,|Z|\}} R^k$.*

Theorem 26. *Let Z be the set of primary vertices chosen by Algorithm 5.1. Then, $|S| \leq 3 \cdot |Z| \leq 6 \cdot |G_{opt}|$.*

Proof. Consider the subset $Z' \subseteq Z$ of primary vertices such that, for each $z_k \in Z'$, the vertex guards placed due to z_k marks not only z_k itself but also all vertices of $\mathcal{OVV}^-(z_k)$. Observe that, for any pair $z_k, z_j \in Z'$ where $j > k$, $z_j \notin \mathcal{OVV}^-(z_k)$, and therefore, by Lemma 13, G_{opt} requires $|Z'|$ vertex guards for guarding $\bigcup_{z_k\in Z'}(\{z_k\} \cup \mathcal{OVV}^-(z_k))$. Let us denote the set of these $|Z'|$ optimal guards as G'. Consider the remaining subset of $Z'' = Z \setminus Z'$ of primary vertices. Consider the remaining subset of $G'' = G_{opt} \setminus G'$ of optimal guards. We show that $|Z''| \leq |G''|$, so that we can finally claim that $|Z| \leq |G_{opt}|$.

Note that by the definition of Z', for any $z_j \in Z''$, $\mathcal{VVP}^-(z_j) \cap \mathcal{VVP}^-(z_k) = \emptyset$ for every $z_k \in Z'$. Consider any arbitrary primary vertex $z_k \in Z''$. By Lemma 4, we know that the vertex guard at $s_3^k = l(z_k)$ sees all vertices belonging to B^k. Note that, since the vertex guards placed due to z_k do not mark all vertices of $\mathcal{OVV}^-(z_k)$, $A_1^k \cup C_1^k \neq \emptyset$. This implies that $L^k \cup R^k \neq \emptyset$. By Lemma 23, we know that guarding only vertices of L^k and R^k are enough for guarding all vertices of A^k and C^k.

For each $z_k \in Z''$, let $L^k \subseteq A^k$ and $R^k \subseteq C^k$ denote the maximum set of vertices creating nested right pockets and nested left pockets respectively. Let $L = \bigcup_{k:z_k\in Z''} L^k$, and similarly, let $R = \bigcup_{k:z_k\in Z''} R^k$. Note that a vertex a may belong to L^k for some $z_k \in Z''$, and also belong to R^j for some $z_j \in Z''$, where $j > k$, or vice versa. In that case, if $z_l \in Z''$ denotes the last primary vertex such that $a \in \mathcal{OVV}^-(z_l)$ before a became marked, then we consider a to belong exclusively to L (respectively R) if it belongs to L^l (respectively R^l). This ensures that the sets L and R are disjoint from each other, and so we have $|L \cup R| = |L| + |R|$. By Lemmas 16 and 18, we know that any single guard $g \in G''$ can see at most one vertex of L and at most one vertex of R. Therefore, in order to guard all vertices of L and R, we must have $|G''| \geq |L|$ and also $|G''| \geq |R|$. This implies that $|G''| \geq \max(|L|, |R|) \geq (|L| + |R|)/2 = |L \cup R|/2$.

From Lemmas 24 and 25, we know that for guarding all vertices belonging to $L \cup R$, there must exist at most $|L \cup R|$ primary vertices belonging to Z''. Thus, $|Z''| \leq |L \cup R| \leq 2 \cdot |G''|$. So, we have $|Z| = |Z'| + |Z''| \leq |G'| + 2 \cdot |G''| \leq 2 \cdot |G_{opt}|$. Therefore, $|S| \leq 3 \cdot |Z| \leq 6 \cdot |G_{opt}|$. □

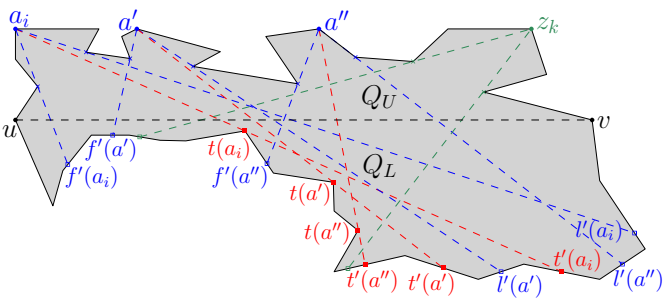

Figure 13: The vertex $a_i \in L^k$, whereas the vertices $a', a'' \notin L^k$. Since $f(a') \prec t(a_i)$, $a' \in \overline{L}_i^k$, and since $t'(a_i) \prec l(a'')$, $a'' \in \overline{L}_i^k$.

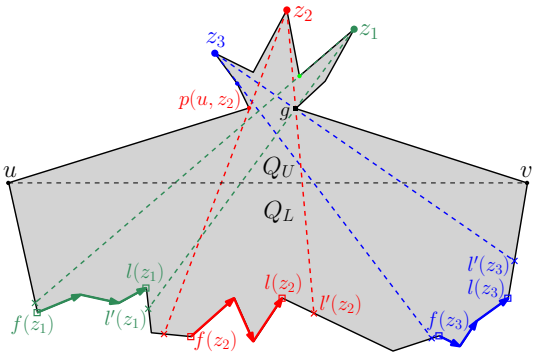

Figure 14: A guard g on $bd_c(u, v)$ sees z_1, z_2 and z_3, but three guards are necessary on $bd_{cc}(f(z_1), l(z_1))$, $bd_{cc}(f(z_2), l(z_2))$ and $bd_{cc}(f(z_3), l(z_3))$ respectively to see them.

Algorithm 5.1 An $\mathcal{O}(n^4)$-algorithm for computing a guard set S for all vertices of Q_U

Initialize $k \leftarrow 0$ and $S \leftarrow \emptyset$
Initialize all vertices of Q_U as unmarked
while there exists an unmarked vertex in Q_U **do**
    Set $k \leftarrow k + 1$ ▷ Variable k keeps count of the current iteration
    $z_k \leftarrow$ the first unmarked vertex of $bd_c(u, v)$ in clockwise order
    $q \leftarrow z_k$
    while $q \neq v$ **do**
        $q \leftarrow$ vertex next to q in clockwise order on $bd_c(u, v)$
        if q is unmarked and $l'(q) \prec l'(z_k)$ **then**
            $z_k \leftarrow q$ ▷ Update z_k to q whenever q is unmarked and $l'(q) \prec l'(z_k)$
        end if
    end while ▷ Variable z_k is now the primary vertex for the current iteration
    Compute the ordered set $\mathcal{OVV}^-(z_k) = \{x_1^k, x_2^k, \ldots, x_{m(k)}^k\}$
    Partition $\mathcal{OVV}^-(z_k)$ into the sets A^k, B^k and C^k
    if $\mathcal{CI}(A^k \cup B^k \cup C^k \cup \{z_k\}) \neq \emptyset$ **then**
        $s_3^k \leftarrow$ any vertex belonging to $\mathcal{CI}(A^k \cup B^k \cup C^k)$
        $S^k \leftarrow \{s_3^k\}$ ▷ See Figure 8
    else
        $s_3^k \leftarrow l(z_k)$ ▷ See Figures 9, 10, 11 and 12
        if $\mathcal{CI}(A^k) \neq \emptyset$ **then**
            $s_1^k \leftarrow$ any vertex belonging to $\mathcal{CI}(A^k)$ ▷ See Figures 9 and 11
        else
            $q' \leftarrow f(z_k)$
            while $q' \neq l(z_k)$ **do**
                $q' \leftarrow$ vertex next to q' in counter-clockwise order on $bd_{cc}(u, v)$
                if $q' = t(x_i^k)$ for some $x_i^k \in A^k$ **then**
                    $s_1^k \leftarrow q'$ ▷ See Figures 10 and 12
                    break
                end if
            end while
        end if
        if $\mathcal{CI}(C^k) \neq \emptyset$ **then**
            $s_2^k \leftarrow$ any vertex belonging to $\mathcal{CI}(C^k)$ ▷ See Figures 9 and 10
        else
            $q' \leftarrow f(z_k)$
            while $q' \neq l(z_k)$ **do**
                $q' \leftarrow$ vertex next to q' in counter-clockwise order on $bd_{cc}(u, v)$
                if q' immediately precedes $t'(x_j^k)$ for some $x_j^k \in C^k$ **then**
                    $s_2^k \leftarrow q'$ ▷ See Figures 11 and 12
                    break
                end if
            end while
        end if
        $S^k \leftarrow \{s_1^k, s_2^k, s_3^k\}$
    end if
    $S \leftarrow S \cup S^k$
    Mark all vertices of Q_U visible from new guards
end while
return the guard set S

Theorem 27. *Algorithm 5.1 has a worst-case time complexity of* $\mathcal{O}(n^4)$.

5.4 Placement of guards in the general scenario

Let us consider the general scenario where $G^L_{opt} \subset G_{opt}$ and $G^U_{opt} \neq \emptyset$. If Algorithm 5.1 is executed in this scenario, there may exist a subset $Z' \subseteq Z$ of primary vertices that are visible from an optimal guard belonging to G^U_{opt} (see Figure 14). Therefore, it is necessary to choose both inside and outside vertex guards corresponding to each primary vertex $z_k \in Z$, so that it is ensured that distinct optimal guards are required for guarding every two primary vertices. So, keeping this in mind, let us modify Algorithm 5.1 so that, in addition to the three outside guards s^k_1, s^k_2 and s^k_3 it also places at most three inside guards s^k_4, s^k_5 and s^k_6 for every $z_k \in Z$.

For any primary vertex z_k, let us denote by $\mathcal{OVV}^+(z_k)$ the set of unmarked vertices of Q_U whose inward visible vertices overlap with those of z_k. In other words,

$$\mathcal{OVV}^+(z_k) = \{x \in \mathcal{V}(Q_U) : \ x \text{ is unmarked, and } \mathcal{VVP}^+(z_k) \cap \mathcal{VVP}^+(x) \neq \emptyset\}$$

Note that all vertices of $bd_c(p(u, z_k), z_k)$ may not belong to $\mathcal{OVV}^+(z_k)$. Further, every vertex of $\mathcal{OVV}^+(z_k)$ is at a link distance of 1 from some vertex of $\mathcal{VVP}^+(z_k)$ and at a link distance of 2 from z_k. In the modified algorithm, two inside guards s^k_4 and s^k_5 are placed at $p(u, z_k)$ and $p(v, z_k)$ respectively, for every primary vertex $z_k \in Z$. For the placement of an additional inside guard s^k_6, consider the following cases.

Case 1: All vertices of $\mathcal{OVV}^+(z_k)$ are visible from s^k_4 or s^k_5. So, placement of s^k_6 is not required.

Case 2: If all vertices of $\mathcal{OVV}^+(z_k)$ are not visible from s^k_4 or s^k_5, and there exists one common vertex that sees all vertices of $\mathcal{OVV}^+(z_k)$, then s^k_6 is placed on that common vertex.

Case 3: If all vertices of $\mathcal{OVV}^+(z_k)$ are not visible from s^k_4 or s^k_5, and there does not exist any common vertex that sees all vertices of $\mathcal{OVV}^+(z_k)$ (see Figure 16), then s^k_6 is placed as follows. The vertex guard s^k_6 is placed at the farthest vertex w_k of $\mathcal{VVP}^+(z_k)$ in clockwise order, starting from $p(u, z_k)$, so that it can see all vertices of $\mathcal{OVV}^+(z_k)$ that are visible from any vertex of $\mathcal{VVP}^+(z_k)$ lying on $bd_c(p(u, z_k), w_k)$ (see Figure 17).

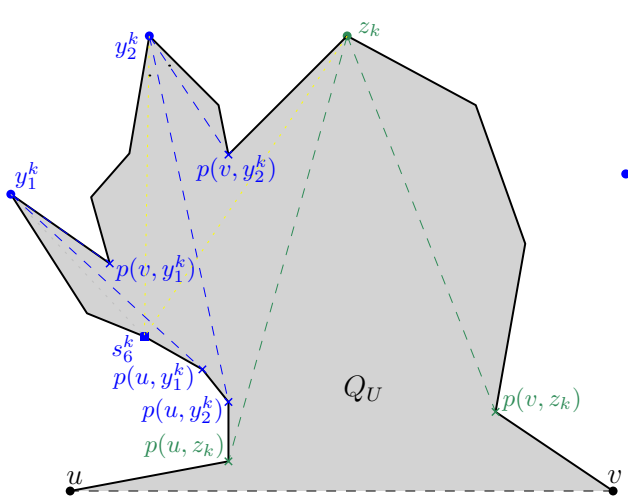

Figure 15: Vertices y^k_1 and y^k_2 can be guarded by an inside guard s^k_6, but they are not visible from guards at $s^k_4 = p(u, z_k)$ and $s^k_5 = p(v, z_k)$.

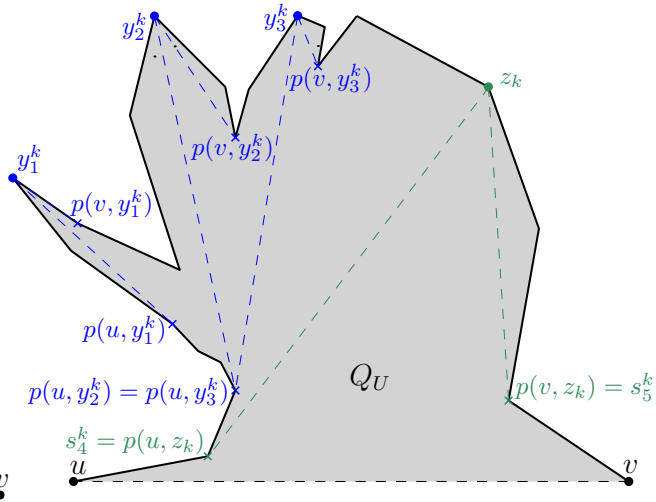

Figure 16: Vertices y^k_1 and y^k_2 cannot be guarded by inside guards at $s^k_4 = p(u, z_k)$ and $s^k_5 = p(v, z_k)$. Moreover, no single additional guard s^6_k can see both of them.

Let us discuss these cases in the presence of guards of G^U_{opt}. Assume that the current primary vertex z_k is visible from an optimal guard $g \in G^U_{opt}$. So, $z_k \in Z^U$ and $g \in \mathcal{VVP}^+(z_k)$. Thus,

if Case 1 holds true for z_k, then all visible vertices of $\mathcal{OVV}^+(z_k)$ from g are also visible from s_4^k and s_5^k. Similarly, if Case 2 holds true for z_k, then all visible vertices of $\mathcal{OVV}^+(z_k)$ from g are also visible from s_6^k, s_4^k or s_5^k. However, if Case 3 holds true for z_k, then there exists a non-empty subset of vertices $U^k \subset \mathcal{OVV}^+(z_k)$ that are visible from g but not visible from s_6^k, s_4^k or s_5^k. If $y_i^k \in U^k$ does not become a primary vertex later, then y_i^k is guarded by guards placed due to some other primary vertex. Moreover, if this happens for every $y_i^k \in U^k$, then no additional inside guard on $\mathcal{VVP}^+(z_k)$ is required.

Consider the other case where there exists two primary vertices z_j and z_m, where $m > j > k$, such that $z_j, z_m \in U^k$. Consider another primary vertex z_m, where $m > j$, such that $z_m \in \mathcal{OVV}^+(z_k)$. Since z_j and z_m are later primary vertices, we know that both z_j and z_m are not visible from s_6^k, s_4^k or s_5^k. If z_j is visible from g, then $g \in bd_c(p(u, z_j), p_v(z_j))$ and $g \in \mathcal{VVP}^+(z_j)$. Similarly, if z_m is visible from g, then $g \in bd_c(p(u, z_m), p_v(z_m))$ and $g \in \mathcal{VVP}^+(z_m)$. Now, if $bd_c(p(u, z_j), p_v(z_j))$ and $bd_c(p(u, z_m), p_v(z_m))$ are disjoint parts of $bd_c(p(u, z_k), z_k)$ (see Figure 19), then g cannot simultaneously belong to $bd_c(p(u, z_j), p_v(z_j))$ and $bd_c(p(u, z_m), p_v(z_m))$, and therefore needs another optimal g' lying on $bd_c(p(u, z_k), z_k)$ in order to guard both. Consider the special case where $p(u, z_j) = p(v, z_m)$, and so $bd_c(p(u, z_j), p_v(z_j))$ and $bd_c(p(u, z_m), p_v(z_m))$ are not totally disjoint (see Figure 18). In this case, if g has to simultaneously belong to $bd_c(p(u, z_j), p_v(z_j))$ and $bd_c(p(u, z_j), p_v(z_j))$, then the only possibility for g is to lie on $p(u, z_j) = p(v, z_m)$. But in this case, z_m cannot be a primary vertex later, since it becomes visible from $s_4^j = p(u, z_j)$, and hence marked. Finally, consider the case where $bd_c(p(u, z_m), p_v(z_m))$ is a part of $bd_c(p(u, z_j), z_j)$ (see Figure 20). Even in this case, there exists no vertex on $bd_c(p(u, z_m), p_v(z_m))$ which can see both z_m and z_j. Therefore, g cannot simultaneously see both z_j and z_m. We summarize these observations in the following lemma.

Lemma 28. *If three primary vertices z_k, z_j and z_m, where $k < j < m$, are such that $bd_c(p(u, z_j), p_v(z_j))$ and $bd_c(p(u, z_m), p_v(z_m))$ are both part of $bd_c(p(u, z_k), z_k)$, then an optimal guard $g \in G_{opt}^U$ that sees z_k cannot also see both z_j and z_m.*

Corollary 29. *Any optimal guard $g \in G_{opt}^U$ can see at most two primary vertices.*

The above corollary leads to the following theorem.

Theorem 30. *For Algorithm 5.2, $|S| \leq 6 \cdot |Z^U| \leq 12 \cdot |G_{opt}^U|$.*

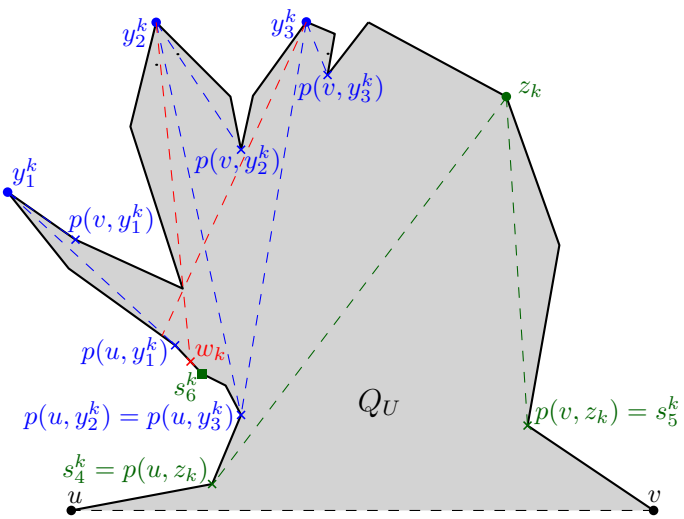

Figure 17: The guard s_6^k is placed at the farthest vertex from $p(u, z_k)$ beyond which it cannot move without losing the visibility of $y_2^k \in \mathcal{OVV}^+(z_k)$.

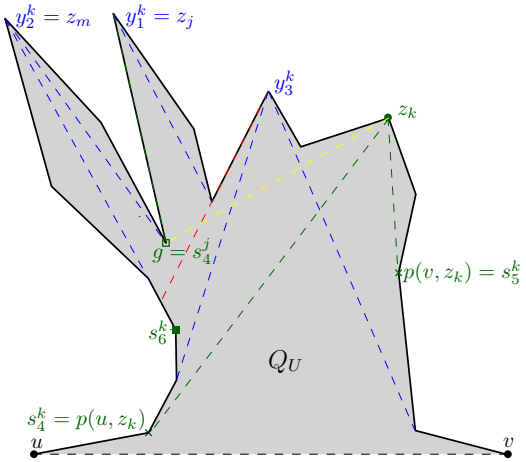

Figure 18: The two boundaries $bd_c(p(u, z_m), p(v, z_m))$ and $bd_c(p(u, z_j), p(v, z_j))$, that are both a part of $bd_c(p(u, z_k), z_k)$, share only one vertex $p(u, z_j) = p(v, z_m) = g = s_4^j$.

Algorithm 5.2 An $\mathcal{O}(n^4)$-algorithm for computing a guard set S for all vertices of Q_U

Initialize $k \leftarrow 0$ and $S \leftarrow \emptyset$
Initialize all vertices of Q_U as unmarked
Compute $SPT(u)$ and $SPT(v)$
while there exists an unmarked vertex in Q_U **do**
    Set $k \leftarrow k+1$ ▷ Variable k keeps count of the current iteration
    $z_k \leftarrow$ the first unmarked vertex of $bd_c(u,v)$ in clockwise order
    $q \leftarrow z_k$
    while $q \neq v$ **do**
        $q \leftarrow$ vertex next to q in clockwise order on $bd_c(u,v)$
        if q is unmarked and $l'(q) \prec l'(z_k)$ **then**
            $z_k \leftarrow q$ ▷ Update z_k to q whenever q is unmarked and $l'(q) \prec l'(z_k)$
        end if
    end while ▷ Variable z_k is now the primary vertex for the current iteration
    $s_4^k \leftarrow p(u,z_k)$, $s_5^k \leftarrow p(v,z_k)$, $S^k \leftarrow \{s_4^k, s_5^k\}$ ▷ See Figures 15 and 16
    Mark all vertices of Q_U visible from guards currently in S^k
    Compute the ordered set $\mathcal{OVV}^+(z_k) = \{y_1^k, y_2^k, \ldots, y_{n(k)}^k\}$
    if $\mathcal{OVV}^+(z_k) \neq \emptyset$ **then**
        if $\mathcal{CI}(\mathcal{OVV}^+(z_k)) \neq \emptyset$ **then**
            $s_6^k \leftarrow$ any vertex of $\mathcal{CI}(\mathcal{OVV}^+(z_k))$ ▷ See Figure 15
        else ▷ where $\mathcal{CI}(\mathcal{OVV}^+(z_k)) = \emptyset$ ▷ See Figure 16
            $q \leftarrow p(u,z_k)$, $VSF = \emptyset$
            while $(q \neq z_k)$ and every vertex of VSF is visible from q **do**
                $s_6^k \leftarrow q$
                $q \leftarrow$ vertex of $\mathcal{VVP}^+(z_k)$ next to q in clockwise order on $bd_c(u,v)$
                $VSF \leftarrow VSF \cup (\mathcal{VP}(s_6^k) \cap \mathcal{OVV}^+(z_k))$ ▷ See Figures 17, 19 and 20

            end while
        end if
        $S^k \leftarrow S^k \cup \{s_6^k\}$
    end if
    Mark all vertices of Q_U visible from guards currently in S^k
    Compute the ordered set $\mathcal{OVV}^-(z_k) = \{x_1^k, x_2^k, \ldots, x_{m(k)}^k\}$
    Partition $\mathcal{OVV}^-(z_k)$ into the sets A^k, B^k and C^k
    Compute s_1^k, s_2^k, and s_3^k as per Algorithm 5.1
    $S^k \leftarrow S^k \cup \{s_1^k, s_2^k, s_3^k\}$
    Mark all vertices of Q_U visible from guards currently in S^k
    $S \leftarrow S \cup S^k$
    Mark all vertices of Q_U visible from guards newly included in S
end while
return the guard set S

Theorem 31. *For Algorithm 5.2,* $|S| \leq 6 \cdot |Z| \leq 12 \cdot |G_{opt}|$.

Proof. From Theorem 26, we know that $|Z^L| \leq 2 \cdot |G_{opt}^L|$. Similarly, from Theorem 30, we know that $|Z^U| \leq 2 \cdot |G_{opt}^U|$. Further, we know that a maximum of 6 vertex guards, viz. s_1^k, s_2^k, s_3^k, s_4^k, s_5^k and s_6^k, are chosen for each primary vertex $z_k \in Z$, and so $|S| \leq 6 \cdot |Z|$. Combining all the above inequalities, we obtain: $|S| \leq 6 \cdot |Z| \leq 6 \cdot (|Z^U| + |Z^L|) \leq 12 \cdot (|G_{opt}^U| + |G_{opt}^L|) \leq 12 \cdot |G_{opt}|$ □

Theorem 32. *The running time for Algorithm 5.2 is* $\mathcal{O}(n^4)$.

Proof. While executing Algorithm 5.2 as stated, we need to precompute $SPT(u)$ and $SPT(v)$ respectively. Also, for every vertex z belonging to Q_U, we need to precompute $\mathcal{VVP}^+(z)$, $\mathcal{VVP}^-(z)$, which we can do by constructing the visibility graph of z in $\mathcal{O}(n^2)$ using the algorithm by Ghosh and Mount [15]. Note that $\mathcal{O}(n)$ vertices are chosen in total. Therefore, in order to get the overall running time for Algorithm 5.2, let us consider the running times for all the operations performed by Algorithm 5.2 in the outer while-loop corresponding to each primary vertex $z_k \in Z$.

Since $SPT(u)$ and $SPT(v)$ are precomputed, it takes only $\mathcal{O}(1)$ time to choose the guards $s_4^k = p(u, z_k)$ and $s_5^k = p(v, z_k)$. For choosing the guard s_6^k, it is required to compute $\mathcal{OVV}^+(z_k)$ and then $\mathcal{CI}(\mathcal{OVV}^+(z_k))$. The former operation takes $\mathcal{O}(n^2)$ time, since $\mathcal{VVP}^+(z)$ is precomputed. For the latter operation, it is required to compute $|\mathcal{OVV}^+(z_k)| = \mathcal{O}(n)$ intersections, and since each intersection takes $\mathcal{O}(n^2)$ time, the total time required for the operation is $\mathcal{O}(n^3)$. If $\mathcal{CI}(\mathcal{OVV}^+(z_k))$ is non-empty, then the choice of s_6^k requires only $\mathcal{O}(1)$ additional time. Otherwise, the choice of s_6^k requires a linear scan along $bd_c(u, v)$, which takes $\mathcal{O}(n)$ time. Since $\mathcal{VVP}^-(z)$ is precomputed, it takes only $\mathcal{O}(1)$ time to choose the guard $s_3^k = l(z_k)$. However, for choosing the guards s_1^k and s_2^k, it is required to compute $\mathcal{OVV}^-(z_k)$, which takes $\mathcal{O}(n)$ time, and then partition $\mathcal{OVV}^-(z_k)$ into the sets A^k, B^k and C^k, which takes a further $\mathcal{O}(n^2)$ amount of time. Finally, the the choice of s_1^k (and similarly s_2^k) requires a linear scan along $bd_{cc}(u, v)$, which takes $\mathcal{O}(n)$ time.

From the discussion above, it is clear that all the operations corresponding to a single primary vertex $z_k \in Z$ are completed by Algorithm 5.2 in $\mathcal{O}(n^3)$ time in the worst case, and since at most $\mathcal{O}(n)$ primary vertices are chosen, the overall worst case running time of Algorithm 5.2 is $\mathcal{O}(n^4)$. □

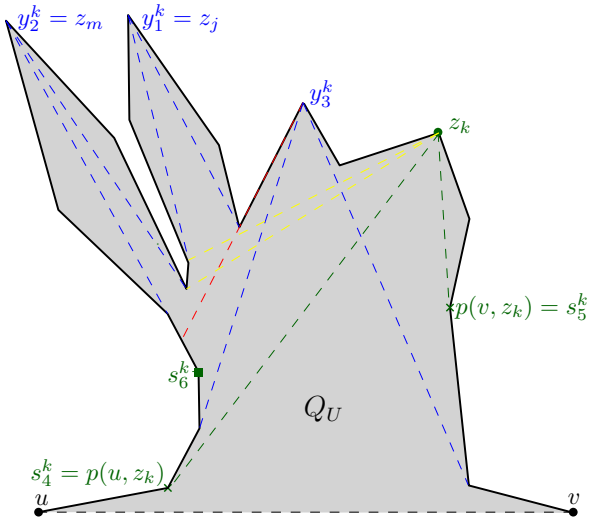

Figure 19: Boundaries $bd_c(p(u, z_m), p(v, z_m))$ & $bd_c(p(u, z_j), p(v, z_j))$ are disjoint parts of $bd_c(p(u, z_k), z_k)$.

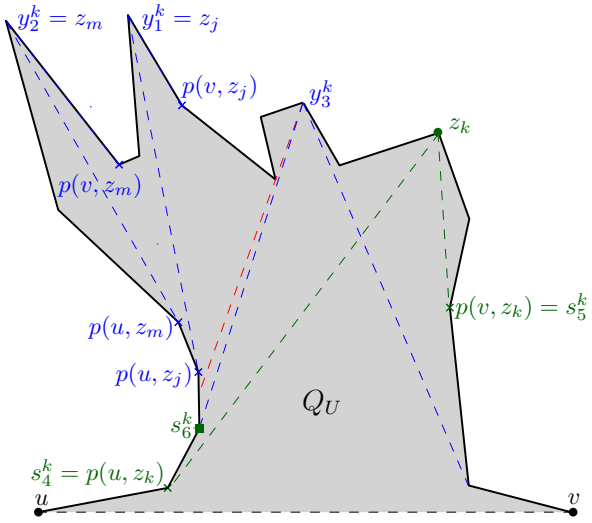

Figure 20: Boundary $bd_c(p(u, z_m), p(v, z_m))$ is contained in $bd_c(p(u, z_j), p(v, z_j))$, which in turn is contained in $bd_c(p(u, z_k), z_k)$.

5.5 Guarding all interior points of a polygon

In the previous subsection, we have presented an algorithm (see Algorithm 5.2) that returns a guard set S such that all vertices of Q_U are visible from guards in S. Recall that the art gallery problem demands that S must see all interior points of Q_U as well. However, it may not always be true that the guards in S see all interior points of Q_U. Consider the polygon shown in Figure 21. Assume that Algorithm 5.2 places guards at $p(u, z_k)$ and $p(v, z_k)$, and all

vertices of $bd_c(p(u, z_k), p(v, z_k))$ become visible from $p(u, z_k)$ or $p(v, z_k)$. However, the triangular region $Q_U \setminus (VP(p(u, z_k))) \cup VP(p(v, z_k))$, bounded by the segments x_1x_2, x_2x_3 and x_3x_1, is not visible from $p(u, z_k)$ or $p(v, z_k)$. Also, one of the sides x_1x_2 of the triangle $x_1x_2x_3$ is a part of the polygonal edge a_1a_2.

Suppose there exists another guard g lying on $bd_c(p(u, z_k), p(v, z_k))$ (see Figure 21) that sees the part of the triangle $x_1x_2x_3$ containing the side x_1x_2, but does not see the other part containing x_3. In that case, such a vertex g cannot be weakly visible from uv, which is a contradiction. Hence, for any such region invisible from guards $s_4^k, s_5^k \in S^k$ corresponding to some $z_k \in Z$, henceforth referred to as an *invisible cell*, one of the sides must always be a part of a polygonal edge. The polygonal edge which contributes as a side to the invisible cell is referred to as its corresponding *partially invisible edge*.

Observe that s_4^k and s_5^k can in fact create several invisible cells, as shown in Figure 22. Each invisible cell must be wholly contained within the intersection region (which is a triangle) of a left pocket and a right pocket. For example, in Figure 21, the invisible cell $x_1x_2x_3$ is actually the entire intersection region of the left pocket of $VP(s_4^k)$ and the right pocket of $VP(s_5^k)$. In general, where $VP(s_4^k)$ has several left pockets and $VP(s_5^k)$ has several right pockets which intersect pairwise to create multiple invisible cells (as shown in Figure 22), every such cell can be seen by placing guards on the common vertices between adjacent pairs of cells. Further, if G_{opt} is also constrained to guard these invisible cells using only inward guards from Q_U, then the number of such additional guards required can be at most twice of G_{opt}, as shown by Bhattacharya et al. [3]. However, in the absence of any constraint on placing guards, G_{opt} may place an outside guard in Q_L that sees several such invisible cells. So, it is natural to explore the possibility of being able to guard all such invisible cells by using additional guards from Q_L, in combination with guards from Q_U.

We present a modified algorithm that ensures that all partially invisible edges are guarded completely, and therefore the entire $bd_c(u, v)$ is guarded. For every pair of visible vertices in Q, extend the visible segment connecting them till they intersect the boundary of Q_U. These intersection points partition the boundary of Q_U into distinct intervals called *minimal visible intervals*. We have the following lemmas.

Lemma 33. *Every minimal visible interval on the boundary of Q_U is either entirely visible from a vertex or totally not visible from that vertex.*

Proof. Let ab be a minimal visible interval. If ab is partially visible from a vertex g, then there must exist another vertex g' such that the extension of the segment gg' intersects ab at some point, which contradicts the fact that ab is a minimal visible interval. □

Corollary 34. *If a minimal visible interval ab is entirely visible from a vertex g of Q, then the entire triangle gab lies totally inside Q.*

The modified Algorithm 5.3 first computes all minimal visible intervals and chooses one internal representative point from each minimal visible interval on the boundary of Q_U. These representative points are referred to as *pseudo-vertices*. Alongside the original polygonal vertices of Q, all pseudo-vertices are introduced on the boundary of Q_U, and the modified polygon is denoted by Q'. Note that the endpoints of minimal visible intervals are not introduced in Q'. In Algorithm 5.3, the pseudo-vertices of Q' are treated in almost the same manner as the original vertices. We compute $\mathcal{VVP}^+(z)$ and $\mathcal{VVP}^-(z)$ for all vertices of Q', irrespective of whether it is a pseudo-vertex, and some of the psuedo-vertices may even be chosen as primary vertices. However, while computing $\mathcal{VVP}^+(z)$ for any vertex z of Q', no pseudo-vertices are included in $\mathcal{VVP}^+(z)$, since they cannot be vertex guards in any case.

Algorithm 5.3 An $\mathcal{O}(n^5)$-algorithm for computing a guard set S for all vertices of Q_U

Initialize $k \leftarrow 0$ and $S \leftarrow \emptyset$
Compute $SPT(u)$ and $SPT(v)$
Compute all minimal visible intervals on boundary of Q_U
Introduce representative points from each minimal visible interval as pseudo-vertices
Initialize all vertices and pseudo-vertices of Q_U as unmarked
while there exists an unmarked vertex or pseudo-vertex on Q_U **do**
    Set $k \leftarrow k + 1$
    Choose the current primary vertex z_k as per Algorithm 5.2
    $s_4^k \leftarrow p(u, z_k)$, $s_5^k \leftarrow p(v, z_k)$, $S^k \leftarrow \{s_4^k, s_5^k\}$
    Choose the inside guard s_6^k (if required) as per Algorithm 5.2
    $S^k \leftarrow S^k \cup \{s_6^k\}$
    Compute s_1^k, s_2^k, and s_3^k as per Algorithm 5.1
    $S^k \leftarrow S^k \cup \{s_1^k, s_2^k, s_3^k\}$
    Mark all vertices of Q_U visible from guards currently in S^k
    $S \leftarrow S \cup S^k$
    Mark all vertices of Q_U visible from guards newly included in S
end while
return the guard set S

Suppose an invisible cell is created by the guards placed on the parents of some primary vertex. This implies that there exists a pseudo-vertex on the partially invisible edge contained in the invisible cell which has been left unmarked. So, either this pseudo-vertex is marked due to the guards chosen for some later primary vertex, or else it is eventually chosen as a primary vertex itself. If a pseudo-vertex on a partially invisible edge is chosen as a primary vertex z_k by Algorithm 5.3, then the entire invisible cell containing the partially invisible edge must be visible from s_4^k and s_5^k. Moreover, if the adjacent invisible cell to the left or right shares a parent, then s_4^k or s_5^k also sees both invisible cells. Therefore, if all invisible cells are guarded by guards in G_{opt}^U, then the number of pseudo-vertices that are chosen as primary vertices is at most twice of the number of guards in G_{opt}^U [3].

Observe that several invisible cells may be seen by a few outside guards belonging to G_{opt}^L, unlike G_{opt}^U which can see only two such cells at the most. Assume that a pseudo-vertex is chosen as a primary vertex z_k. Every vertex and pseudo-vertex belonging to B^k is marked due to the placement of the guard at s_3^k, and therefore no additional guards are introduced for guarding the vertices or pseudo-vertices in B^k. Consider the outside guards introduced due to A^k. Assume that A^k has pseudo-vertices. If L^k does not have pseudo-vertices, then the pseudo-vertices do not create new disjoint right pockets, and therefore guards placed to guard vertices of L^k are enough to guard pseudo-vertices in A^k. If L^k contains a pseudo-vertex, then a new disjoint right pocket is created, and therefore guards placed for other disjoint right pockets cannot see this pseudo-vertex. So, an additional outside guard is required, as well as an additional optimal guard in G_{opt}^L. The same argument holds for C^k and R^k. Thus Lemmas 20, 21, 24, 25, 28 and 29, and Theorems 26 and 30 hold even after the introduction of pseudo-vertices, and so the overall bound remains $|S| \leq 12 \cdot |G_{opt}|$, as in Theorem 31. We state this result in Theorem 35.

Theorem 35. *For the guard set S computed by Algorithm 5.3, $|S| \leq 6 \cdot |Z| \leq 12 \cdot |G_{opt}|$.*

Theorem 36. *The running time of Algorithm 5.3 is $\mathcal{O}(n^5)$.*

Proof. While executing Algorithm 5.3 as stated, the precomputation of $SPT(u)$ and $SPT(v)$ requires $\mathcal{O}(n^2)$ time as the number of pseudo-vertices is $\mathcal{O}(n^2)$. Also, for every vertex (or pseudo-vertex) z belonging to Q', the precomputation of $\mathcal{VVP}^+(z)$ and $\mathcal{VVP}^-(z)$ can be done

by constructing the visibility graph using the output-sensitive algorithm of Ghosh and Mount [15]. Since the total number of vertices (including pseudo-vertices) in Q' is $\mathcal{O}(n^2)$, and the visibility edges are not computed between pseudo-vertices, the size of the visibility graph for Q' is $\mathcal{O}(n^3)$, and thus $\mathcal{O}(n^3)$ time is required to precompute $\mathcal{VVP}^+(z)$ and $\mathcal{VVP}^-(z)$. Note that, while each of the original vertices of Q_U may be chosen as primary vertices, only at most one of the pseudo-vertices belonging to a single edge may be chosen as a primary vertex, which means that there can be at most $2n = \mathcal{O}(n)$ primary vertices chosen by Algorithm 5.3. Therefore, in order to get the overall running time for Algorithm 5.2, let us consider the running times for all the operations performed by Algorithm 5.2 in the outer while-loop (see lines ...) corresponding to each primary vertex $z_k \in Z$.

Since $SPT(u)$ and $SPT(v)$ are precomputed, it takes only $\mathcal{O}(1)$ time to choose the guards $s_4^k = p(u, z_k)$ and $s_5^k = p(v, z_k)$. For choosing the guard s_6^k, it is necessary to compute $\mathcal{OVV}^+(z_k)$ and then $\mathcal{CI}(\mathcal{OVV}^+(z_k))$. The former operation takes $\mathcal{O}(n^2)$ time, since $\mathcal{VVP}^+(z)$ is precomputed. For the latter operation, $|\mathcal{OVV}^+(z_k)| = \mathcal{O}(n^2)$ intersections are computed; since each intersection takes $\mathcal{O}(n^2)$ time, the total time required for the operation is $\mathcal{O}(n^4)$. If $\mathcal{CI}(\mathcal{OVV}^+(z_k))$ is non-empty, then the choice of s_6^k requires only $\mathcal{O}(1)$ additional time. Otherwise, the choice of s_6^k requires a linear scan along $bd_c(u, v)$, which takes $\mathcal{O}(n^2)$ time. Since $\mathcal{VVP}^-(z)$ is precomputed, it takes only $\mathcal{O}(1)$ time to choose the guard $s_3^k = l(z_k)$. However, for choosing the guards s_1^k and s_2^k, it is required to compute $\mathcal{OVV}^-(z_k)$, which takes $\mathcal{O}(n)$ time, and then the partition of $\mathcal{OVV}^-(z_k)$ into the sets A^k, B^k and C^k requires a further $\mathcal{O}(n^3)$ amount of time. Finally, the the choice of s_1^k (and similarly s_2^k) requires a linear scan along $bd_{cc}(u, v)$, which takes $\mathcal{O}(n^2)$ time.

From the discussion above, it is clear that all the operations corresponding to a single primary vertex $z_k \in Z$ are completed by Algorithm 5.2 in $\mathcal{O}(n^4)$ time in the worst case, and since at most $\mathcal{O}(n)$ primary vertices are chosen, the overall worst case running time of Algorithm 5.2 is $\mathcal{O}(n^5)$. □

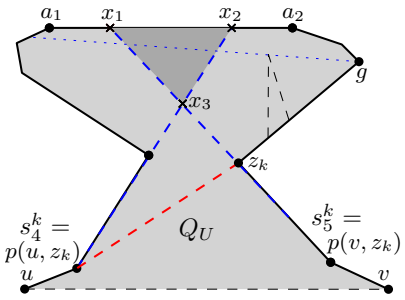

Figure 21: All vertices are visible from $p(u, z_k)$ or $p(v, z_k)$, but the triangle $x_1x_2x_3$ is invisible.

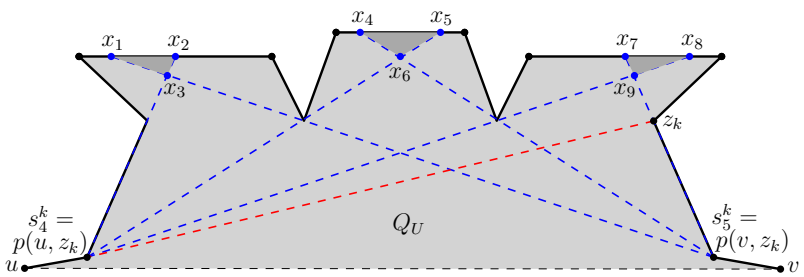

Figure 22: Multiple invisible cells exist within the polygon that are not visible from the guards placed at $p(u, z_k)$ and $p(v, z_k)$.

Algorithm 5.3 chooses a guard set S that ensures no partially invisible edge in Q_U. However, there is no guarantee that S sees the entire interior of Q_U, as there may remain residual invisible cells in the interior of Q_U (see Figure 23). Consider a residual invisible cell that is a part of an invisible cell $x_1x_2x_3$, where x_1x_2 is contained in a partially invisible edge. For such a residual invisible cell, there exists a pseudo-vertex on x_1x_2 whose parents can see the entire cell $x_1x_2x_3$, as discussed earlier in the context of placing inside guards for guarding entire visibility cell. So, placing a guard at an appropriate parent, such as z_k in Figure 23, guarantees that the residual invisible cell is totally visible. Since such an additional inside guard on Q_U corresponds to an unique outward guard in Q_L, the additional number of inside guards can be at most the number of outside guards. This amounts to placing at most (3+3)=6 inside guards and 3 outside guards corresponding to each primary vertex, while the number of primary vertices chosen remains at most $2 \cdot |G_{opt}|$. We summarize the result in the following theorem.

Theorem 37. *Let Q be a polygon of n vertices that is weakly visible from an internal chord uv. Then, a vertex guard set S can be computed in $\mathcal{O}(n^5)$ time, and $|S| \leq 18 \times |G_{opt}|$.*

6 Final Results

In Section 5, we have presented an approximation algorithm for guarding a polygon Q weakly visible from a chord uv. This algorithm chooses primary vertices z_k according to the ordering of their last visible point $l'(z_k)$. So the algorithm does not depend on a chord of the weak visibility polygon. Therefore, if this algorithm is executed on the union of overlapping weak visibility polygons Q, then it chooses primary vertices in the same way irrespective of chords in Q, producing a guard set that sees the entire union Q. So, if this algorithm is used for every overlapping weak visibility polygon in P, then the entire polygon P can be guarded by the union of the guard sets produced for guarding these overlapping weak visibility polygons. Note that there is no increase in running time for this overall algorithm, since each vertex can appear in at most 2 weak visibility polygons from the hierarchy W.

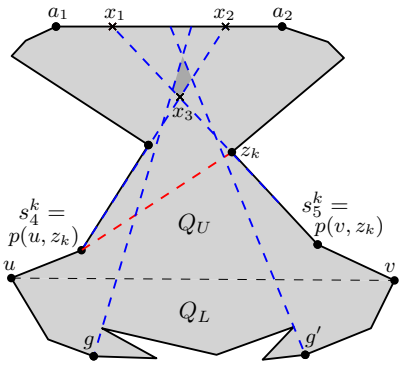

Figure 23: Two outside guards g and g' can create a residual invisible cell that is a part of the triangle $x_1x_2x_3$.

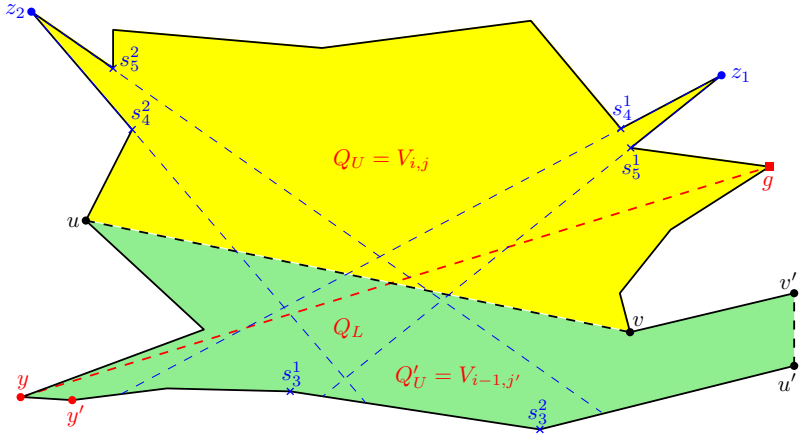

Figure 24: The vertex $y \in Q_L \cap Q'_U$ is visible from $g \in G^U_{opt}$ in Q_U, but it is not visible from any of the guards s_3^1, s_4^1, s_5^1, s_3^2, s_4^2 or s_5^2 placed in Q.

Let $g \in G_{opt}$ be an optimal guard in $V_{i,j}$. It can be seen that g is either a guard in G^U_{opt} in the weak visibility polygon Q whose chord uv is the constructed edge between $V_{i,j}$ and its parent (say $V_{i-1,j'}$), or a guard in G^L_{opt} for the overlapping weak visibility polygon Q' whose chords are constructed edges that separate $V_{i,j}$ from its children. Let $g \in Q \cap Q'$. So g is a guard in G^L_{opt} in Q' or a guard in G^U_{opt} in Q. Consider the case where g belongs to G^L_{opt} in Q'. Observe that all vertices of Q'_U that are visible from any such $g \in G^L_{opt}$ in Q' are guarded by s_1^k, s_2^k, s_3^k for all primary vertices $z_k \in Q'_U$. Therefore, the approximation bound for Algorithm 5.3 run on Q' does not change in this case.

Consider the other case where $g \in G^U_{opt}$ in Q. Observe that all vertices of Q_U that are visible from any such $g \in G^U_{opt}$ in Q are guarded by s_4^k, s_5^k, s_6^k for all primary vertices $z_k \in Q_U$. However, all vertices of Q_L that are visible from any such $g \in G^U_{opt}$ in Q may not necessarily be guarded by s_1^k, s_2^k, s_3^k,s_4^k, s_5^k, s_6^k for all primary vertices $z_k \in Q_U$ (see Figure 24), since the guards chosen by Algorithm 5.3 are not meant for guarding vertices in Q_L.

Let $y \in Q_L \cap Q'_U$ be a vertex that is visible from $g \in G^U_{opt}$ on Q, but not visible from any of the guards placed in Q by Algorithm 5.3. Since y remains unmarked, y can be chosen as a primary vertex during the execution of 5.3 on Q'. Then, the guards placed on the parents see not only y, but also see all other such vertices y' visible from g (see Figure 24) due to cross-visibility across two adjacent weak visibility polygons in the partition hierarchy W. So, this amounts to choosing

an extra primary vertex in Q, and thus the number of primary vertices visible from $g \in Q^U_{opt}$ also increases by at most 1. Since there could be at most one extra primary corresponding to every $g \in G^U_{opt}$, for counting purposes, we can attribute these to Q_U as an extra primary vertex. So, $|Z| \leq 2 \cdot |G_{opt}|$ is replaced by $|Z| \leq 3 \cdot |G_{opt}|$ in our bound for all such $Q_U = V_{i,j}$. We have the following results.

Theorem 38. *Let P be a simple polygon of n vertices. Then, a vertex guard set S for guarding all vertices of P can be computed in $\mathcal{O}(n^4)$ time, and $|S| \leq 18 \times |G_{opt}|$.*

Theorem 39. *Let P be a simple polygon of n vertices. Then, a vertex guard set S for guarding the entire boundary of P can be computed in $\mathcal{O}(n^5)$ time, and $|S| \leq 18 \times |G_{opt}|$.*

Theorem 40. *Let P be a simple polygon of n vertices. Then, a vertex guard set S for guarding interior and boundary points can be computed in $\mathcal{O}(n^5)$ time, and $|S| \leq 27 \times |G_{opt}|$.*

7 Concluding Remarks

We have presented three approximation algorithms for guarding simple polygons using vertex guards. Though the approximation ratios for our algorithms are on the higher side, they settle the long-standing conjecture by Ghosh by providing constant-factor approximation algorithms for this problem. We feel that, in practice, our algorithms will provide guard sets that are much closer in size to an optimal solution. Further, we conjecture that better approximation ratios can be proved for these 3 guarding problems. We believe that a suitable modification of our algorithm may lead to a constant-factor approximation for the version of the problem that uses perimeter point guards, which is still open.

References

[1] Alok Aggarwal. *The art gallery theorem: its variations, applications and algorithmic aspects.* PhD thesis, The Johns Hopkins University, Baltimore, Maryland, 1984.

[2] Pritam Bhattacharya, Subir Kumar Ghosh, and Bodhayan Roy. Vertex Guarding in Weak Visibility Polygons. In *Proceedings of the 1st International Conference on Algorithms and Discrete Applied Mathematics (CALDAM 2015)*, volume 8959 of *Lecture Notes in Computer Science (LNCS)*, pages 45–57. Springer, 2015.

[3] Pritam Bhattacharya, Subir Kumar Ghosh, and Bodhayan Roy. Approximability of guarding weak visibility polygons. *Discrete Applied Mathematics*, 228:109 – 129, 2017.

[4] Édouard Bonnet and Tillmann Miltzow. An Approximation Algorithm for the Art Gallery Problem. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[5] Václav Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, 1975.

[6] Ajay Deshpande, Taejung Kim, Erik D. Demaine, and Sanjay E. Sarma. A pseudopolynomial time $o(\log n)$-approximation algorithm for art gallery problems. In *WADS*, pages 163–174, 2007.

[7] Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Information Processing Letters*, 100(6):238–245, 2006.

[8] Stephan Eidenbenz, Christoph Stamm, and Peter Widmayer. Inapproximability of some art gallery problems. *Canadian Conference on Computational Geometry*, pages 1–11, 1998.

[9] Stephan Eidenbenz, Christoph Stamm, and Peter Widmayer. Inapproximability results for guarding polygons and terrains. *Algorithmica*, 31(1):79–113, 2001.

[10] Steve Fisk. A short proof of Chvátal's watchman theorem. *Journal of Combinatorial Theory, Series B*, 24(3):374, 1978.

[11] Subir Kumar Ghosh. Approximation algorithms for art gallery problems. In *Proceedings of Canadian Information Processing Society Congress*, page 429–434. Canadian Information Processing Society, 1987.

[12] Subir Kumar Ghosh. *Visibility Algorithms in the Plane*. Cambridge University Press, 2007.

[13] Subir Kumar Ghosh. Approximation algorithms for art gallery problems in polygons. *Discrete Applied Mathematics*, 158(6):718–722, 2010.

[14] Subir Kumar Ghosh. Approximation algorithms for art gallery problems in polygons and terrains. *WALCOM: Algorithms and Computation*, pages 21–34, 2010.

[15] Subir Kumar Ghosh and David M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM Journal of Computing*, 20(5):888–910, 1991.

[16] John Hershberger. An optimal visibility graph algorithm for triangulated simple polygons. *Algorithmica*, 4(1):141–155, 1989.

[17] J. Kahn, M. Klawe, and D. Kleitman. Traditional galleries require fewer watchmen. *SIAM Journal of Algebraic and Discrete Methods*, 4(2):194–206, 1983.

[18] Matthew J. Katz and Gabriel S. Roisman. On guarding the vertices of rectilinear domains. *Computational Geometry*, 39(3):219–228, 2008.

[19] James King and David G. Kirkpatrick. Improved approximation for guarding simple galleries from the perimeter. *Discrete & Computational Geometry*, 46(2):252–269, 2011.

[20] Erik A Krohn and Bengt J Nilsson. Approximate Guarding of Monotone and Rectilinear Polygons. *Algorithmica*, 66:564–594, 2013.

[21] D.T. Lee and A. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, 1986.

[22] Joseph O'Rourke. An alternative proof of the rectilinear art gallery theorem. *Journal of Geometry*, 211:118–130, 1983.

[23] Joseph O'Rourke. *Art gallery theorems and algorithms*. Oxford University Press, London, 1987.

[24] Joseph O'Rourke and Kenneth J. Supowit. Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory*, 29(2):181–189, 1983.

[25] Dietmar Schuchardt and Hans-Dietrich Hecker. Two NP-Hard Art-Gallery Problems for Ortho-Polygons. *Mathematical Logic Quarterly*, 41:261–267, 1995.

[26] Subhash Suri. A linear time algorithm with minimum link paths inside a simple polygon. *Comput. Vision Graph. Image Process.*, 35(1):99–110, July 1986.

[27] Subhash Suri. *Minimum link paths in polygons and related problems*. PhD thesis, The Johns Hopkins University, Baltimore, Maryland, 1987.