Complexity of Art Gallery Variants

Nathan Sheffield, Alek Westover

September 6, 2024

1 Introduction

The art gallery problem is a classic question in computational geometry: given a description of a polygon, compute the minimum number of guard points needed such that every point in the interior of the polygon is visible from one of guard points. This problem has been long known to be NP-hard [LL86], but it's not clear it's in NP – the guard points are allowed to have arbitrary real coordinates, so it's not obvious that an optimal placement must have a short witnessing description. In fact, a few years ago it was shown by Abrahamsen, Adamaszek, and Miltzow that the art gallery problem is complete for $\exists\mathbb{R}$ [AAM18], a class somewhere between NP and PSPACE that seems to be the natural resting place for a number of computational geometry problems. In this project, we analyze the complexity of several variants of the art gallery problem.

1.1 The Art Gallery Problem

The PointGuard Problem is defined as follows.
Input: Polygon P, number of guards g.
Output: Is there a placement of g points that can ***see*** all of P?
The polygon is represented as a list of n vertices (in the order in which they are connected in the polygon) which are pairs of B bit binary numbers in $\left\{i \cdot 2^{-B} \mid i \in [2^B]\right\}$. Thus, the input size is $\text{poly}(n, B)$. Formally, a point x in the polygon can see another point y in the polygon if the line segment joining x, y is completely contained within the polygon. An example Art Gallery is drawn in Figure 1.

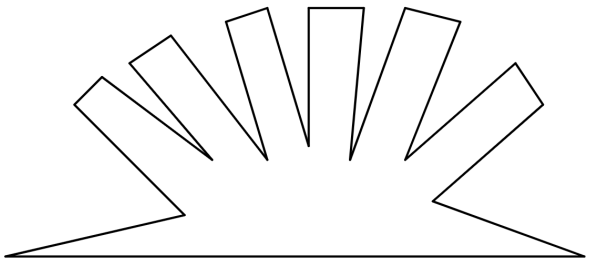

Figure 1: An Art Gallery

A classic result of Chvatal states that it is always possible to place $\lfloor n/3 \rfloor$ guards on vertices to guard any art gallery. In fact, one can compute a set of $\lfloor n/3 \rfloor$ vertex guards that guards the polygon efficiently using an algorithm based on triangulating P. However, if we want to minimize the number of guards can't just use vertices, as shown by Figure 2 and Figure 3. This is a first hint that the problem might be harder than NP. A second hint is that there are some polygons where it is necessary to place guards at irrational coordinates, e.g., $\sqrt{2}$. Abrahamsen, Adamaszek, and Miltzow prove in [AAM18] that PointGuard is $\exists\mathbb{R}$-complete, a complexity class that will be described shortly which is widely believed to strictly contain NP.

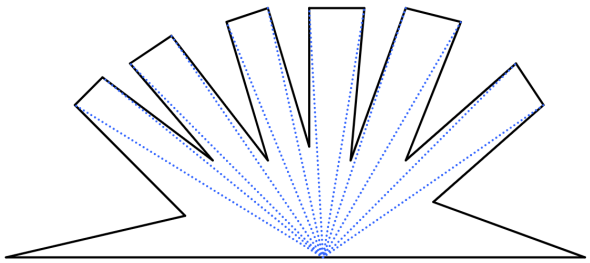

Figure 2: One guard suffices, but need $\Omega(n)$ vertex guards

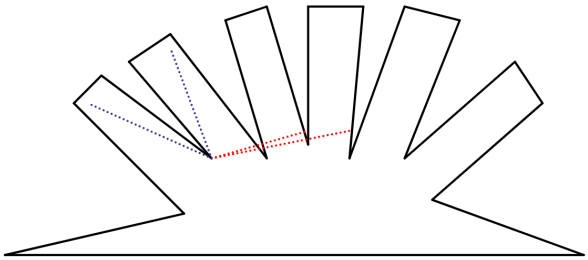

Figure 3: One guard suffices, but need $\Omega(n)$ vertex guards

1.2 Our Results on the Complexity of Art Gallery Variants

In this note we analyze the complexity of several variants of the Art Gallery Problem. Several of our variants involve guards that are not points but other geometric objects, namely line segments, circles, and squares. The guard shapes must be placed completely within P, and we say that a guard can see a point x if there is a line segment connecting point x to some point y in the guard's shape.

In Section 2 we discuss the class $\exists\mathbb{R}$, share the proof that PointGuard $\in \exists\mathbb{R}$, and give a (very) high level overview of the proof that PointGuard is $\exists\mathbb{R}$-hard.

In Section 3 we consider the LineGuard problem: "Is a polygon P guardable by g line guards?" We adapt the proof that PointGuard $\in \exists\mathbb{R}$ proof in an interesting and somewhat involved way to show:

Theorem 1.1. LineGuard $\in \exists\mathbb{R}$.

In Section 4 we use similar techniques as in Section 3, but now to show that two problems are in NP. First, we analyze PaintGuard. In PaintGuard you are given a polygon P and a set of n points called ***paintings*** in the polygon. Your goal is to determine whether it is possible to place g point guards and see all the paintings; you are not required to see the rest of the art gallery. We prove

Theorem 1.2. PaintGuard $\in$ NP.

Next, we consider a problem called PromiseGuard. In PromiseGuard, you are given a polygon P, a guard number g and a value ε. Your goal is to distinguish between two cases: (1) P can be guarded by g point guards. (2) P can't even be guarded by g radius ε circle guards. We prove

Theorem 1.3. PromiseGuard $\in$ NP.

Finally, in Section 5 we consider a 3-dimensional variant of LineGuard. We show

Theorem 1.4. 3DLineGuard is $\exists\mathbb{R}$-hard.

Remark 1.5. Although all proofs presented in sections Section 3, Section 4, Section 5 are our own, we are not extremely familiar with the literature and it is quite possible that some of these results have been proven elsewhere.

2 Preliminaries

In this section we present background on the complexity class $\exists\mathbb{R}$ and give an overview of Abrahamsen, Adamaszek, and Miltzow's proof that Art Gallery is $\exists\mathbb{R}$-complete. Having a basic high-level understanding of why Art Gallery is $\exists\mathbb{R}$-complete will be important for the rest of our paper because (1) we use similar ideas to show $\exists\mathbb{R}$ containment of line guard and shape guard, and (2) our proof that 3D line guard is hard is not a black box reduction from Art Gallery: it relies on the details of Abrahamsen, Adamaszek, and Miltzow's proof.

2.1 The Complexity Class $\exists\mathbb{R}$

The $\exists\mathbb{R}$ complexity class is a natural complexity class for many geometric problems. Informally, an $\exists\mathbb{R}$ formula is set of polynomial inequalities. More precisely, an $\exists\mathbb{R}$ formula is formed by using variables $X_1, \ldots, X_n$ along with the symbols $\vee, \wedge, \neg, 0, 1, +, -, \cdot, (,), =, <, \leq$ to form a proposition $\Phi(X_1, \ldots, X_n)$, and then taking the proposition

$$\exists X_1, X_2, \ldots, X_n \in \mathbb{R} \mid \Phi(X_1, \ldots, X_n).$$

The language $\exists\mathbb{R}$ is the set of all problems that can be reduced in polynomial time to an $\exists\mathbb{R}$ formula. Clearly $\mathsf{NP} \subseteq \exists\mathbb{R}$, and a classic but quite nontrivial result of Canny states that $\exists\mathbb{R} \subseteq \mathsf{PSPACE}$ [Can88]. A widely believed conjecture is:

Conjecture 2.1. $\mathsf{NP} \subsetneq \exists\mathbb{R}$.

Why is $\exists\mathbb{R}$ potentially larger than NP?
Intuitively, problems in NP have short certificates, but the variables we quantify over in $\exists\mathbb{R}$ can be real, and it seems hard to write down $X_1, \ldots, X_n$ efficiently a way that makes checking $\Phi(X_1, \ldots, X_n)$ easy to do efficiently. An important thing to note about $\exists\mathbb{R}$ formulas is that, while we can encode constants other than $0, 1$ by adding 1 to itself many times and then dividing, the length in bits of the numerator and denominator of a rational fraction will influence the formula length. It is also worth noting that you can force variables to take irrational values (although the values will always be algebraic). For instance, the formula: $\exists x \mid x \cdot x = 2$, forces $x = \sqrt{2}$.

A simple example of an $\exists\mathbb{R}$ formula is a formula for checking if an intersection of half-planes with rational slopes is empty. To build such a formula we first note that the determinant can be used to check if one vector is to the left or to the right of another vector. For rational numbers a_1, b_1, a_2, b_2 we define the proposition $\Phi_{L((a_1,b_1),(a_2,b_2))}(x, y)$ for checking if point x, y lies above the line that passes through $(a_1, b_1), (a_2, b_2)$ as follows:

$$(a_2 - a_1)(y - b_1) - (x - a_1)(b_2 - b_1) \geq 0.$$

Now, given several half-planes we can write a formula of the form

$$\exists x, y \mid \Phi_{L((a_1,b_1),(a_2,b_2))} \wedge \Phi_{L((a_3,b_3),(a_4,b_4))} \wedge \Phi_{L((a_5,b_5),(a_6,b_6))} \tag{1}$$

and this formula is satisfied by some $x, y \in \mathbb{R}$ if and only if the intersection of half-planes described by the formula is non-empty.

2.2 Art Gallery is in $\exists\mathbb{R}$

The following simple lemma will be relevant to several of our arguments in the remainder of the paper, so we include it here. We say that a set of regions $\mathcal{R}'$ is a ***refinement*** of a set of regions $\mathcal{R}$ if every region in $\mathcal{R}'$ is a subset of a region in $\mathcal{R}$.

Lemma 2.2. Fix point x in polygon P. Let X be the set of vertices of P unioned with $\{x\}$. Let $\mathcal{L}$ be the set of lines joining two vertices of X. Let $\mathcal{R}$ be the set of faces in the arrangement induced by $\mathcal{L}$. Let $\mathcal{R}'$ be any refinement of $\mathcal{R}$. Fix region $R \in \mathcal{R}'$. If x can see any point inside of R then x can see all of R.

Proof. Suppose for contradiction that x can see part of region R, but not all of region R. Then there must be something part of the polygon inside of region R partially obstructing x's view. But any such obstacle partially obstructing x's view must have an endpoint in R, which would have caused R to be further subdivided. □

Now we are prepared to prove that PointGuard $\in \exists\mathbb{R}$.

Theorem 2.3. Let P, g be an instance of the art gallery problem, where P is a polygon with n vertices, each of which has rational coordinates represented by at most B bits. There is a polynomial time algorithm that turns P, g into an $\exists\mathbb{R}$ formula Φ such that the $\exists\mathbb{R}$ formula has length $\text{poly}(n, B)$ and such that the Φ is satisfiable if and only if P can be guarded by g guards.

Proof. First we show how to turn an art gallery problem into a simple formula which is not quite in the $\exists\mathbb{R}$ class of formulas, but is quite close. Specifically, the problem will be that it has a $\forall$ quantifier. The formula is as follows:

$$\Psi := \exists x_1, y_1, \ldots, x_k, y_k \forall p_x, p_y, \text{INSIDE-POLYGON}(p_x, p_y) \implies \bigvee_{i=1}^{k} \text{SEES}(x_i, y_i, p_x, p_y).$$

Now we expand out the INSIDE-POLYGON and SEES sub-formulas. INSIDE-POLYGON can be implemented by triangulating the polygon and doing an OR over the triangles in the triangulation the proposition that point p_x, p_y is in each triangle. To check containment in a triangle we can use the formula described earlier (1) for checking intersection of half planes. We simply orient the edges of the triangle counter-clockwise and then check if the point is to the left of each of these edges. The INSIDE-POLYGON predicate has length $O(nB)$.

For the SEES predicate we insert we take an AND over every pair of consecutive edges, of a predicate that checks that e_1, e_2 doesn't block the line of sight between p_x, p_y and x_i, y_i; this can be done with a constant number of determinants per check.

Eliminating the $\forall$ quantifier To convert the formula Ψ into an $\exists\mathbb{R}$ formula we create a ***witness set***: a small subset of points such that if these points are guarded then it guarantees that the entire polygon is guarded. An important note is that the witness set will be a function of the guard locations. That is, we will first quantify over the guard locations, then quantify over the points in the witness set, and then check that the points in the witness set satisfy some constraints.

To construct the witness set, we will use Lemma 2.2. Specifically, we let X denote the set of corners of the polygon union with the set of guard locations. We define $\mathcal{L}$ as the set of all lines joining two points in X. We define $\mathcal{R}$ to be the set of faces of the arrangement induced by $\mathcal{L}$. And finally we define $\mathcal{C}$ to be the set of centroids of regions $R \in \mathcal{R}$. C will serve as the witness set. The key property that we have by Lemma 2.2 is that if the guards can see all points in the witness set, then they can see the entire polygon. So, instead of quantifying over all points in the polygon, we can take an AND over this finite (in fact polynomially sized) witness set.

To turn this into an $\exists\mathbb{R}$ formula, we essentially will create variables to represent the vertices of the arrangement, add constraints to check that the variables are set to vertices of the arrangement, then add variables to represent centroids of regions, and add constraints to check that these variables are set to centroids. We omit a description of the detailed mechanics of how to do this as it is not particularly illuminating. These details can be found in [AAM18].

□

2.3 Art Gallery is $\exists\mathbb{R}$ hard

The first hint that the art gallery problem might be harder than NP-hard is that there are polygons where it is necessary to place the guards at irrational points; see Figure 4.

This indicates that, if the problem is in NP, an encoding of the witness will have to be more complicated than just giving a list of guard points as rational numbers. Generally, when this is the case, a more complicated encoding that shows NP containment is unlikely to exist – in the words of Till Miltzow, "if you see something that smells like $\exists\mathbb{R}$, its almost always $\exists\mathbb{R}$". However, actually proving $\exists\mathbb{R}$ hardness can be very tricky – we will give only a rough overview of the approximately 70 page paper showing the reduction [AAM18].

The first step that Abrahamsen, Adamaszek and Miltzow take to show $\exists\mathbb{R}$-hardness is to identify a relatively simple $\exists\mathbb{R}$-complete problem to reduce from, which they call ETR-INV.

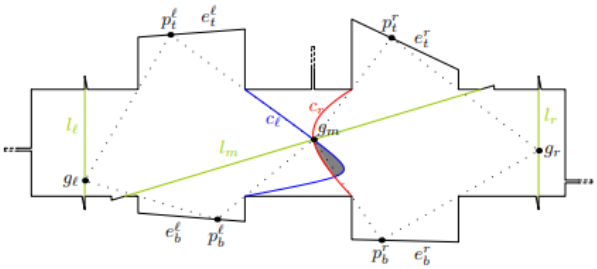

Figure 4: A figure from [AAM17] illustrating an art gallery guardable with 3 point guards, but which requires 4 point guards if guards are placed at rational coordinates. Guards are restricted by triangular holes to lie along the green segments, and in order to guard all 4 of the large quadrilateral holes, the middle guard must lie to the left of the blue quadratic curve and to the right of the red quadratic curve – these two curves intersect at an irrational point.

Definition 2.4. An instance of ETR-INV consists of variables $x_1, x_2, \ldots, x_n$, and constraints of the following forms:

- $x_i = 1$
- $x_i + x_j = x_k$
- $x_i \cdot x_j = 1$. The problem is a YES instance if and only if there exists an assignment to each of the variables such that all variables lie within $[1/2, 2]$ and all the constraints are satisfied.

Theorem 2.5. Deciding ETR-INV is $\exists\mathbb{R}$-complete.

The proof of this theorem is some somewhat involved algebra. This problem is a useful starting point for showing $\exists\mathbb{R}$-hardness because it requires fewer types of constraints to be implemented than the general existential theory of the reals problem (e.g. instead of having to encode arbitrary multiplications, the reduction only needs to provide a gadget for inverting a variable, and instead of encoding arbitrary inequality constraints, we just need equalities and the fact that all variables lie within $[1/2, 2]$).

Abrahamsen, Adamaszek and Miltzow show hardness of PointGuard by reduction from ETR-INV. Their construction involves a large region with a row of variable gadgets at the bottom, and clause gadgets on the sides. The value of a variable is encoded by where a guard is placed along a specific segment; clause gadgets copy these guard placements onto new segments, and enforce particular relationships between them.

A variable gadget is shown in Figure 5.

The clause gadgets are arranged along the walls. They consist of small passageways into additional tucked-away rooms, as shown in Figure 6. Those rooms are designed such that the number of guards required to guard them increases by at least 1 if some constraint on the variable gadgets fails.

The way a clause gadget functions is by copying values from 2 or 3 different variable gadgets onto new guard segments within the gadget. This copying allows the gadget to check that the constraint is satisfied without interfering with the rest of the construction; an illustration of the idea is shown in Figure 7.

The copy gadgets work by using ***nooks*** and ***umbras***, as shown in Figure 8 and Figure 9.

After constructing copy gadgets, what remains for the reduction is to construct clause gadgets enforcing constraints of the form $x_i + x_j = x_k$ and $x_i \cdot x_j = 1$. The constructions of these gadgets are rather complex, with several components, and values found by computer search – we will not present them here. The only aspect of any of this that is actually relevant to our result in this paper is that, for the entire construction, every guard must be placed either:

- on a specific line segment, or
- at a specific point,

and that each of these guard points and guard segments requires exactly 1 guard.

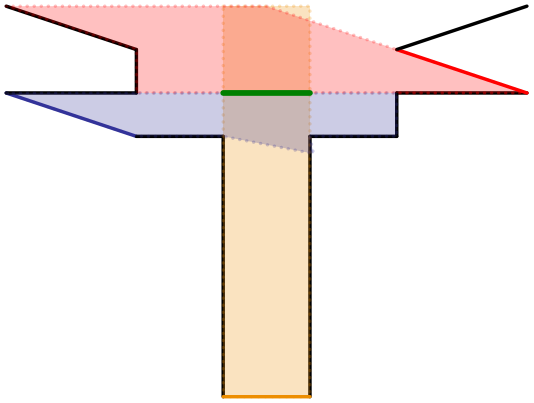

Figure 5: In order for a single guard to be able to see all of the blue edge, the red edge, and the yellow edge of the polygon, it must lie within all 3 of the corresponding highlighted regions. The intersection of these 3 regions is a single line segment, shown in green – this gadget is satisfied by a single guard if and only if that guard is placed somewhere along the green segment. The placement of the guard along the segment encodes the value of the corresponding variable.

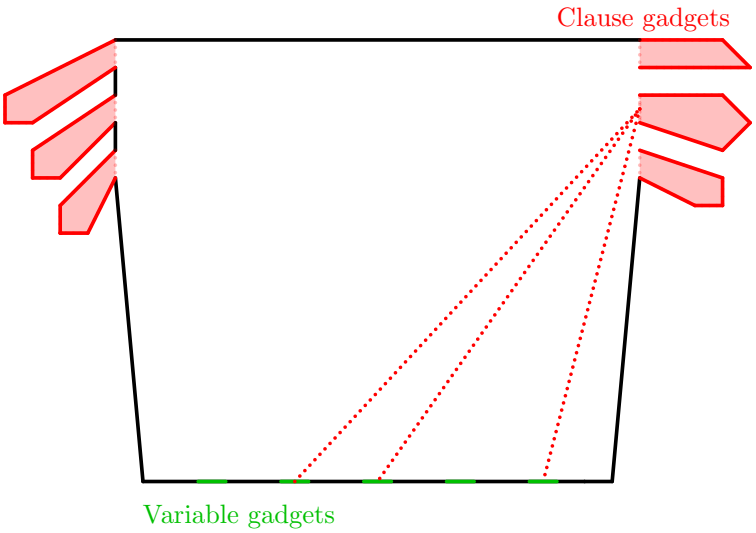

Figure 6: Each clause gadget is structured internally so that it relies on visibility from at most 3 of the variable gadgets.

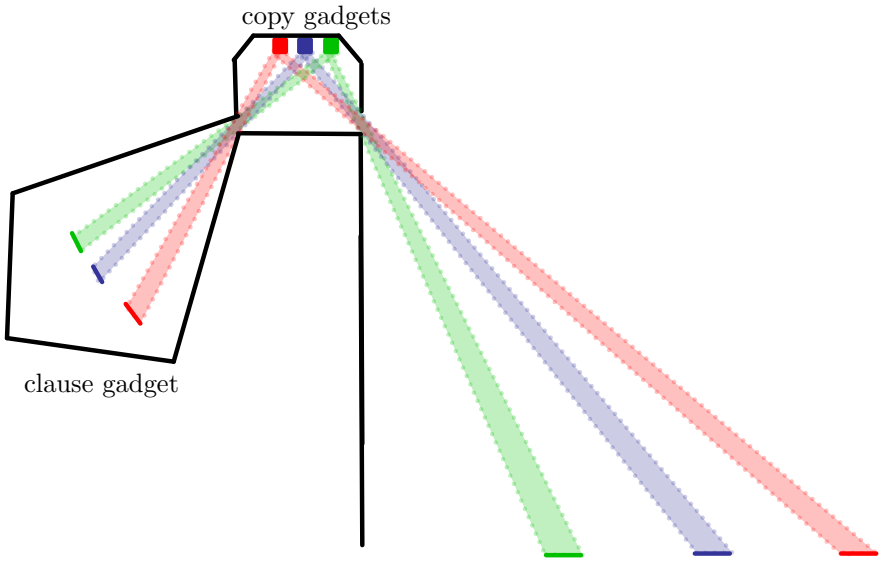

Figure 7: By making use of carefully placed copy gadgets, we can build a clause gadget whose satisfiability depends only on the placement of guards in a couple specific variable gadgets.

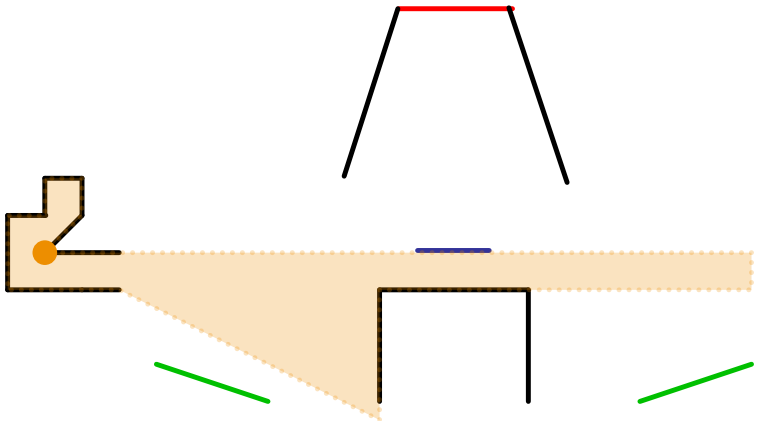

Figure 8: The green segments represent guard segments (one corresponding to a variable gadget, the other to the segment we're copying it to in a clause gadget), and the orange point represents a guard (note that the room we've constructed around it forces a guard to be placed at that exact point). Here, we call the red segment a ***nook***, and the blue segment an ***umbra***. Note that, for both guard segments, the closer to the middle the guard is placed the more of the red nook segment it can see, but the less of the blue umbra segment it can see. The idea is that, in order to be able to see all of both the nook and the umbra between the two of them, the guard on the second segment's location is determined exactly by the guard on the first segment. The purpose of the yellow guard in this construction is to be able to see everything below the umbra segment, but nothing above it, so that the whole construction is guarded iff the guard segments can see both the nook and umbra.

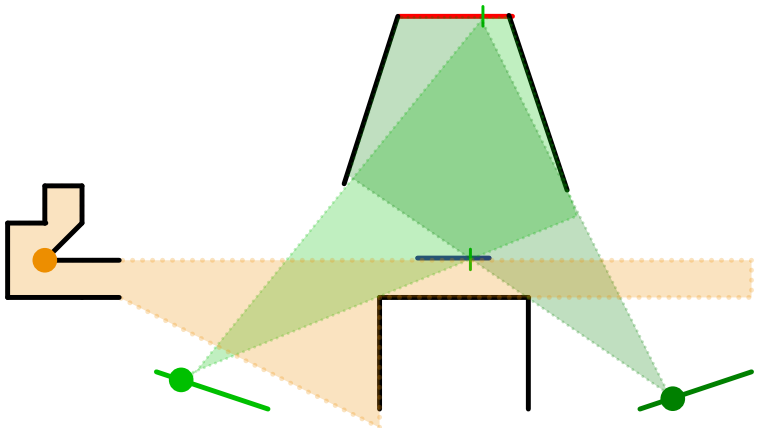

Figure 9: By constructing this gadget precisely, we ensure that wherever the first guard is placed, in order to see the remaining portions of the nook and umbra, the second guard must be placed exactly at a corresponding point.

3 LineGuard is in $\exists\mathbb{R}$

In this section we prove that LineGuard is contained in $\exists\mathbb{R}$. Recall that the LineGuard problem is defined as follows: given a polygon P and guard number g, is it possible to place g line-segments in P (in particular the line segments are not allowed to have any parts outside of P) such that every point in P is visible from somewhere on some line segment? The proof technique is, naturally, similar to the proof that PointGuard is contained in $\exists\mathbb{R}$. The key difference is that we replace Lemma 2.2 with the more subtle Lemma 3.1, as follows.

Lemma 3.1. Fix polygon P and a set of guard line segments. Let X be the set of vertices of P union with the set of endpoints of guard line segments. Let $\mathcal{L}$ denote the set of lines joining pairs of points in X. Let $\mathcal{R}$ denote the faces of the regions in the arrangement inudced by $\mathcal{L}$. Let $\mathcal{C}$ be the set of centroids of regions in R.

If a guard line segment ℓ can see any point in region $R \in \mathcal{R}$ then ℓ can see all of R.

Proof. Fix region R. If the endpoint of any guard line segment can see R, then by Lemma 2.2 that endpoint can see all of R. Thus, it suffices to consider the case that no endpoint of any guard line segment can see R.

Now, suppose some point x on guard line segment ℓ can see a point in region R (where x is not an endpoint of ℓ). Let x' be the farthest to the left point of x on ℓ such that all points between x', x on the line can see a point of R (and x' is also not an endpoint). There must be some obstacle that prevents points to the left of x' from being able to see R. Specifically, the line of sight from x' to R must pass through some corner v of the polygon (it can't pass through an edge of P because edges of P are not internal to regions of the arrangement). Then, by Lemma 2.2 the vertex v can see all of P.

Let y, y' be the furthest clock-wise, and furthest counter-clock-wise points of R measured as angles from v. That is, if the line joining (v, y) clock-wise about v then it will no longer intersect R; an analogous statement holds for y'. Let F denote the field-of-view cone, defined by the region between the line joining (v, y) and the lint joining (v, y'). Suppose that F contained an endpoint z of a guard line segment. Then the line joining (z, v), which is part of the arrangement, would further subdivide the region R, which is impossible because we assumed that R is a region of the arrangement. Now, suppose that there is some obstacle in F that obstructs the portion of the guard line segment in F from seeing v. First, suppose that the obstacle is a vertex v' in F. In this case the line joining (v, v'), which is included in the arrangement, would further split region R, so this cannot happen. If the obstacle is an edge of the polygon, then it cannot intersect the guard line segment, and so would need to block x' from seeing v, which cannot be the case because x' can see v by definition. Hence there are no obstacles in this portion of F.

Thus, for any point z' in R, if we draw the line of sight from z' to v, and extend it back to the guard line segment, this entire segment described is unobstructed, so ℓ can see the point z'. Thus, ℓ can see all of

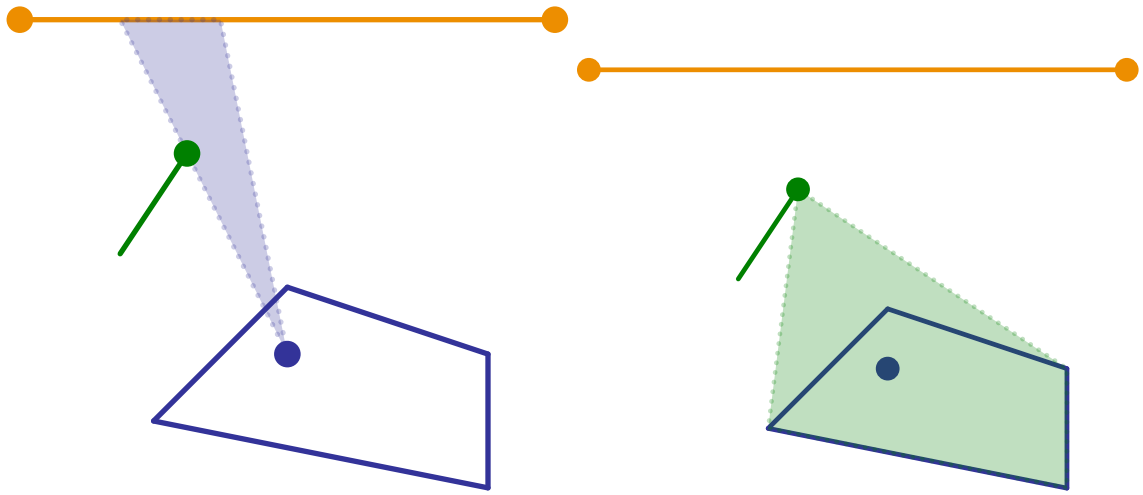

Figure 10: Assuming the region R isn't visible from an endpoint of the guard segment, there is some vertex $v \in V$ such that the guard segment can see v, and v can see a point of R. This means by Lemma 2.2 that v can see all of R.

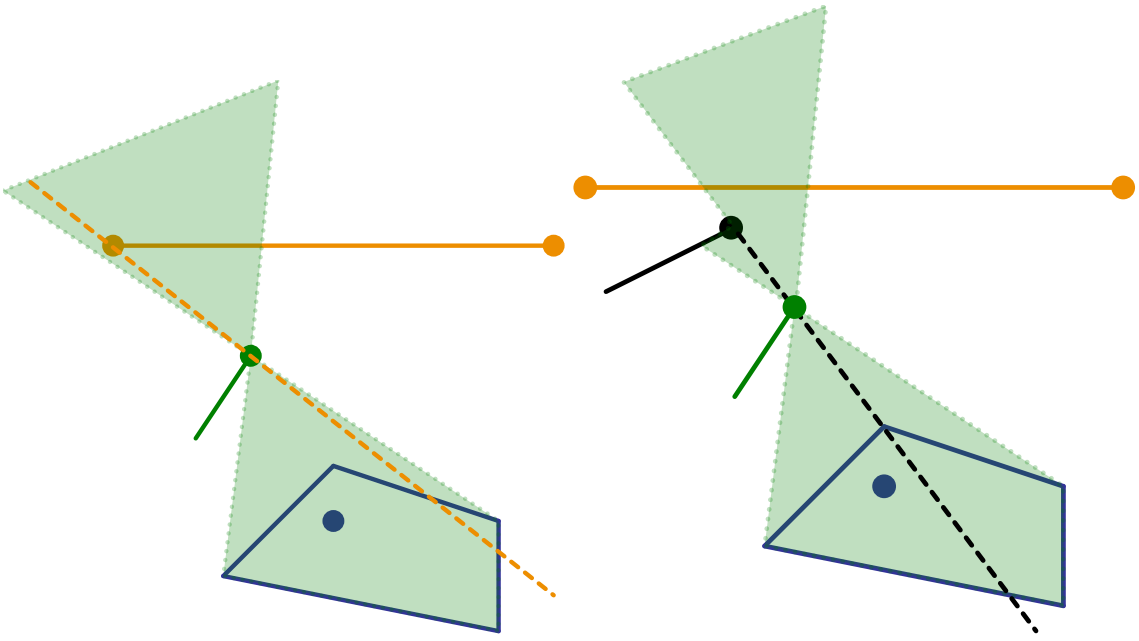

Figure 11: Projecting v's "field of view" of R backwards, we know that it can't contain either an endpoint of the guard segment or a vertex of P, since this would cause a line bisecting R in the arrangement. But this means that every line-of-sight from v to R can be extended backwards to a point on the guard segment, meaning the guard segment sees all of R.

R. □

Using Lemma 3.1 it is relatively straightforward to prove that LineGuard $\in \exists\mathbb{R}$.

Theorem 3.2. LineGuard $\in \exists\mathbb{R}$.

Proof. Let N be an upper bound on the maximum possible size of the set C of centroids of regions; note N is polynomial in the input size and is determined by P, g. Note also that, again as in the proof of Theorem 2.3, it is possible to write down a polynomial-sized $\exists\mathbb{R}$ formula SEE-CENTERS$((x_1, y_1) \dots, (x_N, y_N))$ that is satisfied if and only if its input points can see all elements of C. The formula looks like: first, encode the algorithm for finding the arrangement and taking centroids as an $\exists\mathbb{R}$ formula (possible because Diophantine equations can encode arbitrary computations), then add a constraint encoding that at least one of the input points sees each of those centroids.

Now, to show LineGuard $\in \exists\mathbb{R}$, the formula will first existentially quantify over the g endpoints of the guard segments, then it will existentially quantify over N "guard points". It will verify that each guard point lies on one of the guard segments, and that SEE-CENTERS(guard points) holds. This suffices to show $\exists\mathbb{R}$ containment: if the centroids are visible from the guard segments, then since there's at most N of them, there exist N points along the guard segments that collectively see all of them.

□

We also have the following corollary.

Corollary 3.3. Given a polygon P, guard number g, and quantity ε, it is possible to determine in $\exists\mathbb{R}$ whether there is a set of g side-length ε square guards that can see all of P.

Proof. Points in the interior of the square are irrelevant. Once we quantify over the placement of the squares we simply have a set of line segments and use the same procedure as in Theorem 3.2 to check if these line segments can see all of P. Of course, we could replace "side-length ε square" here with any fixed polygon or $\exists$-recognizable family of polygons. □

4 PaintGuard and PromiseGuard are in NP

In this section we consider the PaintGuard and PromiseGuard problems. We show that these problems are in NP; in particular this means that these problems are not $\exists\mathbb{R}$-complete, unless $\exists\mathbb{R} =$ NP. Our original motivation for considering these variants of PointGuard was a hope that they were $\exists\mathbb{R}$-hard, and would be useful components of a proof of $\exists\mathbb{R}$-hardness of LineGuard. We think the failure of these problems to be $\exists\mathbb{R}$-hard is interesting and sheds light on what is and is not responsible for the $\exists\mathbb{R}$-hardness of PointGuard. Beyond this, the PaintGuard and PromiseGuard problems are independently interesting[1].

Recall that the PaintGuard problem asks, given a list of n vertices specifying a polygon P, a list of n points within the polygon specifying the locations of ***paintings***, and g a number of guards, determine whether there is a placement of g guards that can see all the paintings.

Theorem 4.1. PaintGuard $\in$ NP.

Proof. Let X be the set of painting locations and vertices of the polygon. Let $\mathcal{L}$ denote the set of all lines passing through two points in X. Let $\mathcal{R}$ denote the set of cells in the arrangement induced by $\mathcal{L}$.

Now we give an NP algorithm for solving PaintGuard. Compute $\mathcal{R}$, and for each region $R \in \mathcal{R}$ compute a list of all paintings that are visible from region R. Specifically, to check if region R can see painting x, draw the line segment between x and the centroid of R and check if this line segment intersects any edge of the polygon. We are using a key fact here, namely that Lemma 2.2 implies that if a painting x can see the centroid of a region $R \in \mathcal{R}$, then the painting can see all of region R. After creating these lists for each region, non-deterministically choose a set of g regions in $\mathcal{R}$ to place guards in, and check if the union of the paintings visible from these chosen regions contains all paintings.

The correctness and efficiency of the algorithm are clear. □

[1] We recently found a mention of the NP hardness of PaintGuard known in the literature [Sta23], but are choosing to still include the proof here because it's simple and nice.

Recall the PromiseGuard problem: We are given a polygon P, guard number g, and radius ε, expressed as a B' bit binary decimal. The problem is to distinguish between two cases: (1) P can be guarded by g point guards. (2) P cannot be guarded by g radius ε circle guards.

Theorem 4.2. PromiseGuard $\in$ NP.

Proof. Our algorithm is as follows. Non-deterministically place g side-length $\varepsilon/10$ square guards centered at lattice points in the grid formed by partitioning $[0,1]^2$ into $\varepsilon/100 \times \varepsilon/100$ sized squares[2]. Then, check if these guards can see all of P; we describe how to perform this check momentarily. If the guards can see all of P output that we are in case (1). Otherwise, output that we are in case (2).

The square guards can see all of P if and only if the line segments defining their boundaries can see all of P. To check this we construct the same arrangement as in Theorem 3.2: we let X be the set of endpoints of line segments and vertices of P, and consider the arrangement induced by joining all pairs of points in X. By Lemma 3.1 we have that if an edge ℓ of a square guard can see a single point in region R then ℓ can see all of R. Thus it suffices to check whether for each centroid c of a region R there is an edge ℓ of a square guard that can see c. Because sight is symmetric, we can equivalently check that each centroid c can see some edge ℓ. To perform this check we construct a *second* arrangement. We let X' be the set of vertices in the polygon unioned with a centroid $\{c\}$, and let $\mathcal{R}'$ be the arrangement induced by the lines joining every pair of points in X'. By Lemma 2.2 we have that for every region $R \in \mathcal{R}'$, if c can see a single point in R then c can see all of R. Now, we compute the set of regions $R \in \mathcal{R}'$ visible from c, and check if there is any edge of a guard square that has non-empty intersection with one of these regions R visible from c.

Because the guard squares were placed at points on a lattice which is not too fine, all of the arrangements mentioned in the above analysis are defined by "nice coordinates" so all the computations described above can be performed in $\text{poly}(n, B, B')$ time. Thus, our algorithm runs in NP.

We now prove the correctness of our algorithm.

Claim 4.3. If P is guardable by g point guards then our algorithm reports that we are in case (1).

Proof. Suppose that there exists G, a set of g point guards that can see all of P. For each point $x \in G$, there is a lattice point x' in our $\varepsilon/100 \times \varepsilon/100$ grid such that a square of side-length $\varepsilon/10$ centered at x' covers x. Thus, there is a placement of square guards onto the grid that can see all of P, and our NP algorithm can non-deterministically guess this placement. □

Claim 4.4. If P cannot be guarded by g radius ε guards then our algorithm reports that we are in case (2).

Proof. Suppose that our algorithm outputs that we are in case (1). Then, there must be a placement of $\varepsilon/10$ side-length squares onto points in the grid that could see the entire polygon. Then, radius ε circles centered at the same points can also see all of P, so we are indeed in case (1). □

□

5 3D line guard is $\exists\mathbb{R}$ hard

In 3DLineGuard, you are given as input P: a union of intersection of half-spaces and a number g. The problem is to determine whether it is possible to place g line segments in P such that every point in P is visible from some point on some line segment. Again, we say that point x is visible from point y if the line segment joining x, y lies completely within P.

Theorem 5.1. 3DLineGuard is $\exists\mathbb{R}$-hard.

Proof. Recall that in Abrahamsen, Adamaszek, and Miltzow's reduction proving the $\exists\mathbb{R}$-hardness of the Art Gallery Problem, there exist g many ***fixed guard points*** and ***guard segments*** (with nice rational coordinates) of the polygon such that any solution to the problem, if it exists, must place exactly one guard along each guard segment and one guard at each fixed point. We will exploit this property to show $\exists\mathbb{R}$ hardness of 3DLineGuard by reduction from PointGuard. Our reduction will be slightly unsatisfying because

[2] Recall that P is defined to be contained in $[0,1]^2$.

it will make heavy use of planes with 0 width (we hoped for a while that maybe we could modify this reduction to use regions of very small width instead of literal planes, but our approach for doing so was a reduction from PromiseGuard, which we eventually realized was in fact in NP).

The idea of the reduction is to take the original PointGuard instance as an infinitely thin plane, and then for every guard segment or fixed guard point, intersect upwards a tall thin plane strip or line segment respectively, as in Figure 12. Then, add $100g$ many separate horizontal plane regions intersecting each of those vertical lines and strips, as in Figure 13.

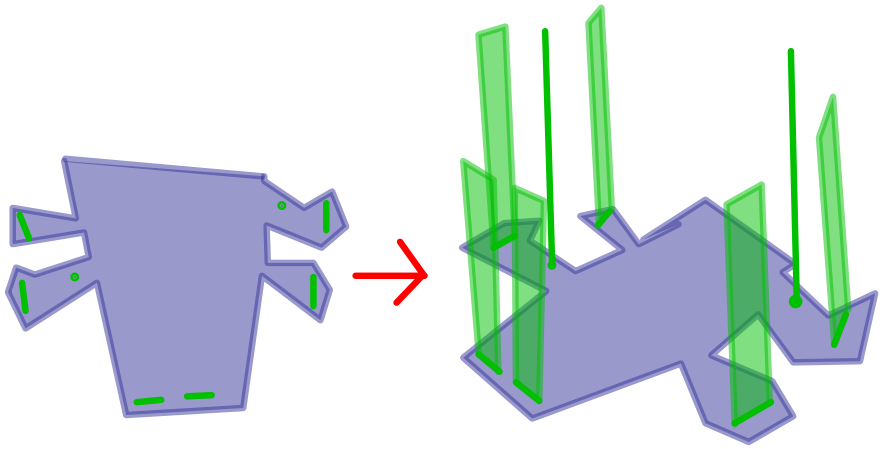

Figure 12: Converting a 2-dimensional art gallery into a 3-dimensional art gallery formed as the union of several infinitely thin planes and lines.

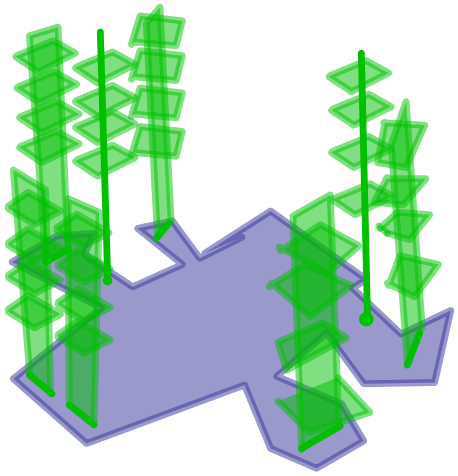

Figure 13: Each of these upward strips and lines gets a bunch of horizontal regions crossing it. This ensures that in order to guard each of them there must be a guard segment passing vertically up (it's simply impossible to see all those horizontal regions otherwise).

Now, suppose we want to guard the gallery with g many line guards. if the original PointGuard instance had a solution with g point guards, we could take this solution, and extend each point up into a line segment passing up its corresponding vertical line or strip. The single point intersections with the polygon would cover the whole base, and the segments passing up through the strips would let the guards see every horizontal plane region, so the whole gallery would be covered. On the other hand, suppose the original PointGuard instance didn't have a solution with g point guards – in that case, in order to cover the base at least one of our line guards must intersect with the base in more than one point (no other points in the construction can see any points in the base – this is why infinite thinness is necessary). To intersect with the base in more

than one point, the guard line must lie entirely within the base. Since we have g many line guards and g many vertical strips/lines, this means at least one of these vertical objects must not have a vertical guard running up through it – but then it is impossible to see all of it. So, this 3DLineGuard instance is solvable if and only if the original PointGuard instance was.

□

6 Conclusion

There are lots of interesting Art Gallery variants. We have considered a few here, and shown some basic things about their hardness. The most pressing open question we are left with is: Is the two-dimensional problem LineGuard $\exists\mathbb{R}$-hard? Doing this problem in 3 dimensions definitely feels like cheating – the two-dimensional version almost certainly requires a lot more tricky constructions.

It would also be interesting to consider hardness of approximation variants of the problem, where you need to distinguish between an art gallery being guardable by g guards and not guardable by, e.g., $100g$ guards. There are gap NP-hardness results known [ESW01], but as far as we're aware nothing is known about corresponding notions of gap $\exists\mathbb{R}$-hardness.

Finally there are several problems spiritually related to the art gallery problem that could be interesting to show $\exists\mathbb{R}$ hardness for, if true. One of these is the classic ***watchman route*** problem of minimizing the length of a route through a polygon that sees every point, and another is the ***visibility-based pursuit evasion*** problem of determining whether a set of mobile guards has a movement strategy that will guarantee them to find an arbitrarily-fast robber. Both of these problems have been studied, and are known to be NP-hard, but not known to be in NP [SY92; CN86; Gui+97]. It seems very plausible that both of these problems could be $\exists\mathbb{R}$-hard. However, although these problems seem related to the art gallery problem in flavour, it seems like any reduction would have to look radically different.

References

[AAM17] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. "Irrational Guards are Sometimes Needed". In: *33rd International Symposium on Computational Geometry (SoCG 2017)*. Ed. by Boris Aronov and Matthew J. Katz. Vol. 77. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017, 3:1–3:15. ISBN: 978-3-95977-038-5. DOI: 10.4230/LIPIcs.SoCG.2017.3. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.SoCG.2017.3.

[AAM18] Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. "The art gallery problem is $\exists\mathbb{R}$-complete". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2018. Los Angeles, CA, USA: Association for Computing Machinery, 2018, pp. 65–73. ISBN: 9781450355599. DOI: 10.1145/3188745.3188868. URL: https://doi.org/10.1145/3188745.3188868.

[Can88] John Canny. "Some algebraic and geometric computations in PSPACE". In: *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*. STOC '88. Chicago, Illinois, USA: Association for Computing Machinery, 1988, pp. 460–467. ISBN: 0897912640. DOI: 10.1145/62212.62257. URL: https://doi.org/10.1145/62212.62257.

[CN86] Wei-pang Chin and Simeon Ntafos. "Optimum watchman routes". In: *Proceedings of the second annual symposium on Computational geometry*. 1986, pp. 24–33.

[ESW01] Stephan Eidenbenz, Christoph Stamm, and Peter Widmayer. "Inapproximability Results for Guarding Polygons and Terrains". In: *Algorithmica* 31 (Sept. 2001), pp. 79–113. DOI: 10.1007/s00453-001-0040-8.

[Gui+97] Leonidas J Guibas et al. "Visibility-based pursuit-evasion in a polygonal environment". In: *Algorithms and Data Structures: 5th International Workshop, WADS'97 Halifax, Nova Scotia, Canada August 6–8, 1997 Proceedings 5*. Springer. 1997, pp. 17–30.

[LL86] D. Lee and A. Lin. "Computational complexity of art gallery problems". In: *IEEE Transactions on Information Theory* 32.2 (1986), pp. 276–282. DOI: 10.1109/TIT.1986.1057165.

[Sta23] Jack Stade. *The Point-Boundary Art Gallery Problem is $\exists\mathbb{R}$-hard*. 2023. arXiv: 2210.12817 [cs.CG].

[SY92] Ichiro Suzuki and Masafumi Yamashita. "Searching for a Mobile Intruder in a Polygonal Region". In: *SIAM Journal on Computing* 21.5 (1992), pp. 863–888. DOI: 10.1137/0221051. eprint: https://doi.org/10.1137/0221051. URL: https://doi.org/10.1137/0221051.