# A Practical Algorithm with Performance Guarantees for the Art Gallery Problem

**Simon B. Hengeveld**
Université Rennes 1, France

**Tillmann Miltzow**
Utrecht University, The Netherlands

## Abstract

Given a closed simple polygon $P$, we say two points $p, q$ see each other if the segment $\text{seg}(p, q)$ is fully contained in $P$. The art gallery problem seeks a minimum size set $G \subset P$ of guards that sees $P$ completely. The only currently correct algorithm to solve the art gallery problem exactly uses algebraic methods. As the art gallery problem is $\exists\mathbb{R}$-complete, it seems unlikely to avoid algebraic methods, for any exact algorithm, without additional assumptions.

In this paper, we introduce the notion of *vision-stability*. In order to describe vision-stability consider an *enhanced* guard that can see "around the corner" by an angle of $\delta$ or a *diminished* guard whose vision is by an angle of $\delta$ "blocked" by reflex vertices. A polygon $P$ has vision-stability $\delta$ if the optimal number of enhanced guards to guard $P$ is the same as the optimal number of diminished guards to guard $P$. We will argue that most relevant polygons are vision-stable. We describe a *one-shot vision-stable* algorithm that computes an optimal guard set for vision-stable polygons using polynomial time and solving one integer program. It guarantees to find the optimal solution for every vision-stable polygon. We implemented an *iterative vision-stable* algorithm and show its practical performance is slower, but comparable with other state-of-the-art algorithms. The practical implementation can be found at: `https://github.com/simonheng/AGPIterative`. Our iterative algorithm is inspired and follows closely the one-shot algorithm. It delays several steps and only computes them when deemed necessary. Given a chord $c$ of a polygon, we denote by $n(c)$ the number of vertices visible from $c$. The *chord-visibility width* ($\text{cw}(P)$) of a polygon is the maximum $n(c)$ over all possible chords $c$. The set of vision-stable polygons admit an FPT algorithm when parameterized by the chord-visibility width. Furthermore, the one-shot algorithm runs in FPT time when parameterized by the number of reflex vertices.

**2012 ACM Subject Classification** Theory of computation → Computational geometry

**Keywords and phrases** Art Gallery, Parametrized complexity, Integer Programming, Visibility

**Digital Object Identifier** 10.4230/LIPIcs.SoCG.2021.44

**Related Version** *Full Version*: `https://arxiv.org/abs/2007.06920`

**Supplementary Material** *Software (Source Code)*: `https://github.com/simonheng/AGPIterative` archived at `swh:1:dir:36fab957090b1005b19be68b37849e23e23fe973`

**Funding** *Simon B. Hengeveld*: MULTIBIOSTRUCT (ANR-19-CE45-0019).
*Tillmann Miltzow*: NWO Veni grant EAGER.

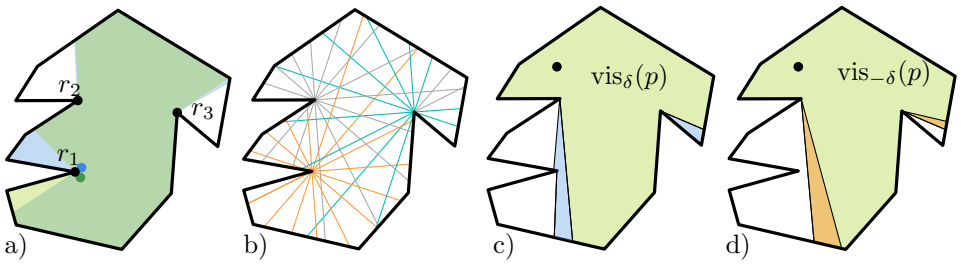


**Figure 1** a) A small positional change largely influences how much visibility of the two guards is blocked by $r_1$, but only has a small effect on the way that $r_2$ or $r_3$ blocks the visibility of the guards. b) Shooting rays from reflex vertices. c) Enhanced visibility region. d) Diminished visibility region.

## 1 Introduction

For most algorithmic problems, there is a *practice-theory gap*. That is a gap between algorithms that perform best in practice and that perform best in theory. A striking example is the euclidean traveling salesperson problem. It has long been known to be NP-hard, see [46] for the most recent hardness results. Despite those theoretical hardness results, in practice, new records of optimally solved large scale instances keep making the news [11].

In this work, we narrow this gap for the art gallery problem. We present several closely related algorithms. Some of them have provable performance guarantees under some mild assumptions. Other algorithms perform comparably well to the best state-of-the-art practical algorithms on the art gallery problem, however at the cost of losing these theoretical performance guarantees.

**Motivation.** The theoretical study [57] of algorithms has three main motivations:

**Prediction:** Given a specific algorithm, we want to predict how well it will perform. This can help us to decide if we want to implement it in the first place.

**Explanation:** Assuming that we know how well an algorithm performs, we want to find an underlying reason for its performance.

**Invention:** Can we design better algorithms in practice?

These goals are an important measure of success. In an ideal world, theory and practice go hand-in-hand. Theory researchers design and analyze algorithms and practically oriented researchers implement those algorithms and rigorously test them. These tests help to adapt models, assumptions, and performance measures. At the same time, practical researchers may find other approaches fruitful and then in turn theoretical researchers need to find out why those approaches work. This *theory-practice cycle* should ideally lead to a very good theoretical understanding and very fast practical algorithms. Unfortunately, as we will lay out, the study on the art gallery problem has two almost independent lines of research. One solely theoretical and another purely practical one. Thus our theoretical study has limited value to predict, to explain, or to invent. We believe that a solid theoretical understanding will be also useful for experimental research. Our main contribution is to narrow this gap and give a starting point to the theory-practice cycle.

**Related work.** Let us start by considering practical research. Various researchers implemented algorithms and found the optimal solution on synthetic instances of up to 500 vertices [68, 36, 21, 49, 25, 20, 26, 10, 27, 19]. The idea is to discretize the problem and hope that the solution to the discretized problem also gives the optimal solution to the original problem. To be more specific, the approach is to generate a *candidate* set $C$ and a *witness* set $W$, then compute the optimal way to guard $W$ using the minimum number of guards in $C$. In an iterative manner, more candidates and witnesses are generated until the optimal solution is found. While these algorithms *usually* find the optimal solution, there is a simple polygon on which these algorithms run *forever* [2]. We do not know sufficient conditions under which these practical algorithms would give the optimal solution in a finite amount of time.

Let us continue by considering worst-case optimality. We know that the art gallery problem is decidable using tools from real algebraic geometry. The idea is to encode guards by real numbers and use polynomial equations and inequalities to encode visibility [32]. Currently, researchers working on solving polynomial equations repeatedly report that 12 is the maximum number of variables they could handle, using exact methods. (The second author asked this at several conferences and workshops.) To express the art gallery problem by polynomial equations has a considerable blow-up and needs existentially and universally

quantified variables. To summarize, we would be surprised if these exact methods could even find an optimal solution of size two for the art gallery problem. As the art gallery problem is $\exists\mathbb{R}$-complete [3], we know that methods from real algebraic geometry are unavoidable for any exact algorithm, without additional assumptions. The $\exists\mathbb{R}$-completeness of the art gallery problem [3] is a very good explanation of why researchers were not able to find algorithms that avoid the aforementioned algebraic methods. Furthermore, it explains why it is hard to prove that practical discretization schemes work from a theoretical perspective. To be precise, if there would be a discretization scheme with worst-case guarantees then NP $\neq \exists\mathbb{R}$.

To the reader not familiar with the *existential theory of the reals* ($\exists\mathbb{R}$), it may be insightful to get some background information. It is defined in an analogously to NP. Recall that a problem is in NP if for every input there is a binary witness $w \in \{0,1\}^*$ and a verification algorithm that runs on the word RAM in polynomial time. We say a problem is contained in $\exists\mathbb{R}$ if for every input there is a *real* witness $w \in \mathbb{R}^*$ and a verification algorithm that runs on the *real RAM* in polynomial time. Note that the usage of witness in the context of complexity classes is very different from the usage of witness in the context of the art gallery problem [35]. The complexity class $\exists\mathbb{R}$ is important as it gives a precise characterization of many important algorithmic problems [1, 3, 5, 15, 22, 23, 29, 31, 34, 37, 42, 47, 51, 52, 53, 56, 58, 59, 60, 62, 63, 64, 4, 28]. None of these algorithmic problems are known to be contained in NP. The problem to show NP-membership is that it is seemingly impossible to describe a discrete witness of polynomial-size. The complexity-class $\exists\mathbb{R}$ relates the issue for each of those algorithmic problems. To be more specific, either all of those algorithmic problems admit a polynomial size witness or none of them do. This also makes it difficult to discretize any of those problems as any such discretization, usually, would also give rise to a polynomial sized witness.

Maybe the main approach to overcome the discretization problem for the art gallery problem was to consider variants. Specifically, restricting guard placements to the vertices of the polygon avoids the discretization problem, see [55, 14, 45, 7, 54, 38]. While many of these results are very intricate and innovative, we are not aware of any of these algorithms being implemented in practice. We think that there are two main reasons for this. In practice, the discretization problem is solved to a fairly large degree. That is the candidates and witnesses are usually sufficient to determine an optimal solution, even if this cannot be proven, i.e., restricting the guards to the vertices has only a small added benefit. The second reason is that theoretical research focuses on the study of approximation algorithms, which may not be so relevant in practice.

The study of approximation algorithms is mainly motivated by the fact that the art gallery problem and many of its variants are NP-hard [33, 50, 61, 18]. Thus we cannot expect a polynomial-time algorithm, assuming P $\neq$ NP. However, there are four important facts to consider when dealing with the practical performance of approximation algorithms:

**Discretization:** While the concept of approximation relaxes the worst-case condition, it has not been shown to help sufficiently to find a nice discretization [17]. (See below for a detailed discussion.)

**IP-solvers:** For some inexplicable reason, once we have a discretization of the art gallery problem, IP-solvers often find the optimum for that discretized problem very fast in practice. (Often only 10% of the total running time is spent on solving IPs [27].) Thus the aim of having a polynomial-time algorithm is not so important.

**Visibility:** The real bottleneck in practical performance seems to be computing visibilities between candidates and witnesses. Approximation algorithms have to do these computations as well. Thus they may just not be any faster.

**Non-optimality:** By definition approximation algorithms do not give the optimal solution. This may just be a too high price to pay.

**Contribution.** In this paper, we introduce the notion of vision-stability. We argue that most practical polygons are vision-stable. Using this assumption, we give a theoretical solution to the discretization problem. Based on this discretization, we develop the *one-shot (vision-stable)* algorithm that is *guaranteed* to find an optimal solution for every vision-stable polygon. The algorithm takes polynomial preprocessing time and thereafter solves one integer program. In particular, this shows that the art gallery problem is in NP for vision-stable polygons. We refine the algorithm and present the *iterative (vision-stable)* algorithm. The iterative algorithm delays many steps of the one-shot algorithm. Both algorithms are *reliable*, in the sense that even if the input polygon is not vision-stable, the reported result is correct. In order to make the iterative algorithm comparable in performance to the state-of-the-art, we cut many corners in our implementation. The downside of this approach is that we are not able to show theoretical performance guarantees for this practical version of the iterative algorithm. On the upside, our implementation has slower but similar performance to previous practical algorithms.

We believe that the concept of vision-stability contributed to all three main goals, as mentioned above. It gives an *explanation* of why the discretization problem is solvable in practice. It led to the *design* of a new practical algorithm, that, as we will see, works quite differently from previous algorithms in many aspects. And finally, it *predicted* correctly that this algorithm is feasible, which we verified in practice.

Let us emphasize here that there is still a large gap, between theory and practice. Our predicted running times are far worse than the observed practical running time. Furthermore, in practice, we used many techniques for which it seems very hard to give a profound theoretical explanation of why they work so well. We hope our work contributes to a deepened theoretical understanding of the art gallery problem.

**Vision-stability.** Before we formally define the notion of vision-stability, which is the key concept of this paper, let us give some basic intuition on discretizing the art gallery problem. The aim of the discretization process is to define a suitable *candidate set* $C$ such that $\text{opt}[C]$, the optimum guard set restricted to $C$, is "pretty close" to the actual optimum opt. After defining $C$ in a suitable fashion, we can compute $\text{opt}[C]$ by solving an integer program. We know [17] that the grid $\Gamma = w\mathbb{Z}^2 \cap P$ with a small enough width $w$ is a good such candidate set in the sense that $|\text{opt}[\Gamma]| \leq 10\,|\text{opt}|$, under some mild general position assumptions. Using smoothed analysis [30, 35] and a suitable random model of perturbation, we even know that $|\text{opt}[\Gamma]| = |\text{opt}|$, with high probability. In summary, a fine enough grid contains the optimal solution, under certain assumptions, however, the size of the candidate set $C$ is also important. The grid $\Gamma$ as described above is huge. Thus computing $\text{opt}[\Gamma]$ is infeasible in practice, making it very desirable to attain a candidate set $C$ with similar properties as $\Gamma$ and of polynomial size.

As a first step, we realize that a *uniformly distributed* candidate set would either be too large or too coarse. Thus at some spots, we want the candidate set to be denser and at other places in the polygon, we want it to be more sparse. Let us say two guard positions $g, g'$ have almost the same visibility region then hopefully it is not so important to keep track of both positions, but keeping only one of the positions is hopefully sufficient. Thus if the visibility regions of $g$ and $g'$ are very different, we should potentially include both of them in our candidate set $C$. Another, almost trivial, observation is that a small movement of $g$ towards $g'$ may dramatically change visibility regions, if $g$ and $g'$ are close to a reflex vertex. If $g$ and $g'$ are both very far away from all reflex vertices their visibility regions change almost not at all, see Figure 1 a). Those two observations suggest that we may want to pick a candidate set that is denser closer to reflex vertices and more sparse farther away from reflex vertices.

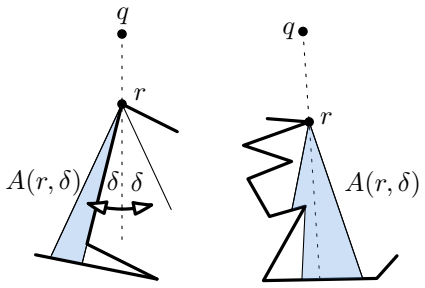


**Figure 2** A ray is rotated around a reflex vertex $r$. It defines a region that is either added or removed from the visibility region.

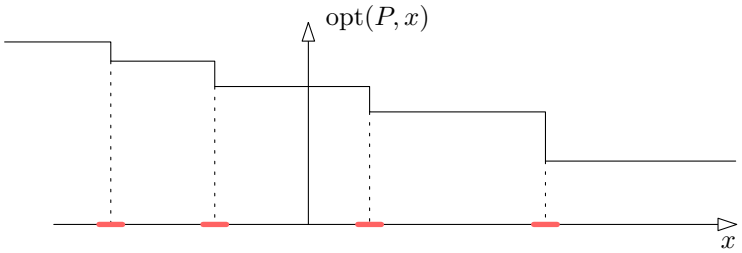


**Figure 3** On the $x$-axis, we have the value by which we either diminish or enhance guards. On the $y$-axis we display the optimal number of guards. The function $\mathrm{opt}(P, x)$ only takes discrete values and is monotonically decreasing. Thus it has a finite number of breakpoints.

This motivates the following approach. Shoot rays from every reflex vertex, such that the angle between any two rays is at most some given angle $\delta$. This defines an arrangement $\mathcal{A}$. All intersection points of the rays within the polygon $P$ define our candidate set $C$, see Figure 1 b). On an intuitive level, for every point $p \in P$ there is a candidate $c \in C$ such that the visibility regions of $p$ and $c$ are similar. When $r$ denotes the number of reflex vertices, we get a total number of rays upper bounded by $O(\frac{r}{\delta})$. Thus, the candidate set has size $O(\frac{r^2}{\delta^2})$.

We are now ready for the formal definition of vision-stability. Given a simple polygon $P$ and a point $q$, the visibility region of $q$ is defined as $\mathrm{vis}(q) = \{x \in P : x \text{ sees } q\}$. Let $r$ be a reflex vertex of $P$ and we assume that $q$ sees $r$. Then, given some $\delta > 0$, we can define the *visibility enhancing region* $A = A(q, r, \delta)$ as follows. Rotate a ray $v$ with apex $r$ by some angle of $\delta$, clockwise and counter-clockwise. At each time of the rotation, the ray $v$ defines a maximal segment inside $P$ with endpoint $r$. In some cases, the segment is the single point $r$, see Figure 2. The region $A(r, \delta)$ is the union of all those segments. For some $\delta > 0$, we define the $\delta$-enhanced visibility region $\mathrm{vis}_\delta(q)$ of $q$ as $\mathrm{vis}(q)$ and for every suitable reflex vertex, we add the region $A(r, \delta)$. We define the $\delta$-diminished visibility region $\mathrm{vis}_{-\delta}(q)$ of $q$ as $\mathrm{vis}(q)$ after we remove the regions $A(r, \delta)$, for every applicable reflex vertex $r$. To be precise, we define $\mathrm{vis}_\delta(q)$ to be a closed set, both for $\delta > 0$ and $\delta \leq 0$. Given a polygon $P$, we say that $G$ is *$\delta$-guarding* $P$ if $\bigcup_{g \in G} \mathrm{vis}_\delta(g) = P$. We denote by $\mathrm{opt}(P, \delta)$ the size of the minimum $\delta$-guarding set. For brevity, we denote $\mathrm{opt}(P, 0)$, merely by $\mathrm{opt}(P)$ or, if $P$ is clear from the context, by opt. We say that a polygon $P$ is *vision-stable* or equivalently has vision-stability $\delta > 0$, if $\mathrm{opt}(P, -\delta) = \mathrm{opt}(P, \delta)$. Note that for $\delta' > \delta > 0$, it holds that $P$ has vision-stability $\delta'$ implies that $P$ also has vision-stability $\delta$. Thus the smaller the vision-stability the larger we expect the candidate set to be. To avoid confusion, at no time are we interested in actually computing $\mathrm{opt}(P, x)$, for any value $x \neq 0$. The notion of vision-stability is purely a theoretical concept in order to formulate assumptions on the underlying polygon.

Here, we will give a summary of the justification of vision-stability. First, without any assumption, we cannot avoid algebraic methods unless NP = $\exists\mathbb{R}$. Furthermore, there is an argument to be made related to smoothed analysis. Consider $\text{opt}(P, x)$ as a function $f$ of $x$, see Figure 3. Clearly, $f$ takes only a discrete number of values, by definition. Furthermore, the function is monotonically decreasing, as $\text{vis}_x(p) \subseteq \text{vis}_{x'}(p)$, if $x \leq x'$. Thus the function $f$ has only a finite number of breakpoints. If a breakpoint happens to be at zero then $P$ is not vision-stable. Intuitively, this seems unlikely. See the full version for more details [40].

**Discretization.** Using vision-stability, we exhibit a candidate set $C$ of polynomial size. The idea is to use the vertices of arrangement $\mathcal{A}$ from Figure 1 b). For technical reasons, we will not use the arrangement $\mathcal{A}$, but a refinement of it. Note that the smaller the vision-stability, the weaker the assumption on the underlying polygon, and thus the larger the required candidate set.

▶ **Theorem 1** (Candidate Set). *Given a polygon with vision-stability $\delta$ and $r$ reflex vertices, it is possible to compute a candidate set $C$ (of size $O(\frac{r^4}{\delta^2})$) in polynomial time on a real RAM. The candidate set $C$ contains an optimal solution.*

Although some of the details of the proof are tedious and involved, we see the main contribution on a conceptual level. It is surprising to us that the vision-stability was not formulated earlier. In particular, regarding the popularity of the art gallery problem within computational geometry and the problematic lack of algorithmic results. Let us stress that while we use augmented and diminished visibilities as abstract concepts, we are only interested in solving the original art gallery problem.

Let us give an intuition of the proof idea of Theorem 1. As mentioned before, we use all the vertices of a refinement of the arrangement $\mathcal{A}$, as in Figure 1 b), as our candidate set $C$. Consider a minimum size set $G_0$ that is $(-\delta)$-guarding $P$. Replace each guard $g \in G_0$ by a guard $g' \in C$ that is "close by". This gives a new solution $G_1 \subseteq C$ of equal size. Using that $G_0$ is $(-\delta)$-guarding, we can show that $G_1$ is guarding $P$ in the usual sense. As $\text{opt}(P, -\delta) = \text{opt}(P, 0) = \text{opt}(P, \delta)$, we can conclude that $G_1 \subseteq C$ is of minimum size. The technical demanding part of the proof is to show that for every point $p \in P$ there exists a point $c \in C$ such that $\text{vis}_{-\delta}(p) \subseteq \text{vis}(c)$.

Theorem 1 answers an open question posed by several authors of related works [10, 66, 67, 27]. For instance, De Rezende et al. [27] states "*Therefore, it remains an important open question whether there exists a discretization scheme that guarantees that the algorithm always converges [...].*"

In the next paragraph, we discuss how the discretization scheme leads to a correct algorithm that avoids algebraic methods.

**One-Shot vision-stable algorithm.** Note that in practice there does not exist an algorithm which can be used to compute whether or not a polygon has vision-stability $\delta$. Therefore, it is important that our algorithms work correctly even if the underlying polygon is not vision-stable. Specifically, our algorithms will either compute the optimal solution or report that the input polygon had lower vision-stability than specified by the user. We say our algorithms are *reliable*, as they never return an incorrect answer.

▶ **Theorem 2** (One-Shot vision-stable Algorithm). *Let $P$ be an $n$ vertex polygon, with $r$ reflex vertices. We assume that a suggested value for $\delta$ is given as part of the input. Then the one-shot algorithm has a preprocessing time of $O(\frac{r^8}{\delta^4} \log n + n \log n)$ on a real RAM and additionally solves exactly one integer program. The algorithm either returns the optimal solution or reports that the given suggested value for $\delta$ is incorrect.*

This is the first algorithm for the classical art gallery problem that avoids using algebraic methods and gives an exact solution.

The core idea of the algorithm is to utilize the candidates from Theorem 1 and use them to build a set-cover instance, which can then be solved using integer programming.

Let us point out that next to vertex-candidates, we also use faces as candidates. This is a distinct feature of our algorithm. All previous algorithms used only points as candidates. In this way, we can easily check that we have not missed a better solution. Say the algorithm returns a solution $G$ of size $k$ and suppose for the purpose of contradiction that there would be a smaller solution $G'$ of size $k' < k$. Pick for each $g \in G'$ a face containing $g$. (If $g$ lies on an edge or a vertex of $\mathcal{A}$ make an arbitrary choice.) This defines a set $\mathcal{F}$ of faces with $|\mathcal{F}| = k'$. Now we arrive at a contradiction, as $\mathcal{F}$ is a valid guarding of the polygon $P$. The algorithm primarily minimizes the number of used guards (vertex or face guards), but among the minimum size solutions, it prefers vertex guards over face guards. We say that using vertex guards is a *soft constraint*.

Similar to previous algorithms, we also use witnesses. In the context of the art gallery problem, we require only that all the witnesses are seen, instead of the entire polygon. The hope is that the computed guards will also see the entire polygon. This is a second important step to discretize the problem. In our case, the witness set $W$ will be all *faces* and vertices of some arrangement $\mathcal{A}$. This is a second feature that makes our algorithm unique. As all the faces of $\mathcal{A}$ are covering $P$, seeing all faces guarantees that $P$ is completely seen. We impose the *hard constraint* that all point-witnesses are seen and the *soft constraint* that each face-witness must be seen by at least one guard.

Due to the hard constraints, we know that our solution is at most the optimal size (theoretically it could be smaller, as face-guards are more powerful than point-guards). If we see all face-witnesses, then we know that the guards are actually guarding $P$. Furthermore, if all guards are points, we know that we have found a minimum size point guard set. In case that one of the soft-constraints is violated, we will be able to deduce that the underlying polygon was not vision-stable, with the suggested value $\delta$ given in the input. The last statement is the technically demanding part of the proof. One of the central concepts of the proof is the *angular capacity of a face*. It measures how small a face is while taking into account the distance to reflex vertices.

*For the remainder of the paper, whenever we refer to a candidate, witness or a guard, we could mean either a point or a face, if not further specified.*

**Parameterized algorithms.** As the size of the integer program of the one-shot algorithm only depends on $r$ and $1/\delta$, it exhibits an FPT algorithm, with respect to the number of reflex vertices $r$, for every fixed $\delta$. The most natural parameter for the art gallery problem is the solution size. As the art gallery problem is W[1]-hard, when parameterized by the solution size [18], research focused on other parameters [9, 7, 8, 12, 43, 44, 6]. Specifically, Agrawal et al. [7] described an elegant FPT algorithm for the art gallery problem. They considered three variants of the art gallery problem defined by restricting guard positions and the part of the polygon that needs to be guarded. By considering the number of reflex vertices as the parameter, they answered a question by Giannopoulos, for those variants. "*Guarding simple polygons has been recently shown to be W[1]-hard w.r.t. the number of (vertex or point) guards. Is the problem FPT w.r.t. the number of reflex vertices of the polygon?*" [39]. We answer the same question, with respect to vision-stable polygons and the classic variant of the art gallery problem.

▶ **Corollary 3** (Reflex-FPT Algorithm)**.** *Given a vision-stable polygon, with any fixed vision-stability. The one-shot algorithm is FPT with respect to the number of reflex vertices.*

**Practical algorithm.** Although the one-shot algorithm does not require algebraic methods and only polynomial preprocessing time, it is still way too slow to be considered practical. Note that the performance bottleneck is not solving the integer program, but computing visibilities. As a next step, we develop the *iterative vision-stable* algorithm. It is practical and follows the basic principle of the one-shot algorithm. Ideally, we would also like to show provable performance guarantees for the iterative algorithm. Note that the statement that an algorithm is both practical and has theoretical guarantees must be taken with caution. Once we have a *practical* algorithm $\mathcal{P}$ and a *theoretical* algorithm $\mathcal{T}$, we can easily get a third algorithm $\mathcal{B}$ that has *both* properties as follows. Run algorithm $\mathcal{P}$ within the running time bound of $\mathcal{T}$. If $\mathcal{P}$ does not return a solution abort and run $\mathcal{T}$. Here, $\mathcal{T}$ serves as a *safe guard* for $\mathcal{P}$. Clearly $\mathcal{B}$ performs as well in practice as $\mathcal{P}$ and has the theoretical bounds of $\mathcal{T}$. Thus, when we say that we show theoretical performance guarantees of some algorithm, we should really ask ourselves, if we show those guarantees for the algorithm that we actually use or for some safe guards that aren't ever used in practice.

The core idea of the iterative algorithm is as follows. We start with a very coarse arrangement $\mathcal{A}$. Using the faces and the vertices of $\mathcal{A}$, we define a candidate set $C$ and a witness set $W$. We compute which candidates see which witnesses. This enables us to build an integer program, that tries to find a minimum guard set $G \subseteq C$, as described for the one-hot algorithm. Note that vertices and faces may serve as guards and some face-witnesses may be unguarded. As a secondary objective function, the integer program tries to minimize the number of face-guards used in $G$ and the number of unseen face-witnesses. If the guard set $G$ contains only point guards and sees all face-witnesses, the iterative algorithm reports the optimal solution. Otherwise, it refines the arrangement $\mathcal{A}$ and goes to the next iteration.

**Irrational-Guard Polygon.** In Figure 4 we show the first 8 iterations of the Irrational-Guard polygon [2]. The orange points and faces represent point- and face-guards in the intermediate solution. The green faces represent faces not fully seen by the current candidate solution. Both orange and green faces are split in the next iteration. Note that for each of the orange and green faces, we draw a random vertex in the same colour, to make also very small faces visible.

**Local complexity.** One of the bottlenecks is the large number of possible visibilities between candidates and witnesses that we have to compute. Due to the low local complexity of the input polygons, most of those pairs are not seeing each other. We exploit this by building a so-called *weak visibility polygon tree*. In this tree, any point $p$ in node $n(p)$ can see a point $q$ in node $n(q)$, if the two nodes are siblings or in a parent-child relationship, see Figure 5.

Before, we can describe the weak visibility polygon tree, recall that the weak visibility polygon $\mathrm{vis}(s)$ (of a segment $s$) is defined by $\mathrm{vis}(s) = \{p \in P : \exists q \in s \text{ s.t. } \mathrm{seg}(p,q) \subseteq P\}$. To build the weak visibility polygon tree $T$ of $P$, we start with an arbitrary edge $e$ on the boundary of $P$. We compute the weak visibility polygon $\mathrm{vis}(e)$ of $e$, which is the root of $T$. For every edge $e'$ of $\mathrm{vis}(e)$, which is not part of the boundary of $P$, we compute the weak visibility polygon $\mathrm{vis}(e')$ with respect to the polygon $P \setminus \mathrm{vis}(e)$. Those weak visibility polygons are the children of $\mathrm{vis}(e)$. We continue recursively to compute the children of every weak visibility polygon. Note that every node of $T$ is a weak visibility polygon $W$ of some defining chord $c$.


**Figure 4** The first 8 iterations of the Irrational-Guard polygon [2].

Interestingly, this inspired a new structural parameter, which we call the chord-visibility width. Given a chord $c$ of $P$, we denote by $n(c)$ the number of vertices visible from $c$. The *chord-visibility width* ($\mathrm{cw}(P)$) of a polygon is the maximum $n(c)$ over all possible chords $c$.

We show that the art gallery problem is FPT with respect to the chord-visibility width.

▶ **Theorem 4** (Chord-Width-FPT). *Let $P$ be a simple polygon with vision-stability at least some fixed $\delta$. Then there is an FPT algorithm for the art gallery problem with respect to the chord-visibility width.*

The core idea is to use dynamic programming along the weak visibility polygon tree, similar to dynamic programming algorithms for tree-width. The challenge here is to find bounds on the number of candidates per node.

**Critical witnesses.** Another practical idea is to reduce the total number of witnesses that we use. Instead of using all faces and vertices of $\mathcal{A}$ as witnesses, we only use some random selection, which we call *critical witnesses*. We use heuristics to update this critical witness set. In deciding whether to add a critical witness or not, we are faced with some trade-offs. If we add very few critical witnesses, we have to make more loops until the IP solution returns a guard set that sees the entire polygon. If we add too many critical witnesses, the set of critical witnesses grows unnecessarily fast and we have to compute many more visibilities.

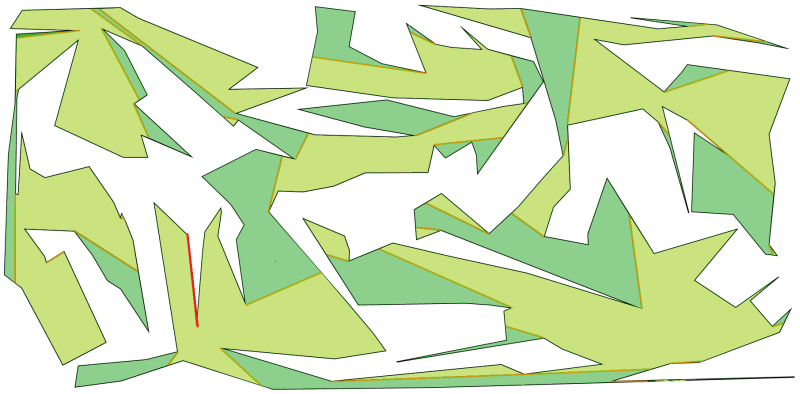

**Figure 5** The polygon together with a weak visibility polygon tree. The polygon has 200 vertices, but each node in the weak visibility polygon tree has only about 20 vertices. The red segment indicates the starting edge of the weak visibility polygon tree.

**Visibility queries.** One of the major bottlenecks at the beginning of this project was the computation of weak visibility queries. That is, we often need to decide if a given face sees a given point or another face. In a follow-up project [41], we developed a fast practical algorithm to compute weak visibility polygons.

**Losing performance guarantees.** The main reason that the iterative algorithm does not have the same performance guarantees as the one-shot algorithm is as follows. It is possible that the iterative algorithm keeps splitting a certain face (and its children) many times, only to conclude much later that it was misled. It could have found a solution much earlier by splitting one of the larger faces. In practice, it seems usually a very good idea to split the faces that were selected by the Integer program solution. Especially, if those faces are small, we made progress and avoided usually unnecessary splits of big unimportant faces. It is easily possible to design an algorithm in a way that it will not split too small faces, before also splitting occasionally bigger faces, that might be useful. In this way, we are still able to ensure theoretical performance guarantees, see Theorem 5. However, this theorem adds little to the goals of explanation, prediction, and invention. Quite the opposite, this analysis suggests that faces should be split in a way that is harmful to practical performance. In the full version [40], we describe several different versions of the iterative algorithm. When we use the safeguard version of the algorithm, we get the following theorem.

▶ **Theorem 5** (Iterative Algorithm). *Let $P$ be an $n$ vertex polygon, with vision-stability $\delta$. Then the iterative algorithm returns the optimal solution to the art gallery problem. It has a running time of $(\frac{n}{\delta})^{O(1)} + T$ per iteration and takes at most $(\frac{n}{\delta})^{O(1)}$ iterations. Here $T$ denotes the time it takes to solve one integer program.*

**Experimental results.** We implemented and tested the iterative algorithm with a 64-bit Windows 10 operating system, an 8-core Intel(R) Core i7-7700HQ CPU at 2800 Mhz and 16 GB of main memory. The practical implementation makes heavy use of version 4.13.1 of CGAL [65]. The IP solver used was IBM ILOG CPLEX version 12.10 [16].

We compared our implementation directly with the algorithm from Tozoni et al. [68], as this is the currently best algorithm, for which there was freely accessible code available. In order to get a deeper understanding, we analyzed various aspects of the running time. One

of them is the distribution of the running time w.r.t. different subroutines. Furthermore, we studied the influence of vision-stability on the running time. We also study the effect of our speed-up methods. Lastly, we study the iterative algorithm on the irrational-guard polygon [2]. For an in-depth description of our experiments with all details, we refer the reader to the full version [40]. Here, we only highlight the most important findings.

**Comparison.** We tested our algorithm and the algorithm of Tozoni et.al. on 5 sets of 30 polygons of sizes 60, 100, 200, and 500 vertices. The results can be seen in Table 1. We see that, except for size 60, our algorithm is slightly faster. However, we want to point out that comparing those running times should be taken with a grain of salt. Both algorithms rely on a software environment that is not identical. To some degree, the differences in the running time may come from performance differences from this software. First note, that Tozoni et al. implemented their algorithm in Linux, whereas, we implemented our algorithm in Windows. Interestingly, CGAL runs between factor $2-3$ faster on Linux compared to Windows [48]. Secondly, our algorithm has to compute visibilities of faces instead of point visibilities. Thirdly, while testing the implementation of Tozoni et al, we could not use the best available IP solver, which might skew the results a bit. Overall, our algorithm seems faster for larger polygons, but does not make a very large improvement.

**Table 1** A comparison of the iterative algorithm without safe guards with the results from Tozoni et al. [68], both the results reported by Tozoni et al. themselves [68] and results found using their implementation on our hardware.

| Sizes | Average time (s) | | |
|---|---|---|---|
| | **Tozoni et al.** | **Tozoni et al. (Our hardware)** | **The Iterative Algorithm** |
| 60 | 0.26 | 0.18 | 0.39 |
| 100 | 0.94 | 0.68 | 0.52 |
| 200 | 3.77 | 2.54 | 2.02 |
| 500 | 35.04 | 22.34 | 18.2 |

**CPU distribution.** We analyzed the distribution of the CPU time of the iterative algorithm. The results are shown below in a pie-chart. We see that solving integer programs is the dominating factor of the running time. This shows that to improve the running time of the algorithm, we must reduce the total number or the size of the IPs. Alternatively, we can optimize the IP solver, or perhaps experiment with different IP solvers.

It may appear strange that we spend so much energy on reducing the CPU time spent on speeding up visibility queries when the CPU usage is dominated by solving IPs. The reason is that before we implemented all of the improvements described before, the running time was dominated by weak-visibility queries.

In general, the CPU distribution needs to be regarded with a grain of salt. It seems that there is usually one subroutine that dominates the running time. Often conceptual improvements decrease the running time by several factors, which in turn makes a different subroutine appear to dominate the running time. Thus the fact that solving IPs says more about the components that we optimized rather than which parts are inherently more difficult. Thus we consider it a success that the running time is now dominated by solving IPs.

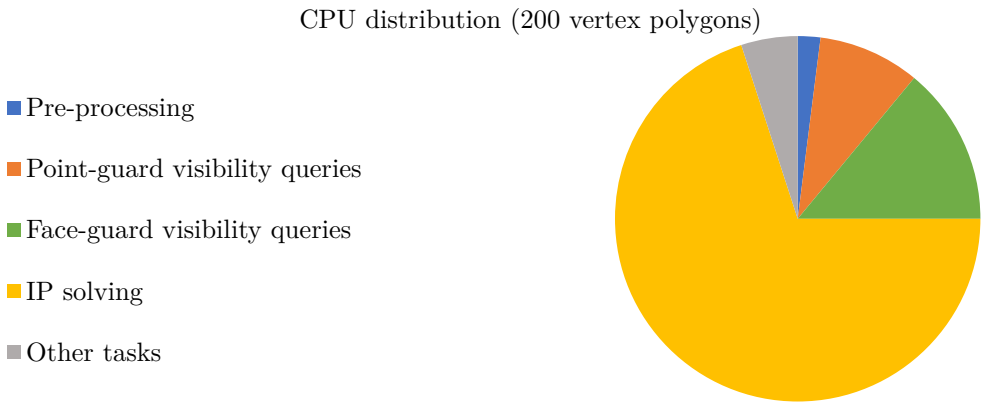


**Figure 6** The chart shows the CPU distribution of the Iterative Algorithm implementation for solving 30 polygons of size 200.

**Vision-stability.** Unfortunately, we cannot measure the vision-stability of a polygon nor approximate it. To get a vague idea of the influence of the vision-stability on the running time, we define the granularity of the subdivision at the end of the iterative algorithm. The granularity is related to the smallest face in the final subdivision. Interestingly, even with polygons of the same size the running times vary widely. The large correlations between the observed running times and the granularity indicate that vision-stability may have a significant influence on the total running time.

**Table 2** The correlation coefficients between the measured minimum granularity and the running time, computed per size.

| **Size** | 60 | 100 | 200 | 500 |
|---|---|---|---|---|
| **Correlation** | 0.07 | 0.32 | 0.21 | 0.76 |

**Weak visibility polygon tree.** To verify the amount of visibility queries that we save by using the weak visibility polygon tree, we conducted an experiment. Note that computing the weak visibility polygon tree requires the computation of weak visibility polygon. In practice, this was achieved by using an efficient new algorithm, about which a follow-up paper will be published [41]. The precise percentage highly depends on the type of polygon. See Table 3 for a detailed overview. Interestingly, the number of reflex vertices per node of the weak visibility polygon grows much slower than the input size. This indicates that chord-visibility width may be a useful practical parameter to study for geometric algorithms in polygons.

**Table 3** We tested 30 input polygons from the AGPLIB library [24] of four sizes. For each size class we see the averages of characteristics of the weak visibility polygon trees.

| **Size** | 60 | 100 | 200 | 500 |
|---|---|---|---|---|
| **Tree size** | 14.2 | 23.0 | 46.3 | 115.0 |
| **Largest polygon** | 20.5 | 23.3 | 26.2 | 28.4 |
| **Largest number of reflex vertices** | 5.9 | 6.2 | 7.0 | 9.2 |
| **Percentage of queries saved** | 16.7% | 35.4% | 63.5% | 87.3% |

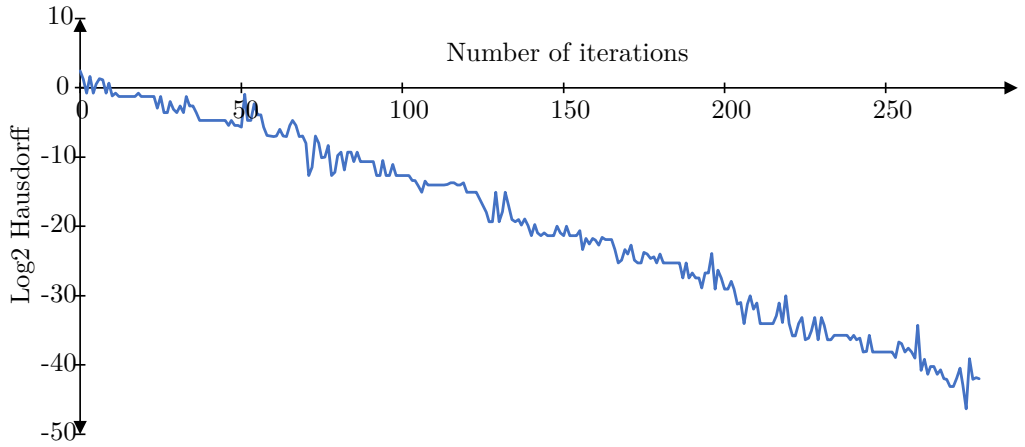


**Figure 7** The iterative algorithm based on the notion of vision-stability reports a sequence of solutions. The Graph shows on the $x$-axis the iterations from 1 to about 300 and on the $y$-axis, the $\log_2$ of the Hausdorff distance to the optimal solution.

**Irrational guards.** We show that the implementation of our algorithm provides a rapidly improving solution even for polygons that are not vision-stable. Specifically, Abrahamsen et al. [2] introduced a small and simple polygon, which requires irrational guards for an optimal guarding using point-guards. Although, the iterative algorithm avoids irrational numbers it still returns a guard set $G_i$ for each iteration $i = 1, 2, 3, \ldots$. Recall that $G_i$ consists of faces and points. As we know the optimal solution $G^*$, we can compute $d_i = d(G^*, G_i)$, see Figure 7.

**Future research.** The vast majority of the work on the art gallery problem focused on variants of the classic question. There are almost no positive theoretical algorithmic results on the original art gallery problem, with some exceptions [13, 32, 17]. We believe that the main reason for this focus on variants is the fact that the art gallery problem is inherently continuous, as is reflected by its $\exists\mathbb{R}$-completeness [3]. Now that we arguably broke that barrier, we hope that more progress will be made on the original problem.

- Can we adapt the algorithm to polygonal domains with holes? Here, the main bottleneck seems to be adapting the visibility queries to polygons with holes.
- Does the iterative algorithm always converge towards the optimal solution, even if the underlying polygon is not vision-stable? Our experimental results suggest that we converge exponentially fast to an optimal solution. It is intriguing to see if this also holds true in general.
- One simple way to make the one-shot algorithm more practical would be to find a smaller witness and candidate set. What is the smallest integer program that guarantees to give the optimal solution for vision-stable polygons? The bound we gave seems to have plenty of room for improvement.

## References

1 Zachary Abel, Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Jayson Lynch, and Tao B. Schardl. Who needs crossings? Hardness of plane graph rigidity. In *32nd International Symposium on Computational Geometry (SoCG 2016)*, pages 3:1–3:15, 2016.

2 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. Irrational guards are sometimes needed. In *SoCG 2017*, pages 3:1–3:15, 2017. `arXiv:1701.05475`.

3 Mikkel Abrahamsen, Anna Adamaszek, and Tillmann Miltzow. The art gallery problem is $\exists\mathbb{R}$-complete. In *STOC 2018*, pages 65–73, 2018. arXiv:1704.06969, doi:10.1145/3188745.3188868.

4 Mikkel Abrahamsen, Linda Kleist, and Tillmann Miltzow. Training neural networks is er-complete. *arXiv*, 2021. arXiv:2102.09798.

5 Mikkel Abrahamsen, Tillmann Miltzow, and Nadja Seiferth. A framework for $\exists\mathbb{R}$-completeness of two-dimensional packing problems. *FOCS*, 2020.

6 Akanksha Agrawal, Pradeesha Ashok, Meghana M. Reddy, Saket Saurabh, and Dolly Yadav. FPT algorithms for conflict-free coloring of graphs and chromatic terrain guarding. *Arxiv*, 1905.01822, 2019. arXiv:1905.01822.

7 Akanksha Agrawal, Kristine V. K. Knudsen, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. The Parameterized Complexity of Guarding Almost Convex Polygons. In *SoCG 2020*, LIPIcs, pages 3:1–3:16, 2020. doi:10.4230/LIPIcs.SoCG.2020.3.

8 Akanksha Agrawal, Sudeshna Kolay, and Meirav Zehavi. Parameter analysis for guarding terrains. In *SWAT*, 2020.

9 Akanksha Agrawal and Meirav Zehavi. Parameterized analysis of art gallery and terrain guarding. In *International Computer Science Symposium in Russia*, pages 16–29. Springer, 2020.

10 Yoav Amit, Joseph S.B. Mitchell, and Eli Packer. Locating guards for visibility coverage of polygons. *International Journal of Computational Geometry & Applications*, 20(05):601–630, 2010.

11 David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. *TSP in practice*, 2021. URL: http://www.math.uwaterloo.ca/tsp/index.html.

12 Pradeesha Ashok and Meghana Reddy. Efficient guarding of polygons and terrains. In *International Workshop on Frontiers in Algorithmics*, pages 26–37. Springer, 2019.

13 Patrice Belleville. Computing two-covers of simple polygons. Master's thesis, McGill University, 1991.

14 Pritam Bhattacharya, Subir Kumar Ghosh, and Sudebkumar Prasant Pal. Constant approximation algorithms for guarding simple polygons using vertex guards. *arXiv*, 2017. arXiv:1712.05492.

15 Daniel Bienstock. Some provably hard crossing number problems. *Discrete & Computational Geometry*, 6(3):443–459, 1991.

16 Robert Bixby. IBM CPLEX. URL: https://www.ibm.com/analytics/cplex-optimizer.

17 Édouard Bonnet and Tillmann Miltzow. An approximation algorithm for the art gallery problem. In *SoCG 2017*, pages 20:1–20:15, 2017. arXiv:1607.05527, doi:10.4230/LIPIcs.SoCG.2017.20.

18 Édouard Bonnet and Tillmann Miltzow. Parameterized hardness of art gallery problems. *ACM Transactions on Algorithms*, 16(4), 2020. doi:10.1145/3398684.

19 Dorit Borrmann, Pedro J. de Rezende, Cid C. de Souza, Sándor P. Fekete, Stephan Friedrichs, Alexander Kröller, Andreas Nüchter, Christiane Schmidt, and Davi C. Tozoni. Point guards and point clouds: solving general art gallery problems. In *SoCG*, pages 347–348. ACM, 2013. doi:10.1145/2462356.2462361.

20 Andrea Bottino and Aldo Laurentini. A nearly optimal sensor placement algorithm for boundary coverage. *Pattern Recognition*, 41(11):3343–3355, 2008.

21 Andrea Bottino and Aldo Laurentini. A nearly optimal algorithm for covering the interior of an art gallery. *Pattern Recognition*, 44(5):1048–1056, 2011.

22 Jean Cardinal, Stefan Felsner, Tillmann Miltzow, Casey Tompkins, and Birgit Vogtenhuber. Intersection graphs of rays and grounded segments. *Journal of Graph Algorithms and Applications*, 22:273–295, 2018.

23 Jean Cardinal and Udo Hoffmann. Recognition and complexity of point visibility graphs. *Discrete & Computational Geometry*, 57(1):164–178, 2017.

**24** Marcelo C. Couto, Pedro J. de Rezende, and Cid C. de Souza. Instances for the Art Gallery Problem, 2009. URL: `www.ic.unicamp.br/~cid/Problem-instances/Art-Gallery`.

**25** Marcelo C. Couto, Pedro J. de Rezende, and Cid C. de Souza. An exact algorithm for minimizing vertex guards on art galleries. *International Transactions in Operational Research*, 18(4):425–448, 2011.

**26** Marcelo C. Couto, Cid C. de Souza, and Pedro J. de Rezende. Experimental evaluation of an exact algorithm for the orthogonal art gallery problem. In *International Workshop on Experimental and Efficient Algorithms*, pages 101–113. Springer, 2008.

**27** Pedro J. de Rezende, Cid C. de Souza, Stephan Friedrichs, Michael Hemmer, Alexander Kröller, and Davi C. Tozoni. Engineering art galleries. *Algorithm Engineering*, pages 379–417, 2016. `doi:10.1007/978-3-319-49487-6_12`.

**28** Argyrios Deligkas, John Fearnley, and Themistoklis Melissourgos. Square-cut pizza sharing is ppa-complete. *arXiv preprint*, 2020. `arXiv:2012.14236`.

**29** Michael G. Dobbins, Linda Kleist, Tillmann Miltzow, and Paweł Rzążewski. $\forall\exists\mathbb{R}$-completeness and area-universality. *WG 2018*, 2018. `arXiv:1712.05142`.

**30** Michael Gene Dobbins, Andreas Holmsen, and Tillmann Miltzow. Smoothed analysis of the art gallery problem. *arXiv*, 2018. `arXiv:1811.01177`.

**31** Michael Gene Dobbins, Andreas Holmsen, and Tillmann Miltzow. A universality theorem for nested polytopes. *arXiv*, 2019. `arXiv:1908.02213`.

**32** Alon Efrat and Sariel Har-Peled. Guarding galleries and terrains. *Inf. Process. Lett.*, 100(6):238–245, 2006. `doi:10.1016/j.ipl.2006.05.014`.

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

**34** Jeff Erickson. Optimal curve straightening is $\exists\mathbb{R}$-complete. *arXiv*, 2019. `arXiv:1908.09400`.

**35** Jeff Erickson, Ivor van der Hoog, and Tillmann Miltzow. Smoothing the gap between NP and $\exists\mathbb{R}$. *accepted to FOCS 2020*, 2020. `arXiv:1912.02278`.

**36** S. Friedrichs. Integer solutions for the art gallery problem using linear programming. Masterthesis, 2012.

**37** Jugal Garg, Ruta Mehta, Vijay V. Vazirani, and Sadra Yazdanbod. ETR-completeness for decision versions of multi-player (symmetric) Nash equilibria. In *ICALP 2015*, pages 554–566, 2015.

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

**39** Panos Giannopoulos. Open problems: guarding problems, 2016.

**40** Simon Hengeveld and Tillmann Miltzow. A practical algorithm with performance guarantees for the art˜ gallery problem. *arXiv*, 2020. `arXiv:2007.06920`.

**41** Simon Hengeveld, Tillmann Miltzow, and Frank Staals. Weak visibility by convex expansion. in preparation 2020.

**42** Ross J. Kang and Tobias Müller. Sphere and dot product representations of graphs. In *SoCG*, pages 308–314. ACM, 2011.

**43** Farnoosh Khodakarami, Farzad Didehvar, and Ali Mohades. A fixed-parameter algorithm for guarding 1.5 d terrains. *Theoretical Computer Science*, 595:130–142, 2015.

**44** Farnoosh Khodakarami, Farzad Didehvar, and Ali Mohades. 1.5 d terrain guarding problem parameterized by guard range. *Theoretical Computer Science*, 661:65–69, 2017.

**45** David G. Kirkpatrick. An $O(\lg\lg \mathrm{OPT})$-approximation algorithm for multi-guarding galleries. *Discrete & Computational Geometry*, 53(2):327–343, 2015. `doi:10.1007/s00454-014-9656-8`.

**46** Sándor Kisfaludi-Bak, Jesper Nederlof, and Karol Węgrzycki. A gap-eth-tight approximation scheme for euclidean tsp. *arXiv preprint*, 2020. `arXiv:2011.03778`.

**47** Linda Kleist. *Planar graphs and faces areas – Area-Universality*. PhD thesis, Technische Universität Berlin, 2018. PhD thesis.

**48** Bernhard Kornberger. Why is cgal significantly slower under windows? URL: `https://stackoverflow.com/questions/58008543/`.

**49** Alexander Kröller, Tobias Baumgartner, Sándor P. Fekete, and Christiane Schmidt. Exact solutions and bounds for general art gallery problems. *Journal of Experimental Algorithmics (JEA)*, 17:2–3, 2012.

**50** Der-Tsai Lee and Arthur K. Lin. Computational complexity of art gallery problems. *IEEE Transactions on Information Theory*, 32(2):276–282, 1986. doi:10.1109/TIT.1986.1057165.

**51** Anna Lubiw, Tillmann Miltzow, and Debajyoti Mondal. The complexity of drawing a graph in a polygonal region. *Arxiv*, 2018. Graph Drawing 2018.

**52** Colin McDiarmid and Tobias Müller. Integer realizations of disk and segment graphs. *Journal of Combinatorial Theory, Series B*, 103(1):114–143, 2013.

**53** Nicolai E Mnëv. The universality theorems on the classification problem of configuration varieties and convex polytopes varieties. In Oleg Y. Viro, editor, *Topology and geometry – Rohlin seminar*, pages 527–543. Springer-Verlag Berlin Heidelberg, 1988.

**54** Rajeev Motwani, Arvind Raghunathan, and Huzur Saran. Covering orthogonal polygons with star polygons: The perfect graph approach. *J. Comput. Syst. Sci.*, 40(1):19–48, 1990. doi:10.1016/0022-0000(90)90017-F.

**55** Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

**56** Jürgen Richter-Gebert and Günter M. Ziegler. Realization spaces of 4-polytopes are universal. *Bulletin of the American Mathematical Society*, 32(4):403–412, 1995.

**57** Tim Roughgarden. Beyond the worst-case analysis of algorithms (introduction). *CoRR*, abs/2007.13241, 2020. arXiv:2007.13241.

**58** Marcus Schaefer. Complexity of some geometric and topological problems. In *Proceedings of the 17th International Symposium on Graph Drawing (GD 2009)*, volume 5849 of *Lecture Notes in Computer Science (LNCS)*, pages 334–344. Springer, 2009.

**59** Marcus Schaefer. Realizability of graphs and linkages. In *Thirty Essays on Geometric Graph Theory*, pages 461–482. Springer, 2013.

**60** Marcus Schaefer and Daniel Štefankovič. Fixed points, Nash equilibria, and the existential theory of the reals. *Theory of Computing Systems*, 60(2):172–193, 2017. doi:10.1007/s00224-015-9662-0.

**61** Dietmar Schuchardt and Hans-Dietrich Hecker. Two NP-hard art-gallery problems for ortho-polygons. *Math. Log. Q.*, 41:261–267, 1995. doi:10.1002/malq.19950410212.

**62** Yaroslav Shitov. A universality theorem for nonnegative matrix factorizations. *arXiv*, 2016. arXiv:1606.09068.

**63** Yaroslav Shitov. The complexity of positive semidefinite matrix factorization. *SIAM Journal on Optimization*, 27(3):1898–1909, 2017.

**64** Peter Shor. Stretchability of pseudolines is np-hard. *Applied Geometry and Discrete Mathematics-The Victor Klee Festschrift*, 1991.

**65** The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.1.3 edition, 2020. URL: https://doc.cgal.org/4.1.3/Manual/packages.html.

**66** Davi C. Tozoni, Pedro J. de Rezende, and Cid C de Souza. A practical iterative algorithm for the art gallery problem using integer linear programming. *Optimization Online*, 2013.

**67** Davi C. Tozoni, Pedro J. de Rezende, and Cid C. de Souza. The quest for optimal solutions for the art gallery problem: A practical iterative algorithm. In *Experimental Algorithms*, pages 320–336, 2013.

**68** Davi C. Tozoni, Pedro J. de Rezende, and Cid C. de Souza. Algorithm 966: A practical iterative algorithm for the art gallery problem using integer linear programming. *ACM Trans. Math. Softw.*, 43(2), 2016. doi:10.1145/2890491.