On conflict-free chromatic guarding of simple polygons

Onur Çağırıcı
Masaryk University, Brno, Czech Republic
onur@mail.muni.cz

Subir Kumar Ghosh
Ramakrishna Mission Vivekananda University, Kolkata, India
ghosh@tifr.res.in

Petr Hliněný
Masaryk University, Brno, Czech Republic
hlineny@fi.muni.cz

Bodhayan Roy
Masaryk University, Brno, Czech Republic
b.roy@fi.muni.cz

Abstract

We study the problem of colouring the vertices of a polygon, such that every viewer can see a unique colour. The goal is to minimize the number of colours used. This is also known as the conflict-free chromatic guarding problem with vertex guards (which is quite different from point guards considered in other papers). We study the problem in two scenarios of a set of viewers.

In the first scenario, we assume that the viewers are all points of the polygon. We solve the related problem of minimizing the number of guards and approximate (up to only an additive error) the number of colours in the special case of funnels. We also give an upper bound of \(O(\log n)\) colours on weak-visibility polygons which generalizes to all simple polygons.

In the second scenario, we assume that the viewers are only the vertices of the polygon. We show a lower bound of 3 colours in the general case of simple polygons and conjecture that this is tight. We also prove that already deciding whether 1 or 2 colours are enough is \(\text{NP}\)-complete.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Graph coloring

Keywords and phrases Computational geometry; Polygon guarding; Visibility graph; Art gallery problem; Conflict-free coloring

Funding Onur Çağırıcı: Supported by the Czech Science Foundation, project no. 17-00837S.
Petr Hliněný: Supported by the Czech Science Foundation, project no. 17-00837S.

1 Introduction

The art gallery problem is one of the best known problems in computational geometry [14][27]. The problem is to find the minimum number of guards to cover an \(n\)-vertex polygon. Originally, Victor Klee asked the question: how many point guards need to be placed in the polygon so that each point in the polygon is seen by at least one of the guards? The Art Gallery Theorem, proved by Chvátal, shows that \([n/3]\) guards are sufficient and sometimes necessary [13]. The guard minimization problem was shown to be \(\text{NP}\)-hard by Lee and Lin [26] and more recently \(\exists\text{R}\)-complete by Abrahamsen et al. [2]. The minimization problem has been studied under many constraints such as the placement of guards being restricted to the polygonal perimeter or vertices [24], the viewers being restricted to vertices, the polygon being terrains [3][6][15], weakly visible from an edge [7], with holes or orthogonal [9][16][23], with respect to parameterization [10], approximability [22].
For most of these cases the problem remains hard, but interesting approximation algorithms have also been provided \cite{11,19}. The variants of the art gallery problems have widespread uses in real life problems, such as placement of security cameras, robot motion planning etc \cite{18,27}. We are interested in one particular class of art gallery problems. Consider the problem of navigating a robot inside a polygon. The robot communicates with wireless sensors of different frequencies. At any position of the robot, it must be able to communicate with a wireless sensor of a unique frequency, to prevent interference. In our settings, a guard corresponds to a wireless sensor, and the frequency of the sensor corresponds to a colour assigned to the guard. Then, the problem is that of placing coloured guards in a polygon, such that any point in the polygon can see a guard of a unique colour (among all the visible guards). This is called the (weak) conflict-free chromatic guarding problem \cite{5}. There is another variant in which all guards seen by any point must be of unique colours, called the strong conflict-free chromatic guarding problem \cite{17}. We, however, study exclusively the former weak variant.

The point-to-point (P2P) conflict-free chromatic guarding (art gallery) problem has been studied in several papers. Bärtschi and Suri gave an upper bound of $O(\log^2 n)$ colours on simple polygons \cite{5}. Later, Bärtschi et al. improved this upper bound to $O(\log n)$ on simple polygons \cite{4}, and Hoffmann et al. \cite{21}, while studying the orthogonal variant of the problem, have given the first nontrivial lower bound of $\Omega(\log \log n / \log \log \log n)$ colours holding also in the general case of simple polygons.

In this paper, we consider restricted guard (and viewer) positions. Namely, we consider vertex-to-point (V2P) guarding where the guards should be placed on polygon vertices and viewers can be any points of the polygon, and vertex-to-vertex (V2V) guarding where both the guards and viewers are restricted to the vertices of the polygon. Note that there are some fundamental differences between point and vertex guards, e.g., funnel polygons (Fig. 1) can always be guarded by one point guard (of one colour) but they may require up to $\Omega(\log n)$ colours in the V2P conflict-free chromatic guarding, as shown in \cite{4}. Here we, in particular, extend the upper bound of $O(\log n)$ colours for point guards on simple polygons by Bärtschi et al. \cite{4} to asymptotically the same bound for more restrictive vertex guards.

We also remark that, specially, the V2V conflict-free chromatic guarding problem (in the formulation from \cite{1}) coincides with the conflict-free colouring problem \cite{1,28} on the visibility graph of the considered polygon. For general graphs, it is known that conflict-free colouring can require arbitrary number of colours, and the problem is NP-complete \cite{1}. If a graph can be conflict-free coloured by only one colour, then the graph is said to have a perfect code, which is a well-studied topic in its own right \cite{6,25}. However, these results on general graphs, as well as the P2P lower bound from \cite{21} which requires viewers inside the polygon, do not seem to imply anything useful for our V2V guarding case.

**Basic definitions** A polygon $P$ is defined as a closed region $R$ in the plane bounded by a finite set of line segments (called edges of $P$) \cite{18}. We consider simple polygons, i.e., simply connected regions (informally, “without holes”). Two points $p_1$ and $p_2$ of a polygon $P$ are said to see each other, or be visible to each other, if the line segment $p_1p_2$ fully belongs to $P$. In this context, we say that a guard $g$ guards a point $x$ of $P$ if the line segment $gx$ fully belongs to $P$. A polygon $P$ is a weak visibility polygon if $P$ has an edge $uv$ such that for every point $p$ of $P$ there is a point $p'$ on $uv$ seeing $p$.

A solution of conflict-free chromatic guarding of a polygon $P$ consists of a set of guards (a subset of the vertex set of $P$ in our case) and an assignment of colours to the guards (one colour per guard) such that the following holds; every viewer $v$ in $P$ can see a guard of colour
such that no other guard seen by \( v \) has the same colour \( c \). In the V2P variant the viewers are all points of \( P \), while in V2V the viewers are picked only among the vertices of \( P \). The goal is to minimize the number of colours used.

**Our results**

1. (Sections 2 and 3 in Theorems 2.3, 3.5 and 3.6) We give an algorithm to find the optimum number \( m \) of vertex-guards to guard all the points of a funnel, and show that the number of colours in the corresponding conflict-free chromatic guarding problem is \( \log m + \Theta(1) \). This leads to an approximation algorithm for V2P conflict-free chromatic guarding of a funnel, with only a constant additive error.

2. (Section 4 in Theorems 4.3 and 4.5) We show that a weak visibility polygon on \( n \) vertices can be V2P conflict-free chromatic guarded with only \( O(\log n) \) guards. We generalize this upper bound to all simple polygons, which is a substantial (and generally best possible) improvement over [4].

3. (Section 5 in Theorems 5.1 and 5.3) We construct a polygon that cannot be V2V chromatic guarded with only two colours, and conjecture that three colours always suffice (unlike in the V2P and P2P settings for which this is unbounded [4,21]). We prove that determining whether a given polygon can be V2V guarded by a conflict-free coloured guard set using only one or two colours is NP-complete.

### 2 Minimizing vertex-to-point guards for funnels

In the next two sections, we focus on the V2P variant of guarding on a special interesting type of polygons – funnels. A polygon \( P \) is a funnel if precisely three of the vertices of \( P \) are convex, and two of the convex vertices share one common edge – the base of the funnel \( P \).

Before turning to the conflict-free chromatic guarding problem, we first resolve the problem of minimizing the total number of vertex guards needed to guard all points of a funnel. The efficient solution we provide here will be helpful for the subsequent results. Actually, we give a couple of algorithms to obtain a guard set for a given funnel. We start by describing a simple procedure (Algorithm 1) that provides us with a guard set which may not always be optimal (but not far from the optimum, see Lemma 2.4). Then, we refine this simple procedure in order to obtain the optimal number of guards in Algorithm 2.

We use some special notation here. See Figure 1. Let the given funnel be \( F \), oriented in the plane as follows. On the bottom, there is the horizontal base of the funnel – the line segment \( l_1 r_1 \) in the picture. The topmost vertex of \( F \) is called the apex, and it is denoted by \( \alpha \). There always exists a point \( x \) on the base which can see the apex \( \alpha \), and then \( x \) sees the whole funnel at once. The vertices on the left side of apex form the left concave chain, and analogously, the vertices on the right side of the apex form the right concave chain of the funnel. These left and right concave chains are denoted by \( \mathcal{L} \) and \( \mathcal{R} \) respectively. We denote the vertices of \( \mathcal{L} \) as \( l_1, l_2, \ldots, l_k \) from bottom to top. We denote the vertices of \( \mathcal{R} \) as \( r_1, r_2, \ldots, r_m \) from bottom to top. Hence, the apex is \( l_k = r_m = \alpha \).

Let \( l_i \) be a vertex on \( \mathcal{L} \) which is not the apex. We define the upper tangent of \( l_i \), denoted by \( \operatorname{upt}(l_i) \), as the ray whose origin is \( l_i \) and which passes through \( l_{i+1} \). Upper tangents for vertices on \( \mathcal{R} \) are defined analogously. Let \( p \) be the point of intersection of \( \mathcal{R} \) and the upper tangent of \( l_i \). Then we define \( \operatorname{ups}(l_i) \) as the line segment \( l_{i+1}p \). For the vertices of \( \mathcal{R} \), \( \operatorname{ups} \) is defined analogously: if \( q \) is the point of intersection of \( \mathcal{L} \) and the upper tangent of \( r_j \in \mathcal{R} \), then let \( \operatorname{ups}(r_j) := \overline{r_jq} \). See again Figure 1.
Guarding of simple polygons

Algorithm 1: Simple vertex-to-point guarding of funnels (uncoloured).

**Input:** A funnel $F$ with concave chains $L = (l_1, l_2, \ldots, l_k)$ and $R = (r_1, r_2, \ldots, r_m)$.

**Output:** A vertex set guarding all the points of $F$.

1. Initialize an auxiliary digraph $G$ with two dummy vertices $x$ and $y$, and declare $ups(x) = l_1 r_1$.
2. Initialize $S \leftarrow \{x\}$.
3. **while** $S$ is not empty **do**
   4. Choose an arbitrary $t \in S$, and remove $t$ from $S$;
   5. Let $s = ups(t)$; /* $s$ is a segment inside $F$ */
   6. Let $i$ and $j$ be the largest indices such that $l_i$ and $r_j$ are not above
      the intersection of the segment $s$ with $L$ and $R$, respectively;
   7. if $l_{i+1}$ can see whole $s$ then $i' \leftarrow i + 1$;
   8. else $i' \leftarrow i$; /* the topmost vertex on the left seeing whole $s$ */
   9. if $r_{j+1}$ can see whole $s$ then $j' \leftarrow j + 1$;
  10. else $j' \leftarrow j$; /* the topmost vertex on the right seeing whole $s$ */
  11. Include the vertices $l_{i'}$ and $r_{j'}$ in $G$;
  12. **foreach** $z \in \{l_{i'}, r_{j'}\}$ **do**
  13. Add the directed edge $(t, z)$ to $G$;
  14. if segment $ups(z)$ includes the apex $l_k = r_m$ then
  15. Add the directed edge $(z, y)$ to $G$; /* $y$ is final dummy vertex */
  16. else $S \leftarrow S \cup \{z\}$; /* more guards are needed above $z$ */
  17. Enumerate a shortest path from $x$ to $y$ in $G$;
  18. Output the shortest path vertices without $x$ and $y$ as the required guard set;
Figure 2 A symmetric funnel with 17 vertices. The gray dashed lines show the upper tangents of the vertices. It is easy to see that Algorithm 1 selects 4 guards, up to symmetry, at \( l_2, r_5, l_7, l_9 \) (the red vertices). However, the whole funnel can be guarded by three guards at \( l_4, r_4, l_8 \) (the green vertices), and it will be the task of Algorithm 2 to consider such better possibility.

The underlying idea of Algorithm 1 is simple. Imagine we proceed bottom-up when building the guard set of a funnel \( F \). Then the next guard is placed at the top-most vertex \( z \) of \( F \), nondeterministically choosing between \( z \) on the left and the right chain of \( F \), such that no “unguarded gap” remains below \( z \). Note that the unguarded region of \( F \) after placing a guard at \( z \) is bounded from below by \( \ups(z) \). The nondeterministic choice of the next guard \( z \) is encoded within a digraph, in which we then find the desired guard set as a shortest path.

Lemma 2.1. Algorithm 1 runs in polynomial time, and it outputs a feasible guard set for all the points of a funnel \( F \).

Proof. As for the runtime, we observe that the number of considered line segments \( s \) in the algorithm is, by the definition of \( \ups \), bounded by at most \( k + m \) (and it is typically much lower than this bound). Each considered segment \( \ups(t) \) of \( t \in \mathcal{S} \) is processed at most once, and it contributes two edges to \( G \). Overall, a shortest path in \( G \) is found in linear time.

We prove feasibility of the output set by induction. Let \( (x = x_0, x_1, \ldots, x_{n-1}, x_n = y) \) be a path in \( G \). We claim that, for \( 0 \leq i \leq a \), guards placed at \( x_0, x_1, \ldots, x_i \) guard all the points of \( F \) below \( \ups(x_i) \). This is trivial for \( i = 0 \), and it straightforwardly follows by induction: Algorithm 1 asserts that \( x_i \) can see whole \( \ups(x_{i-1}) \), and \( x_i \) hence also sees the strip between \( \ups(x_{i-1}) \) and \( \ups(x_i) \) by basic properties of a funnel. Finally, at \( x_{n-1} \), we guard whole \( F \) up to its apex.

Remark 2.2. Unfortunately, the guard set produced by Algorithm 1 may not be optimal under certain circumstances. See the example in Figure 2, the algorithm picks the four red vertices, but the funnel can be guarded by the three green vertices.

We now refine the simple approach of Algorithm 1 to always produce a minimum size guard set. Recall that Algorithm 1 always places one next guard based on the position of the previous one guard. Our refinement is going to consider also pairs of guards (one from the left and one from the right chain) in the procedure. For that we extend the definition of
Guarding of simple polygons

Algorithm 2: Optimum vertex-to-point guarding of funnels (uncoloured).

**Input:** A funnel $F$ with concave chains $\mathcal{L} = (l_1, l_2, \ldots, l_k)$ and $\mathcal{R} = (r_1, r_2, \ldots, r_m)$.

**Output:** A minimum vertex set guarding all the points of $F$.

1. Initialize an auxiliary digraph $G$ with two dummy vertices $x$ and $y$, and declare $ups(x) = l_1 r_1$.
2. Initialize $S \leftarrow \{x\}$.
3. while $S$ is not empty do
   4. Choose an arbitrary $t \in S$, and remove $t$ from $S$;
   5. Let $s = ups(t)$; /* $s$ is a segment or a $\vee$-shape */
   6. Let $i'$ and $j'$ be defined for $s$ as in Algorithm 1.
   7. Let $q$ and $p$ be the ends of $s$ on $\mathcal{L}$ and $\mathcal{R}$, respectively;
   8. Let $i''$ and $j''$ be the largest indices such that $l_{i''}$ lies strictly below $ups(p)$ and $r_{j''}$ strictly below $ups(q)$;
   9. Include the vertices $l_{i''}$, $r_{j''}$ and $(l_{i''}, r_{j''})$ in $G$;
   10. foreach $z \in \{(l_{i''}, r_{j''}), (l_{i''}, r_{j''})\}$ do
       11. Add the directed edge $(t, z)$ to $G$, and
           assign $(t, z)$ weight 2 if $z = (l_{i''}, r_{j''})$, and weight 1 otherwise;
       12. if $ups(z)$ includes the apex $l_k = r_m$ then
           13. Add the directed edge $(z, y)$ to $G$ of weight 0;
   14. else $S \leftarrow S \cup \{z\}$; /* more guards are needed above $z$ */
15. Enumerate a shortest weighted path from $x$ to $y$ in $G$;
16. Output the shortest path vertices without $x$ and $y$, but considering the possible guard pairs, as the required guard set;

$ups$ to pairs of vertices as follows. Let $l_i$ and $r_j$ be vertices of $F$ on $\mathcal{L}$ and $\mathcal{R}$, respectively, such that $ups(l_i) = l_{i+1} p$ intersects $ups(r_j) = r_{j+1} q$ in a point $t$ (see in Figure 1). Then we set $ups(l_i, r_j)$ as the polygonal line ("$\vee$-shape") $\overline{pt} \cup \overline{qt}$. In case that $ups(l_i) \cap ups(r_j) = \emptyset$, we simply define $ups(l_i, r_j)$ as the upper one of $ups(l_i)$ and $ups(r_j)$.

Algorithm 2, informally saying, enriches the two nondeterministic choices of placing the next guard in Algorithm 1 with a third choice: placing a suitable top-most pair of guards $(z_1, z_2)$, $z_1 \in \mathcal{L}$ and $z_2 \in \mathcal{R}$, such that again no “unguarded gap” remains below $(z_1, z_2)$. Figure 2 features a funnel in which placing such a pair of guards $(z_1 = l_4, z_2 = r_4)$ may be strictly better than using any two consecutive steps of Algorithm 1.

Within the scope of Algorithm 2 (cf. line 5), we extend the definition range of $ups(\cdot)$ to include all boundary points of $\mathcal{L}$ and $\mathcal{R}$, as follows. If $p$ is an internal point of $l_{t_{i+1}}$, then we set $ups(p) := ups(l_i)$.

If $p'$ is an internal point of $r_{j_{j_{j+1}}}$, then we set $ups(p') := ups(r_j)$.

$\triangleright$ **Theorem 2.3.** Algorithm 2 runs in polynomial time, and it outputs a feasible guard set of minimum size guarding all the points of a funnel $F$.

**Proof.** Proving polynomial runtime is analogous to Lemma 2.1 only now obtaining a quadratic worst-scenario bound. Likewise the proof of feasibility of the obtained solution is analogous to the previous proof. We only need to observe the following new claim: if $i''$ and $j''$ are defined as on line 8 of Algorithm 2, then $l_{i''}$ and $r_{j''}$ together can see whole $s$ and the strip of $F$ from $s$ till $ups(l_{i''}, r_{j''})$. The crucial part is to prove optimality.

Having two guard sets $A, B \subseteq V(F)$, we say that $A$ covers $B$ if there is an injection

---

6.

---

---
Algorithm 1 can “duplicate” the move, hence making with the source which contradicts maximality of the path w.r.t. the cover relation. Let highest guard from as high on lies back on that x symmetry, assume path P shorter path refined statement by induction on Let

Proof. Lemma 2.4. ▶ use also simpler Algorithm 1 in subsequent applications.

Let A be any feasible guard set of F which covers given D, and such that A is maximal w.r.t. the cover relation. Let P be a maximal directed path in G, starting from x, such that the set of guards B_P listed in the vertices of P (without x) satisfies B_P ⊆ A. Obviously, we aim to show that P ends in y. Suppose not (it may even be that P is a single vertex x and B_P = ∅). Let t be the last vertex of P and denote by s = ups(t) and let q and p be the ends of s on L and R, respectively, as in the algorithm.

Let A’ = A \ B_P ≠ ∅. Then s has to be guarded from A’ (while the whole part of F below is already guarded by B_P by feasibility of the algorithm). Let i, j be such that l_i ∈ A’ ∩ L and r_j ∈ A’ ∩ R are the lowest guards on the left and right chain. Assume, up to symmetry, that l_i can see whole s. By our maximal choice of A we have that no vertex on L above l_i can see whole s, and so the digraph G contains an edge from t to l_i (line 11 of Algorithm 2), which contradicts maximality of the path P.

Otherwise, neither of l_i, r_j can see whole s, and so l_i sees the end p and r_j sees the end q. Consequently, l_i is strictly below ups(p) and r_j strictly below ups(q), and they are topmost such vertices again by our maximal choice of A. Hence the digraph G contains an edge from t to (l_i, r_j), as previously, which is again a contradiction concluding the proof. ▶

Lastly, we establish a relationship between Algorithms 1 and 2, because we would like to use also simpler Algorithm 1 in subsequent applications.

Lemma 2.4. The guard set produced by Algorithm 1 is by at most one guard larger than the optimum solution produced by Algorithm 2.

Proof. Let G^1 with the source x^1 be the auxiliary graph produced by Algorithm 1, and G^2 with the source x^2 be the one produced by Algorithm 2. We instead prove the following refinement by induction on i ≥ 0:

Let P^2 = (x^2 = x^2_0, x^2_1, ..., x^2_r) be any directed path in G^2 of weight k, let Q^2 denote the set of guards listed in the vertices of P^2, and L^2 = L ∩ Q^2 and R^2 = R ∩ Q^2. Then there exists a directed path (x^1 = x^1_0, x^1_1, ..., x^1_k, x^1_{k+1}) in G^1 (of length k + 1), such that the guard of x_k is at least as high as all the guards of L^2 (if x_k ∈ L) or of R^2 (if x_k ∈ R), and the guard of x_{k+1} is strictly higher than all the guards of Q^2.

The claim is trivial for i = 0, and so we assume that i ≥ 1 and the claim holds for the shorter path P^2' = (x^2' = x^2'_0, x^2'_1, ..., x^2'_{r-1}) of weight k' in G^2, hence providing us with a path (x^1 = x^1_0, x^1_1, ..., x^1_k, x^1_{k+1}) in G^1. If k = k' + 1 (i.e., x^2_k represents a single guard), Algorithm 1 can “duplicate” the move, hence making x^2_k or a higher vertex x^2_{k+1} on the same chain an outneighbour of x^1_{k+1} in G^1. Then we set x^1_{k+2} = x^1_{k+1} = x^2_k = x^1_k and we are done.

If k = k' + 2 (i.e., x^2_k represents a pair of guards z_1, z_2), we proceed as follows. Up to symmetry, assume x^1_{k' + 1} ∈ L and z_1 ∈ L, z_2 ∈ R. By the induction assumption, we know that x^1_{k+2} is strictly higher (on L) than the guards from L^2 \ {z_1}. We choose x^1_k as the outneighbour of x^1_{k' + 1} in G^1 that lies on R, and x^1_{k+1} as the outneighbour of x^1_k in G^1 that lies back on L. From Algorithm 2 (line 8) it follows that z_2 sees x^1_{k' + 1}, and so x^1_k is at least as high on R as z_2. Consequently, x^1_{k+1} lies on L strictly higher than z_1 (which sees the highest guard from R^2 \ {z_2}), and we are again done. ▶
Figure 3 An example of a 2-interval $Q$ of a funnel (filled green and bounded by $s_1 = \text{ups}(a_1)$ and $s_2 = \text{los}(b)$). The red vertices $a_1, a_2, a_3, a_4$ are the guards computed by Algorithm 1, and $a_2, a_3$ belong to the interval $Q$. Note that $a_1$ and $b$ by definition do not belong to $Q$. The shadow of $Q$ (filled light gray) is bounded from below by the bottom dotted line, and the inner point $o$ is the so-called observer of $Q$ (seeing all vertices of $Q$ and, possibly, some vertices in the shadow).

3 Vertex-to-point conflict-free chromatic guarding of funnels

In this section, we continue to study vertex-to-point guarding of funnels, now in the context of conflict-free chromatic guarding. To obtain a conflict-free coloured solution, we will simply consider the guards chosen by Algorithm 1 in the ascending order of their vertical coordinates, and colour them in the logarithmic sequence, (also known as the ruler sequence [20]) defined as the sequence where the $i$th term is the exponent of the largest power of 2 that divides $2^i$. The first few terms of the logarithmic sequence are: $1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1, 5, 1, 2, 1, 3, 1, \ldots$.

So, if Algorithm 1 gives $m$ guards, then our approach will use about $\log m$ colours.

Our aim is to show that this is always very close to the optimum, by giving a lower bound on the number of necessary colours of order $\log m - O(1)$. To achieve this lower bound, we will study the following two sets of guards for a given funnel $F$:

- The minimal guard set $A$ computed by Algorithm 1 on $F$ (which is overall nearly optimal by Lemma 2.4); if this is not unique, then we fix any such $A$.

- A guard set $D$ which achieves the minimum number of colours for conflict-free guarding; note that $D$ may be much larger than $A$ since it is the number of colours which matters.

On a high level, our ultimate goal is to show that the colouring of $D$ must (somehow) copy the logarithmic sequence on $A$, thus giving the desired lower bound. For that we will recursively bisect our funnel into smaller “layers”, gaining one unique colour with each bisection.

Analogously to the notion of an upper tangent from Section 2, we define the lower tangent of a vertex $l_i \in \mathcal{L}$, denote by $\text{lot}(l_i)$, as the ray whose origin is $l_i$ and which passes through $r_j \in \mathcal{R}$ such that $r_j$ is the lowest vertex on $\mathcal{R}$ seeing $l_i$. Note that $\text{lot}(l_i)$ may intersect $\mathcal{R}$ in $r_j$ alone or in a segment from $r_j$ up. Let $\text{los}(l_i) := l_i r_j$. The definition of $\text{lot}()$ and $\text{los}()$ for vertices of $\mathcal{R}$ is symmetric.

We now give a definition of “layers” of a funnel which is crucial for our proof.

Definition 3.1 $k$-interval. Let $F$ be a funnel with concave chains $\mathcal{L} = (l_1, l_2, \ldots, l_k)$ and $\mathcal{R} = (r_1, r_2, \ldots, r_m)$, and $A$ be the fixed guard set $A$ computed by Algorithm 1 on $F$. Let
\( s_1 \) be the base of \( F \), or \( s_1 = \text{ups}(p) \) for some vertex \( p \) of \( F \) (where \( p \) is not the apex or its neighbour). Let \( s_2 \) be the apex of \( F \), or \( s_2 = \text{los}(q) \) for some vertex \( q \) of \( F \) (where \( q \) is not in the base of \( F \)). Assume that \( s_2 \) is above \( s_1 \) within \( F \). Then the region \( Q \) of \( F \) bounded from below by \( s_1 \) and from above by \( s_2 \), excluding \( q \) itself, is called an interval of \( F \). Moreover, \( Q \) is called a \( k \)-interval of \( F \) if \( Q \) contains at least \( k \) of the guards of \( A \). See Figure 3.

Having an interval \( Q \) of the funnel \( F \), bounded from below by \( s_1 \), we define the shadow of \( Q \) as follows. If \( s_1 = \text{ups}(i_1) \) (\( s_1 = \text{ups}(r_j) \)), then the shadow consists of the region of \( F \) between \( s_1 \) and \( \text{los}(l_{i+1}) \) (between \( s_1 \) and \( \text{los}(r_{j+1}) \), respectively). If \( s_1 \) is the base, then the shadow is empty.

Lemma 3.2. If \( Q \) is a 13-interval of the funnel \( F \), then there exists a point in \( Q \) which is not visible from any vertex of \( F \) outside of \( Q \).

Proof. By definition, a 13-interval has thirteen guards from \( A \) in it. By the pigeon-hole principle, at least seven of these guards are on the same chain. Without the loss of generality, let these seven guards lie on the left chain \( L \) of \( F \). Let us denote these guards by \( a \), \( b \), \( c \), \( d \), \( e \), \( f \) and \( g \) in the bottom-up order, respectively. We show that the guard \( d \) is not seen by any viewer outside of the 13-interval.

Suppose that \( d \) can be seen by a vertex of \( R \) which lies below \( \text{upt}(a) \cap R \). This means that the vertex of \( R \) immediately below \( \text{upt}(a) \cap R \) (say, denoted by \( x \)) also sees \( d \).

Since \( x \) is below \( \text{upt}(a) \cap R \), \( x \) sees all vertices of \( L \) between \( b \) and \( d \), including both \( b \) and \( d \). Additionally, if \( b \) is not the immediate neighbour of \( a \) on \( L \), then \( x \) sees the vertex of \( L \) immediately below \( b \) as well. Thus, \( x \) sees all points seen by \( b \) or \( c \) on \( L \).

Since \( x \) sees \( d \), all points of \( R \) that are seen by \( b \) or \( c \) and lie above \( x \), are also seen by \( d \). Since \( x \) lies below \( \text{upt}(a) \cap R \), \( a \) sees all points of \( R \) between and including \( x \) and \( \text{lot}(a) \cap R \). Since \( a \) lies below \( b \) and \( c \) on \( L \), none among \( b \) and \( c \) can see any vertex below \( \text{lot}(a) \cap R \). Thus, \( d \) and \( a \) see all points seen by \( b \) or \( c \) on \( R \).

The above arguments show that \( a \), \( d \) and \( x \) together see all the points on the two concave chains seen by \( b \) and \( c \). Observe that since \( x \) is the vertex immediately below \( \text{upt}(a) \cap R \), Algorithm 1 includes in \( G \) either \( x \), or a higher vertex \( x' \) of \( R \) which sees the points of \( F \) exclusively seen by \( x \). This means Algorithm 1 must choose \( x \) (or, \( x' \)) instead of \( b \) and \( c \) to optimize on the number of guards. Hence, we have a contradiction, and no vertex below the 13-interval can see \( d \).

Now suppose that \( d \) is seen by a vertex \( y \) lying above the 13-interval. Since the 13-interval contains two more guards on \( L \) above \( d \), the vertex \( y \) can certainly not lie on \( L \). Therefore, \( y \) must lie on \( R \). If the upper segment of the 13-interval is defined by \( \text{los}(v) \), where \( v \in R \), then \( d \), \( e \), \( f \) and \( g \) must lie below \( \text{los}(v) \cap L \). This means, \( \text{upt}(d) \cap R \) must lie below \( v \). But to see \( d \), \( y \) must lie below \( v \). This makes \( y \) a vertex contained in the 13-interval. So, \( v \) must lie on \( L \).

So, we assume that \( v \) lies on \( L \). At the worst case, \( v \) can be the guard \( f \). This means, \( \text{upt}(d) \cap R \) is above \( \text{lot}(q) \cap R \), and \( y \) lies on the segment of \( R \) between these two points. But then, by an argument similar to above, \( d \), \( g \) and \( y \) together see everything that is seen by \( d \), \( e \), \( f \) and \( g \), and so Algorithm 1 would choose only \( d \), \( g \) and \( y \) to get a shortest path in \( G \), a contradiction. So, the point \( d \) is not visible from any vertex outside of the 13-interval.

After Lemma 3.2, the second crucial ingredient of our coming arguments is the possibility to “almost privately” see the vertices of an interval \( Q \) from one point as follows. If \( s_2 = \text{los}(q) \), then the intersection point of \( \text{lot}(q) \) with \( s_1 \) is called the observer of \( Q \). (Actually, to be precise, we could slightly perturb this position of the observer \( o \) so that the visibility between
Guarding of simple polygons

Let and consider the spine of instead of lot(q). See again Figure 3.

Lemma 3.3. The observer of an interval in a funnel can see all the vertices of , but cannot see any vertex of which is not in and not in the shadow of .

Proof. Without the loss of generality let lie on . Since the observer is the intersection point of lot(q) with , sees all the vertices of , since all of them lie between and lot(q). Again, since the observer is the intersection point of lot(q) with , lies below a vertex on lot(q) ∩ . Then this vertex must be a blocker between and any point of above . So, cannot see any vertex of above that is not in . Let where without the loss of generality lies on . Then, since lies on , cannot see any vertex of below . The shadow of on extends till lot(r′) ∩ , where is the vertex of immediately above . But since lies on ups(r), is above r′, and hence cannot see vertices of below lot(r′) ∩ . Hence, cannot see vertices below the shadow of . □

The last ingredient before the main proof is the notion of sections of an interval of . Let and form the lower and upper boundary of . Consider a vertex such that the lower section of at lies the interval of bounded from below by and from above by . The upper section of at is the interval of bounded from below by and from above by . Sections of are defined analogously.

Lemma 3.4. Let be a -interval of the funnel , and let and be its lower and upper sections at some vertex . Then , , is a -interval such that . (In other words, at most 3 of the guards in are not in .)

Proof. The only vertices of which are not included in are and the vertices of the shadow of . Suppose, for a contradiction, that contain (at least) four guards from ; in either such case, we easily contradict minimality of the guard set in Algorithm 1 by the same argument given in Lemma 3.2. The Algorithm 1 simply chooses and the topmost and bottommost among these (at least) four guards, thus choosing only three guards instead of four. □

Now we are ready to prove the advertised lower bound, which follows straightforwardly using a “bisection approach” based on Lemma 3.4 with the base case in Lemma 3.2.

Theorem 3.5. Any conflict-free chromatic guarding of a given funnel requires at least \(|\log_2(m+3)| - 3\) colours, where is the minimum number of guards needed to guard the whole funnel.

Proof. We will prove the following claim by induction on :\n
- If is a -interval in the funnel and , then any conflict-free colouring of must use at least colours on the vertices of or the shadow of .

In the base , of the induction, we have . By Lemma 3.2, some point of is not seen from outside, and so there has to be a coloured guard in some vertex of , thus giving + 1 = 1 colour.

Consider now . The observer of which sees all the vertices of must see a guard of a unique colour where is, by Lemma 3.3 a vertex of or of the shadow of . In the first case, we consider and , the lower and upper sections of at . By Lemma 3.4, for some , is a -interval of such that . In the second case ( is in the shadow of ), we choose as the lowermost vertex of .
the same chain as $g$, and take only the upper section $Q_1$ of $Q$ at $g'$. We continue as in the first case with $i = 1$.

By induction assumption for $n-1$, $Q_i$ together with its shadow carry a set $C$ of at least $n$ colours. Notice that the shadow of $Q_2$ is included in $Q$, and the shadow of $Q_1$ coincides with the shadow of $Q$, moreover, the observer of $Q_1$ sees only a subset of the shadow of $Q$ seen by the observer $o$ of $Q$. Since $g$ is not a point of $Q_1$ or its shadow, but our observer $o$ sees the colour $c_g$ of $g$ and all the colours of $C$, we have $c_g \notin C$ and hence $C \cup \{c_g\}$ has at least $n+1$ colours, as desired.

Finally, we apply the above claim to $Q = F$. We have $k \geq m$, and for $k \geq m \geq 16 \cdot 2^n - 3$ we derive that we need at least $n + 1 \geq \lceil \log_2(3m + 3) \rceil - 3$ colours for guarding whole $F$. ◀

**Corollary 3.6.** There is a polynomial time algorithm which, given a funnel, outputs a conflict-free chromatic guard set $C$, such that the number of colours used by $C$ is by at most four larger than the optimum.

**Proof.** The algorithm simply runs Algorithm 1 to produce a guard sequence $A = (a_1, a_2, \ldots, a_k)$ (in order of a directed path in the auxiliary graph $G$) of our funnel $F$. We assign colours to members of $A$ according to the logarithmic sequence $1, 2, 1, 3, 1, 2, 1, 4, \ldots$, precisely, the vertex $a_i$ gets colour $c_i$ where $c_i$ is the largest integer such that $2^{c_i} \leq 2i$. Note the following simple property of this colouring: if $c_i = c_j$ for some $i \neq j$, then $c_{(i+j)/2} > c_i$. Consequently, for any $i, j$, the largest value occurring among colours $c_i, c_{i+1}, \ldots, c_{i+j-1}$ is unique. Hence, since every point of $F$ sees a consecutive subsequence of $A$, this is a feasible conflict-free colouring of the funnel $F$.

Let $m$ be the minimum number of guards needed to guard $F$. By Lemma 2.3, it is $m + 1 \geq k \geq m$. To prove the approximation guarantee, observe that for $k \leq 2^n - 1$, our sequence $A$ uses at most $n$ colours. Conversely, if $k \geq 2^n - 1$, then the required number of colours for guarding $F$ is at least $n - 1 - 3 = n - 4$, and hence our algorithm uses at most 4 more colours than the optimum. ◀

## 4 Vertex-to-point conflict-free chromatic guarding of simple polygons

In this section, we extend the scope of the studied problem of vertex-to-point conflict-free chromatic guarding from funnels to general simple polygons. We will establish an $O(\log n)$ upper bound for the number of colours of vertex-guards on $n$-vertex simple polygons, and give the corresponding polynomial time algorithm.

### Subcase of weak visibility polygons

We first establish the upper bound and design an algorithm for weak visibility polygons. We denote a polygon as $W$, with the vertex set of $W$ denoted by $V(W)$ and the edge set by $E(W)$. We say that $W$ is *weakly visible* from an edge $uv \in E(W)$ if every point of $W$ is visible from some point on $uv$. The union of Euclidean shortest paths from a vertex to all vertices of a simple polygon is called the geometric shortest path tree of the polygon, rooted at that vertex [19]. We start with constructing the geometric shortest path tree $T$ of all the vertices of our weak visibility polygon $W$, rooted at the vertex $u$ (see Figure 3).

Our aim is to place coloured guards at all non-leaf vertices of the tree $T$ (this is wasteful, but it makes our arguments smoother and does not increase the number of colours much). In order to use recursion in the algorithm (Alg. 3), we need to say when a partial solution is good: we say that an assignment of coloured guards to a set $X \subseteq V(W)$ has no conflict if,
12 Guarding of simple polygons

Figure 4 Geometric shortest path tree $T$ of a weak visibility polygon $W$ rooted at $u$; the edges of this shortest path tree are dashed in blue colour. Note that every root-to-leaf path of this tree is a left concave chain (this follows from $W$ being weakly visible from the edge $uv$). Our aim is to recursively place coloured guards at all non-leaf vertices of $T$, starting from the “leftmost” leaf-to-root path (the path $t$ to $u$) and continuing through $T$ intuitively to the right, cf. Algorithm 3. For every point $z$ of the polygon $W$, the subset of guards of $X$ visible from $z$ is either empty or contains a guard of a unique colour. Since every point of $W$ is visible from some non-leaf vertex of $T$, an assignment of guards to all non-leaf vertices, that has no conflict, gives a conflict-free chromatic guarding of $W$, as desired.

When assigning guard colours, we again use the logarithmic sequence (cf. Section 3) $c_1, c_2, c_3, \ldots$ where $c_i$ is the largest integer such that $2^{c_i}$ divides $2^i$. An important technical detail is that we actually assign the symbolic term $c_i$ to a guard $g$ (the colour of $g$ is then the value of $c_i$), where we distinguish between $c_i$ and $c_j$ for $i \neq j$ even if $c_i = c_j$ as colours. The logarithmic index of a guard $g$ which is assigned term $c_i$ then equals the index $i$.

In Algorithm 3, we refer to the following geometric setup: the polygon $W$ has $uv$ as its bottom (base) edge, with $u$ on the left. For a vertex $r$ of $T$ and its children $s_1, s_2$, we say that $s_1$ is to the left of $s_2$ if the straight line $rs_2$ separates $s_1$ from $v$ (see again Figure 4).

The core idea of the algorithm then is to construct a no-conflict guard assignment recursively from parents of leaves to the root of $T$, while ordering the children from left to right.

Lemma 4.1. Algorithm 3 computes a conflict-free chromatic guarding of the given weak visibility polygon $W$.

Proof. The algorithm recurses to all non-leaf vertices of $T$ (where $V(T) = V(W)$) exactly once, and so the recursion is finite (and polynomial). All steps of it are clearly sound except possibly the instruction on line 12 which we discuss now. Assume, at some iteration, that the assignment of guards to $D_{j-1}$ has no conflict, but the assignment to $D_j \cup B_j \setminus \{a_j\}$ has a conflict. Hence there is a point $z$ of $W$ seeing some guards of $D_{j-1}$ and some of $B_j \setminus \{a_j\}$, but none of a unique colour. Since $a_j$ is to the right of $a_1, \ldots, a_{j-1}$ and any path from $s$ through $a_j$ to a leaf is a left concave chain, the point $z$ must see $a_j$. Hence there is always a choice of term $c_m$ (e.g., of colour that has not appeared so far) for $a_j$ which makes $D_j$ conflict-free. Consequently, at the end we get a conflict-free chromatic guarding of $W$. ▶

While proving correctness of Algorithm 3 is relatively straightforward, the core is to prove that it uses only $O(\log n)$ colours on an $n$-vertex polygon $W$. For that we are going to inductively show, along the recursion of Alg. 3, that if the vertex $a_j$ is assigned term $c_m$ on line 12 then the number of already processed vertices is at least the logarithmic index $m$. ▶
Algorithm 3: V2P conflict-free chromatic guarding of a weak visibility polygon

**Input:** Weak visibility polygon $W$ whose base edge is $uv$ ($u$ to the left)

**Output:** A V2P conflict-free chromatic guarding of $W$

1. Build the geometric shortest path tree $T$ of $W$ rooted at $u$;
2. Let $c_i$ be the largest integer such that $2^i$ divides $2i$; /* logarithmic sequence */
3. Add a dummy root $u_0$ as the parent of $u$, and call ComputeSubassignmentOf($u_0$);
4. return guards on all non-leaf vert. of $T$, coloured by the values of assigned terms $c_i$;

5. Function ComputeSubassignmentOf($s$):
   /* here we assign symbolic terms $c_j$ of the logarithmic sequence to all non-leaf descendants of $s$ in $T$ (but not to $s$ itself) */
   if all children of $s$ are leaves or $s$ is a leaf then return;
   Let $a_1, a_2, \ldots, a_h$ be the non-leaf children of $s$, ordered from left to right;
   foreach $j := 1, 2, \ldots, h$ do
     Call ComputeSubassignmentOf($a_j$);
     Let $B_j$ consist of $a_j$ and all non-leaf descendants of $a_j$ in $T$, and $D_j = B_1 \cup \cdots \cup B_j$;
     /* note (for $j > 1$) that $B_j \setminus \{a_j\}$ is not visible from $D_{j-1}$ */
     Let $k$ be the highest (logarithmic) index of terms $c_k$ assigned to guards in $B_j$;
     Assign to $a_j$ the term $c_m$ where $m > k$ is the least index such that this assignment of guards to $D_j$ has no conflict;
   return the computed assignment to $D_h$;

Lemma 4.2. Let the vertex $a_j$ on line 12 of Algorithm 3 be assigned symbolic term $c_m$. Then $|D_j| \geq m$.

Proof. We first note on the ordering relation of the terms of the logarithmic sequence. If the first occurrence of the colour $c_i$ in the logarithmic sequence is before the first occurrence of the colour $c_j$, then we naturally write $c_i < c_j$. If the two colours are the same then $c_i = c_j$. From the definition of the logarithmic sequence we have that, if $c_i < c_j$ and there is no $c_k$ such that $c_i < c_k < c_j$, then $j = 2i$.

The proof of the lemma is by induction on the vertices of $T$. For the base case, let $a_1$ be a non-leaf vertex $a_1$ of $T$, such that all the children of $a_1$ are leaves. Clearly, the claim holds for $a_1$, because in this case, $|D_1| = m = 1$. Now let $a_j$ be any non-leaf vertex of $T$. By our induction hypothesis, the claim holds true for all non-leaf children of $a_j$.

Suppose that $k$ is the highest logarithmic index assigned to any non-leaf vertex of $D_j \setminus \{a_j\}$. Let $a_j$ be a non-leaf child of $a_j$ such that the logarithmic index $k$ is assigned to some vertex in $D_i$. If $c_m \leq c_k$ then $|D_j| > |D_i| \geq k \geq m$, and the claim holds. Otherwise, if $c_m > c_k$, then $k$ must be occurring at least twice as a logarithmic index in $D_j \setminus \{a_j\}$. If it occurs twice in the same path from $a_j$ to one of its descendant leaves of $T$, then some point of $W$ sees both of its occurrences since the path is a concave chain and $W$ is a weak visibility polygon. Furthermore this point sees no vertex of $T$ other than the path in question. Then a higher logarithmic index is required on the path to prevent conflicts, contradicting the maximality of $k$ in $D_j \setminus \{a_j\}$.

This means, that there exists a vertex $a_p \in D_j$ (with the possibility that $a_p = a_j$), such that there are at least two distinct non-leaf children of $a_p$ (say, $a_p$ and $a_q$), such that the logarithmic index $k$ occurs once in each of $D_p$ and $D_q$. Then, by our induction hypothesis,
Guarding of simple polygons

$|D_j| > |D_p| + |D_q| \geq 2k$. But due to Algorithm 3, $m$ is the lowest possible logarithmic index of $c_m$ required to prevent conflicts. Since $k$ is the highest logarithmic index among all the vertices in $D_j \setminus \{a_j\}$, as reasoned above, $m = 2k$. Then it follows that $|D_j| \geq m$. ▶

Using lemmas 4.1 and 4.2, we arrive at the following theorem, where we also use the method described in [19] to check for conflicts in polynomial time.

Theorem 4.3. Every $n$-vertex weak visibility polygon has a V2P conflict-free chromatic guarding with $O(\log n)$ colours, as computed in polynomial time by Algorithm 3.

Proof. By Lemma 4.1, the output of Algorithm 3 gives a conflict-free chromatic guarding. By Lemma 4.2, whenever the highest colour $c_m$ has been assigned by the algorithm, we have already encountered at least $m$ vertices of $W$, and $c_m \in O(\log m) \subseteq O(\log n)$.

As for runtime analysis, the shortest path tree $T$ is computed and handled in polynomial time using standard means, and the recursion has $O(n)$ iterations. It remains to argue about testing for conflicts for $D_j$ on line 13. This can be done in polynomial time simply as follows.

Construct the visibility polygons of all the vertices of $W$. The constructed edges so formed partition $W$ into $O(n^4)$ convex regions, in $O(n^4)$ time using the method of Ghosh [19]. It also follows from [19] that the interior, edges, and vertices of each such convex regions are distinct sets of points, such that all the points of each such set see the same polygonal vertices. A point from each of these sets can be chosen, and its unique-colour guard can be searched for in $O(n \log n)$ time, since only $O(\log n)$ colours are used. So, the whole procedure takes polynomial time as well. ▶

Upper bound for simple polygons

Secondly, we use a polygon decomposition technique introduced by Suri [29] to generalize the upper bound from weak visibility polygons to all simple polygons. This technique has already been used by Bártscśi et al. in [4] for the P2P version of our problem. Consequently, we get the following final result – Theorem 4.3, as a straightforward corollary of Theorem 4.3 and this decomposition technique.

We now outline the polygon decomposition technique of Suri [29]. Consider a given simple polygon $P$. Consider any edge $uv$ of $P$ a consider its weak visibility polygon $W_{uv}$. Then $W_{uv}$ is separated from the rest of $P$ by constructed edges, where each constructed edge is a line segment on a ray emanating from $uv$. We assume that the whole of $P$ is not weakly visible from $uv$. Then these constructed edges separate $P$ into $W$ and one or more other subpolygons. Consider $W_e$, the weak visibility polygon of any edge $e$ of $P$, and its corresponding subpolygons. Such a subpolygon, if it lies to the left (or, right) of the ray containing its corresponding constructed edge with respect to $W_e$, is called a left (respectively, right) subpolygon [4]. Let $A$ and $B$ be any two subpolygons, (or, two right subpolygons) formed by the constructed edges of a weak visibility polygon $W_e$. It is shown in [4] that no interior points $A$ and $B$ can see each other, although the vertex on $A \cap W_e$ may see interior points of $B$ and vice-versa. We state the corresponding lemma from [4]. We slightly change the notation and adapt the wording for our further use.

Lemma 4.4. Let $P$ be a simple polygon, and let $W_e$ be a weak visibility polygon with base edge $e \in P$. Let $W_L$ and $W_R$ be the collections of left and right subpolygons with respect to $W_e$, respectively. Let $G_L$, $G_R$ be disjoint guard sets of $W_L$ and $W_R$ such that no guard $g \in G_L$ (respectively $G_R$) lies on a vertex of $W_e$. Define two subpolygons $A, B \in W_L$. Then there are no two points $a$ and $b$ such that $a \in A \setminus W_e$ and $b \in G_L \cap B$ that are mutually visible. Up to symmetry, this claim also holds when $A, B \in W_R$, $b \in B \setminus W_e$, and $a \in G_R \cap A$. 

Our decomposition process includes a slight modification of the decomposition method used in [4], because theirs was a case of P2P guarding. In P2P guarding, during the decomposition method, points suitably near to the endpoints of the constructed edge that cuts out a new weak visibility polygon, can be chosen to place guards, because the guards in general may lie anywhere in the polygon. But in the V2P guarding that we deal with, guards must be placed only on vertices, which might be inconveniently far from the endpoints of the constructed edges. So, hence each time a new weak visibility polygon is found using the method of [4], it must be further divided into major and minor subpolygons, which we discuss below.

We continue the process of decomposition, successively using the constructed edges as the basis for further decomposition $P$. Suppose that we have a weak-visibility polygon $W$ of $P$ in the present step of the decomposition. Without loss of generality, consider a left subpolygon $W_L$ of $W$. By definition, $W_R$ lies to the left of a constructed edge. Note that either one or both endpoints of the constructed edge are vertices of $P$. Denote the constructed edge by $p_ip_j$, where $P_i$ lies to the left of the ray originating at $p_i$ and passing through $p_j$. Traverse the boundary of $P$ from $p_i$ to $p_j$ in the clockwise order and let $y$ denote the vertex of $P$ encountered first, where $p_i \neq y$. Similarly, traverse the boundary of $P$ from $p_j$ to $p_i$ in the counterclockwise order and let $z$ denote the vertex of $P$ encountered first, where $p_j \neq z$. Let the geometric shortest path between $y$ and $z$ be $yv_1v_2\ldots v_kz$, where $v_1,v_2,\ldots,v_k$ are vertices of $P_i$. We call the weak visibility polygons of the edges $yv_1,v_1v_2,\ldots,v_kz$ on the left of the rays $\overrightarrow{yv_1},\overrightarrow{v_1v_2},\ldots,\overrightarrow{v_kz}$ the left major subpolygons of $P_w$ (See Figure 5). We call the polygon $p_1yv_1\ldots v_kz$ a left minor subpolygonal child of $P_w$. Right major subpolygonal children and right minor subpolygonal children are defined analogously. In general, we refer to them as major and minor weak-visibility polygons as well. Next we continue the process by using left major and right major subpolygonal children of $P_w$ in the same manner. Note that the major subpolygonal children share an edge and its two end-vertices with the parent minor subpolygon, and two major subpolygonal children of a minor subpolygon may share a vertex in common with each other. To extend the classification to all the polygons obtained by the decomposition process of Algorithm 4, we consider the first subpolygon obtained during the decomposition as a major subpolygon.

In Figure 5 we see a simple polygon $P$. The vertices labeled as $u$ and $v$ form the edge $\overrightarrow{uv}$. The weak visibility polygon $W$ of the edge $\overrightarrow{uv}$ is denoted by red area. Blue and red areas, labeled with $W_w$, $W_L$, and $W_R$, denote the left subpolygon, and the right subpolygons of $P$, respectively.

Now consider three colour sets, $C_1$, $C_2$, and $C_3$, having all distinct colours. Let $|C_1| = |C_2| = |C_3| = 2S$, where $S$ is the upper bound given by Algorithm 3. We colour a subpolygon $P'$ by one of the colour sets $C_1$, $C_2$, and $C_3$, using the method of Algorithm 3 in the following way. If a subpolygon is a major subpolygon, then we colour it using the second half of the colour set that it is assigned. If it is a minor subpolygon, we use the second half of the colour set that it is assigned. We colour the subpolygons in the same order in which they were obtained while decomposing $P$ using Algorithm 4. If $P'$ is the first weak visibility polygon of the decomposition, then we colour it simply using Algorithm 3. If $P'$ is a minor subpolygon, then its base is a constructed edge, and has at least one polygonal vertex of the original polygon $P$. We choose this vertex as the root for the geometric shortest path tree of Algorithm 4. If the root vertex is already coloured, we let it retain its colour and proceed with colouring the rest of the shortest path tree using Algorithm 3, and likewise proceed with the colouring.

If $P'$ is a major subpolygon, then we must have coloured its parent minor subpolygon
Algorithm 4: Decomposition of a simple polygon into weak visibility polygons

**Input:** Simple polygon \( P \)

**Output:** A decomposition of \( P \) into weak visibility polygons

1. Choose an edge \( uv \) of \( P \) and construct the weak visibility polygon \( W_{uv} \) of \( uv \);
2. Let \( S_\alpha = \emptyset \) be the set of constructed edges for drawing major subpolygons;
3. Let \( S_\beta = \emptyset \) be the set of constructed edges for drawing minor subpolygons;
4. Let \( W = \emptyset \) be the set of weak visibility polygons;
5. \( W := \{ W_{uv} \} \);
6. Initiate \( S \) with the constructed edges of \( W_{uv} \);
7. \( i \leftarrow 1 \);
8. while \( S_\alpha \cup S_\beta \neq \emptyset \) do
   9. Choose an edge \( e \in S_\alpha \cup S_\beta \);
10. if \( e \in S_\beta \) then
    11. Consider the weak visibility polygon of \( e \) whose interior does not intersect with any weak visibility polygon in \( W \) and denote it by \( P_e \);
    12. Traverse the boundary of \( P_e \) in the clockwise order starting from any vert. in \( e \);
    13. Denote the first vertex encountered, which is not a vertex of \( e \), by \( y \);
    14. Denote the last vertex encountered, which is not a vertex of \( e \), by \( z \);
    15. Denote the geometric shortest path between \( y \) and \( z \) by \( yv_1v_2...v_kz \), where \( v_1, v_2, ..., v_k \) are vertices of \( P_e \);
    16. Denote the polygon \( yv_1v_2...v_kz \) by \( W_e \);
    17. \( W = W \cup \{ W_e \} \);
    18. \( S_\alpha = S_\alpha \cup \{ yv_1 \cup v_1v_2 \cup ... \cup v_kz \} \);
19. else if \( e \in S_\alpha \) then
20. Construct the weak visibility polygon of \( e \) and denote it by \( W_e \);
21. \( W = W \cup \{ W_e \} \);
22. \( S_\beta = S_\beta \cup \{ \text{all constructed edges of } W_e \} \);
23. \( S_\alpha \cup S_\beta = (S_\alpha \cup S_\beta) \ \{ e \} \)
24. return \( W \);

before. So, one or both of the base vertices of \( P' \) might contain colours from the colouring of the minor subpolygon. We choose one of these vertices as the root of the geometric shortest path tree of Algorithm 3. If the root vertex is already coloured, again, as before, we let it retain its colour and proceed with colouring the rest of the shortest path tree using Algorithm 3. It can be seen that in each case conflicts are prevented, since the already coloured vertex gets its unique colour guard via the colouring of the parent subpolygon, and the rest of the vertices get their unique colour guard due to Lemma 4.1.

If we use \( C_i \) on \( P' \), then we use \( C_j \) and \( C_k \) on its left and right major subpolygons, respectively, where \( i \neq j \neq k \). Suppose that we find more than one left major subpolygon of \( W \) sharing boundaries with a left minor subpolygon of \( W \). Without loss of generality, suppose that we have to use \( C_1 \) to colour these left major subpolygons using Algorithm 3. Then, we use only half of the colours in \( C_1 \) for colouring these left major subpolygons. The left major subpolygon is yet to have a conflict-free chromatic guarding. But by our construction, the left
Algorithm 5: V2P conflict-free chromatic guarding a simple polygon

**Input:** A simple polygon $P$.

**Output:** A conflict-free chromatic guarding of $P$ with $O(n \log n)$ colours.

1. Decompose $P$ into weak visibility polygons with Algorithm 4 and denote the starting edge by $uv$; /* Algorithm 4 returns $W$ */
2. Let $C_1$, $C_2$ and $C_3$ be three colour sets each having twice the number of colours required by Algorithm 3;
3. Colour $W_{uv} \in W$ with $C_1$ using Algorithm 3;
4. Assign $C_2$ to the left minor and major subpolygons of $W_e$;
5. Assign $C_3$ to the right minor and major subpolygons of $W_e$;
6. $W = W \setminus W_{uv}$;
7. while $W \neq \emptyset$ do
   8. Choose a $W_e \in W$ such that $W_e$ is a major or minor subpolygonal child of an already coloured subpolygon ;
   9. if $W_e$ is a minor subpolygon then
      10. Choose a base-edge vertex of $W_e$ that is a vertex of $P$;
      11. Use this vertex as a root to compute the geometric shortest path tree $T_e$ for Algorithm 3;
      12. if The root vertex of $T_e$ is already coloured then
         13. Let this vertex retain its colour, and use Algorithm 3 to colour the rest of $W_e$ with the first half of the colour set assigned to it;
      else
         15. Use Algorithm 3 to colour $W_e$ with the first half of the colour set assigned to it;
         16. $W = W \setminus W_e$;
      17. else if $W_e$ is a major subpolygon then
         18. Compute the geometric shortest path tree $T_e$ for Algorithm 3;
         19. if The root vertex of $T_e$ is already coloured then
            20. Let this vertex retain its colour, and use Algorithm 3 to colour the rest of $W_e$ with the second half of the colour set assigned to it;
         else
            22. Use Algorithm 3 to colour $W_e$ with the second half of the colour set assigned to it;
            23. $W = W \setminus W_e$;
         24. if $C_1$ is the colour set that was assigned to $W_e$ then
            25. Assign $C_2$ to all the left subpolygonal children of $W_e$;
            26. Assign $C_3$ to all the left subpolygonal children of $W_e$;
         else if $C_2$ is the colour set that was assigned to $W_e$ then
            27. Assign $C_3$ to all the left subpolygonal children of $W_e$;
            28. Assign $C_1$ to all the left subpolygonal children of $W_e$;
         else if $C_3$ is the colour set that was assigned to $W_e$ then
            29. Assign $C_1$ to all the left subpolygonal children of $W_e$;
            30. Assign $C_2$ to all the left subpolygonal children of $W_e$;
      24. return Coloured $P$;
Figure 5 A simple polygon, decomposed into weak visibility polygons, starting from the edge \(uv\). The shades of green, red and blue represent the colour sets \(C_1\), \(C_2\) and \(C_3\) respectively, and are repeated to colour the whole polygon. The dark red polygon \(yv_1v_2zp\) is a minor subpolygon while the light red weak visibility polygon of \(v_1v_2\) is a major subpolygon.

major subpolygon is a weak visibility polygon where the vertices that are not the endpoints of the weak visibility edge, form a concave chain. So, we use the other half of the colours of \(C_1\) to colour all vertices of this concave chain according to the logarithmic sequence. We formalize the above method in Algorithm 5. Clearly, this preserves the conflict-free chromatic guardings of the left major subpolygons and gives a conflict-free chromatic guarding to the left major subpolygon. We have the following theorem.

\[\text{Theorem 4.5.} \text{ Simple polygons on } n \text{ vertices can be vertex-to-point conflict-free chromatic guarded with } O(\log n) \text{ colours, computed in polynomial time.}\]

**Proof.** Algorithm 5 uses Algorithm 3 to colour each weak visibility polygon. Thus, by Theorem 4.3, there is no conflict inside any weak-visibility polygon as far as only its own vertices are concerned. Due to Lemma 4.4, the decomposition of the polygon using Algorithm 4 ensures that the interior points of no two children of a weak visibility polygon in the decomposition can see each other. So, the only possible points of conflict are the vertices of constructed edges formed by a major weak-visibility polygon. But due to Algorithm 4, these must belong to different minor weak-visibility polygons, and during colouring a minor weak visibility polygon, Algorithm 5 does not colour the already coloured vertices of the constructed edge; it instead lets them retain their unique colour from the parent subpolygon. Thus, no point of the simple polygon encounters a conflict, and since all the three colour sets \(C_1\), \(C_2\) and \(C_3\) have a constant multiple of \(\log n\) colours, we get the \(O(\log n)\) upper bound for simple polygons.

The running time is polynomial, which directly follows from the polynomial running times of algorithms 3 and 4.

Unfortunately, our method to produce a conflict-free chromatic guarding with \(O(\log n)\) colours does not immediately generalize to a constant ratio algorithm. E.g., in Figure 6 the...
Figure 6 A fragment of a polygon whose chromatic V2P conflict-free guarding can be obtained using the depicted four colours, but our algorithm of Theorem 4.5 (Alg. 5) generally assigns $O(\log n)$ colours to it.

Figure 7 Two examples of simple polygons requiring at least 2 colours for a V2V conflict-free chromatic guarding (in each, one vertex guard cannot see all the vertices, and any two guards of the same colour make a conflict).

application of the algorithm of Theorem 4.5 requires $O(\log n)$ colours, but only four colours are sufficient for the whole polygon.

5 Vertex-to-vertex conflict-free chromatic guarding

In the last section, we turn to the vertex-to-vertex variant of the guarding problem. As we have noted at the beginning, the V2V weak conflict-free chromatic guarding problem coincides with the graph conflict-free colouring problem [1] on the visibility graph of the considered polygon. While, for the latter graph problem, [1] provided constructions requiring an unbounded number of colours, it does not seem to be possible to adapt those constructions for polygon visibility graphs (and, actually, we propose that the conflict-free chromatic number is bounded in the case of polygon visibility graphs, see Conjecture 5.2).

Lower bound for V2V conflict-free chromatic guarding

We start by showing in Figure 7 that one colour is not always enough. To improve this very simple lower bound further, we will need a more sophisticated construction (next).

Theorem 5.1. There exists a simple polygon which has no V2V conflict-free chromatic guarding with 2 colours.

Proof. We call a “bowl” the simple polygon depicted in Figure 8 (note that it contains two copies of the shape from Fig. 7). In particular, the vertices $p_1, p_2$ see all other vertices of the bowl. We claim the following:

(i) In any conflict-free 2-colouring of the bowl there is a guard placed on $p_1$ or $p_2$ (or both).

Assume the contrary to (i), that is, existence of a 2-colouring of the bowl avoiding both $p_1$ and $p_2$. One can easily check that the subset $A = \{a_1, \ldots, a_8\}$ of the vertices requires both colours, with guards possibly placed at $A \cup \{a_9\}$. Symmetrically, there should be guards.
Guarding of simple polygons

A construction of an example requiring at least 3 colours for a V2V conflict-free chromatic guarding (cf. Theorem 5.1). The bowl shape (see on the left), suitably squeezed and with a tiny opening between \( p_1 \) and \( p_2 \), is placed to four positions within the bowtie shape on the right.

of both colours placed at the disjoint subset \( \{c_1, \ldots, c_8, c_9\} \). Then we have got a colouring conflict at both \( p_1 \) and \( p_2 \), thus proving (i). (On the other hand, placing a single guard at either \( p_1 \) or \( p_2 \) gives a feasible conflict-free colouring of the bowl.)

The next step is to arrange four copies of the bowl within a suitable simple polygon, as depicted on the right hand side of Figure 8. More precisely, let \( S \) be the (bowtie shaped) polygon on the right of the picture. Note that the chains \( C_1 = (r_2, r_1, t, s_1, s_2) \) and \( C_2 = (r_3, r_4, t', s_4, s_3) \) of \( S \) are both concave, and each vertex of \( C_1 \) sees all of \( C_2 \). We construct a polygon \( S' \) from \( S \) by making a tiny opening at each of the vertices \( q_1, q_2, q_3, q_4 \), and gluing there a suitably rotated and squeezed copy of the bowl, where gluing is done along a copy of the edge \( p_1 p_2 \) of the bowl. We call these openings at former vertices of \( S \) the doors of \( S' \). Obviously, the doors can be made so tiny that there is no accidental visibility between a vertex inside a bowl and a vertex belonging to the rest of \( S' \).

Assume, for a contradiction, that \( S' \) admits a conflict-free colouring with two colours, say red and blue. Up to symmetry, let the unique colour seen by vertex \( t \) be blue. Since \( t \) sees all four doors, and (i) every door has a guard, at least three guards at the doors are red. Hence the unique colour seen by symmetric \( t' \) must also be blue. Consequently, either there is only one blue guard at one of \( r_1, r_4, s_1, s_4, q_1, q_2, q_3, q_4 \) (which all see both \( t \) and \( t' \)), or there are two blue guards suitably placed at a pair of vertices from \( r_2, r_3, s_2, s_3 \) (each of those sees one of \( t, t' \)). Moreover, a single blue guard cannot be placed at one of the doors \( q_1, q_2, q_3, q_4 \), because that would leave \( r_3 \) or \( s_3 \) unguarded (seeing two red and no blue). Consequently, the guards placed at the doors \( q_1, q_2, q_3, q_4 \) must all be red, and so all the vertices \( t, r_1, r_2, r_3, r_4, t', s_1, s_2, s_3, s_4 \) must be guarded by a blue guard. The latter is clearly impossible without a conflict (similarly as in Fig. 7).

Our thorough investigation of the V2V polygon chromatic guarding problem, although not giving further rigorous claims (yet), moreover suggests the following conjecture:

**Conjecture 5.2.** Every simple polygon admits a weak conflict-free vertex-to-vertex chromatic guarding with at most 3 colours.

**Hardness of V2V conflict-free chromatic guarding**

In view of Conjecture 5.2 and the algorithmic results for other variants of chromatic guarding, it is natural to ask how difficult is to decide whether using 1 or 2 colours in V2V guarding is
enough for a given simple polygon. Actually, for general graphs the question whether one can find a conflict-free colouring with 1 colour was investigated already long time ago (1973 under the name of a perfect code in a graph) 8, and its NP-completeness was shown by Kratochvíl and Krivánek in 25. Previous lower bounds and hardness results in this area, including recent 1, however, seem to say nothing specifically about the case of polygon visibility graphs.

Here we show that in both the cases of 1 or 2 guard colours, the conflict-free chromatic guarding problem on arbitrary simple polygons is NP-complete. In each case we use a routine reduction from SAT, using a “reflection model” of a formula which we have introduced recently in 12 for showing hardness of the ordinary chromatic number problem on polygon visibility graphs. This technique is, on a high approximate level, shown in Figure 9: the variable values (T or F) are encoded in purely local variable gadgets, which are privately observed by opposite reflection (or copy) gadgets modelling the literals, and then reflected within precisely adjustable narrow beams to again opposite clause gadgets. Furthermore, there is possibly a special guard-fix gadget whose purpose is to uniquely guard the polygonal skeleton of the whole construction, and so to prevent interference of the skeleton with the other local gadgets (which are otherwise “hidden” from each other).

▶ Theorem 5.3. For $c \in \{1, 2\}$, the question whether a given simple polygon admits a weak conflict-free vertex-to-vertex chromatic guarding with at most $c$ colours, is NP-complete.

Proof. For $c = 1$ we reduce from the 1-in-3 SAT variant, which asks for an assignment having exactly one true literal in each clause and which is NP-complete, too. Note that in this case, there is only one colour of guards, and so every vertex must see exactly one guard. Following the general scheme of Figure 9, we now give the particular variable and reflection gadgets:

- The variable gadget is depicted in Figure 10 on the left. The opening between $d_1$ and $d_2$ is sufficiently small to prevent accidental visibility between an inner vertex of this gadget (gray in the picture) and other vertices outside.
- In any conflict-free 1-colouring of a polygon containing this gadget, precisely one of $d_1$, $d_2$ must have a guard (otherwise, the inner vertices cannot be guarded, as in Figure 7, and
there is no other guard on this gadget.

The reflection gadget is depicted in Figure 10 on the right. Note that the visible angles of the vertices \( q \) and \( q' \) can be made arbitrarily tiny and fine-adjusted (independently of each other) by changing horizontal positions of \( q, q', \) and \( s \).

In any conflict-free 1-colouring of a polygon containing this gadget, such that both \( a_1, a_2 \) see a guard (from outside), the following holds: a guard can be placed only at \( q' \) or \( r \), and the guard is at \( q' \) if and only if \( q \) sees a guard from outside (otherwise, there would be a conflict at \( q \) or \( q \) would not be guarded).

Every clause gadget is just a single vertex positioned on a concave chain as shown in Figure 9. There is no guard-fix gadget present in this construction.

For a given 3-SAT formula \( \Phi = (x_i \lor \neg x_j \lor x_k) \land \ldots \), the construction is completed as follows. Within the frame of Figure 9 we place a copy of the variable gadget for each variable of \( \Phi \). We adjust these gadgets such that the combined visible angles of \( d_1, d_2 \) of each variable gadget do not overlap with those of other variables on the bottom base of the frame. (The lower left and right corners of the frame are seen by the first and last variable gadgets, respectively.)

For each literal \( \ell \) containing a variable \( x_i \) we place a copy of the reflection gadget at the bottom of the frame, such that its opening \( a_1, a_2 \) is visible from both \( d_1 \) and \( d_2 \) of the gadget of \( x_i \). We adjust the visible angle of \( q \) such that \( q \) sees \( d_1 \) but not \( d_2 \) if the literal \( \ell \) is \( x_i \), and \( q \) sees \( d_2 \) but not \( d_1 \) if \( \ell = \neg x_i \). Then we adjust the visible angle of \( q' \) such that it sees exactly the one vertex (on the top concave chain of the frame) which represents the clause containing \( \ell \).

This whole construction can clearly be done in polynomial time and precision (cf. the similar arguments in [12]). To recapitulate, each of the vertices of our constructed polygon is

- coming from a copy of the variable gadget for each variable of \( \Phi \), or
- from a copy of the reflection gadget added for each literal in \( \Phi \), or
- is a singleton representative of one of the clauses of \( \Phi \), or
- is the auxiliary lower-left or right corner.
Assume that $\Phi$ has a 1-in-3 satisfying assignment. Then we place a guard at $d_1$ of a variable gadget of $x_i$ if $x_i$ is true, and at $d_2$ otherwise. Moreover, at a reflection gadget of a literal $\ell$, we place a guard at $q'$ if $\ell$ is true, and at $r$ otherwise. No other guards are placed.

In this arrangement, every vertex of a variable or reflection gadget sees precisely one guard, regardless of the evaluation of $\Phi$. Moreover, since the assignment of $\Phi$ makes precisely one literal of each clause true, every clause vertex also sees one guard. This is a valid conflict-free 1-colouring.

Conversely, assume we have a conflict-free 1-colouring of our polygon. Since precisely one of $d_1, d_2$ of each variable gadget of $x_i$ has a guard, this correctly encodes the truth value of $x_i$ (true iff the guard is at $d_1$), and we know that both $a_1, a_2$ of each reflection gadget of a literal $\ell$ see an outside guard. Hence, by our construction, the gadget of $\ell$ has a guard at $q'$ if and only if $\ell$ is true in our derived assignment of variables.

Furthermore, any clause vertex can see a guard only at a vertex $q'$ of some reflection gadget. Otherwise (including the case of a guard at some clause vertex), we would necessarily get a guard conflict at a vertex $a_1$ of some reflection gadget. Consequently, every clause contains precisely one true literal (as determined by the guard visible from this clause vertex). The NP-completeness reduction for $c = 1$ is finished.

We now move onto the $c = 2$ case, which we reduce from the NP-complete not-all-equal positive 3-SAT problem (also known as 2-colouring of 3-uniform hypergraphs). This special variant of 3-SAT requires every clause to have at least one true and one false literal, and there are no negations allowed. We again follow the same general scheme as for $c = 1$, but this time the main focus will be on implementing the guard-fix gadget. Let our guard colours be

![Figure 11](image-url)
Guarding of simple polygons

Figure 12 Placement of the variable/reflection/clause gadgets for conflict-free 2-colouring: the clause gadgets are the single vertices $c_1, c_2, \ldots$, the variable gadgets are formed by copies of the bowl at $x_1, x_2, \ldots$, and the reflection gadgets are simply triples of vertices as $l_1, l_2, l_3$ at the bottom line. In this example, the value (colour red or blue) of the variable $x_1$ is reflected towards clauses $c_1$ and $c_4$ (which are thus assumed to contain literal $x_1$), and the value of $x_2$ is reflected towards $c_3$.

red and blue (then every vertex must see exactly one red guard, or exactly one blue guard).

The left and right walls of the schematic frame from Figure 9 are constructed as shown in Figure 11. In the construction, we use four copies of the bowl shape from Figure 8 and we adopt the terminology of glueing the bowls and of the doors from the proof of Theorem 5.1. For simplicity, while keeping in mind that the door of each bowl is a narrow passage formed by a pair of vertices, we denote each door by a single letter $d_i$, $i = 1, 2, 3, 4$ (as other vertices).

This guard-fix gadget is constructed such that both $d_1, d_2$ see all the $8$ vertices of the “pocket” $A$ on the left, but neither of $a_1, a_2$ does so. We recall the following property from the proof of Theorem 5.1:

(i) In any conflict-free 2-colouring of the bowl there is a guard (or two) placed at the door. Consequently, the guards placed at $d_1$ and $d_2$ must be of different colours (red and blue); otherwise, say for two red guards at $d_1, d_2$, each of the $8$ vertices in $A$ would have to be guarded by a blue guard within $A \cup \{a_1, a_2\}$ which is not possible. (Though, we have not yet excluded the case that, say, $d_1$ would have a red guard and $d_2$ a blue and a red guards.)

In the next step, we note that $d_3, d_4$ see all the $8$ vertices of the “pocket” $B$, but none of $b_1, b_2, c_3, c_4$ does so. So, by analogous arguments, the guards placed at $d_3, d_4$ must be of different colours (red and blue). We remark that this does not necessarily cause a conflict with the guards at $d_1, d_2$ since the visibility between $b_1, d_3$ and between $b_2, d_4$ is blocked by $c_3$ and $c_4$, respectively. However, the vertices $b_1$ and $b_2$ are now “exhausted” in the sense that one sees (at least) one red and two blue guards, and the other one blue and two red guards. Altogether, this implies that

(ii) there is exactly one red and one blue guard among $d_1, d_2$ and the same holds among $d_3, d_4$, and no other vertex visible from both $b_1$ and $b_2$ can have a guard.

The rest of the construction is, within the frame constructed above (Figure 11), already quite easy. Let $\Phi$ be a given 3-SAT formula without negations. See Figure 12.

We again represent each clause of $\Phi$ by a single vertex on a concave chain on the top of our frame. This chain of clause vertices is “slightly hidden” in a sense that it is not visible from $d_1$ or $d_2$, but it is all visible from $b_1$ and $b_2$. 
- Each variable $x_i$ of $\Phi$ is represented by a copy of the bowl, also placed on the top of the frame (but separate from the clause vertices). As before, the visible angles of the variable gadgets are adjusted so that they do not overlap on the bottom line of the frame.
- Each literal $l = x_i$ is represented by a triple of vertices as $l_1, l_2, l_3$ in Figure 12, such that $l_1, l_3$ are in the visible angle of the variable-$x_i$ gadget, while $l_2$ is “deeply hidden” so that $l_2$ sees only the clause vertex $c_j$ that $l$ belongs to.

The final argument is analogous to the $c = 1$ case. Assume we have a not-all-equal assignment of $\Phi$. We put red guards to $d_1$ and $d_3$ and blue guards to $d_2$ and $d_4$. For each variable $x_i$, we put one guard to $x_1$ of blue colour if $x_i$ is true and of red colour if $x_i$ is false. Whenever $\ell = x_i$ is a literal represented by the triple $l_1, l_2, l_3$, we put to $l_2$ a guard of the same colour as of the guard at $x_i$. The every clause $c_j = (x_a \lor x_b \lor x_c)$ will see the colours of guards at $x_a, x_b, x_c$, and since the values assigned to $x_a, x_b, x_c$ are not all the same, one of the colours is unique to guard $c_j$. We have got a conflict-free 2-colouring.

On the other hand, assume a conflict-free 2-colouring. By (ii), there are no guards at the clause vertices $c_1, c_2, \ldots$, and so those vertices can be only guarded from the reflection gadgets. Assume a reflection gadget of a literal $\ell = x_i$. Then, again by (ii), the vertices $l_1, l_3$ of this gadget have no guards, and they see a red and a blue guard from $d_1, d_2$. On the other hand, $l_2$ cannot see any other guard except one placed at $l_2$, and so there has to be a guard at $l_2$. If, moreover, $l_1, l_3$ see a red (say) guard placed at $x_i$ (and $x_i$ as the door of a glued bowl there must have a guard by (i)), then the guard at $l_2$ must also be red (or $l_1$ would have a conflict). Consequently, every clause $c_j = (x_a \lor x_b \lor x_c)$ sees the colours of the guards placed at $x_a, a_0$ and $x_c$, and since the colouring is conflict-free, there have to be both colours visible (one red plus two blue, or one blue plus two red). From this we can read a valid not-all-equal variable assignment of $\Phi$. □

6 Concluding remarks

We have designed an algorithm for producing a V2P guarding of funnels that is optimal in the number of guards. We have also designed an algorithm for a V2P conflict-free chromatic guarding for funnels, which gives only an additive error with respect to the minimum number of colours required. We believe that the latter can be strengthened to an optimum solution by sharpening the arguments involved (though, it would likely not be easy).

For V2P conflict-free chromatic guarding simple polygons, we have given an efficient upper bound of $O(\log n)$, first for the special case of weak-visibility polygons and then, as a corollary, for all simple polygons. However, this result only gives an upper bound, which can be arbitrarily bad compared to the optimum solution, as demonstrated in Figure 12. However, we think that generalizing the technique of Algorithm 1 could possibly give a constant ratio approximation algorithm at least for weak-visibility polygons.

For the V2V guarding case, it is easy to see that there is a natural upper bound by the domination number of the visibility graph. Also, in the V2V case for funnels, our Algorithm 1 can be used to decide whether a given funnel can be guarded by only one colour, whereas a trivial V2V conflict-free chromatic guarding with two colours exists for any funnel (see Figure 13).

For general simple polygons, we have shown that the problem of V2V conflict-free chromatic guarding is NP-complete when the number of colours is one or two. We have also given a simple example where the polygon requires three colours, and conjectured that no more colours are needed no matter how convoluted the polygon might be.
Guarding of simple polygons

Figure 13 An illustration of a V2V conflict-free chromatic guarding of a funnel with only two colours. Every vertex sees a unique colour guard on its own concave chain.

Although our V2P upper bound of $O(\log n)$ colours implies the P2P upper bound for simple polygons of [4], our V2V NP-hardness results do not readily imply hardness results for the V2P and P2P cases. Hence, the complexity of the latter two cases remain open.

To summarize, we propose the following particular open problems for future research:

- Improve Corollary 3.6 to an exact algorithm for guarding a funnel.
- Turn an upper bound in Theorem 4.3 to a constant-factor approximation algorithm.
- Prove Conjecture 5.2 or, at least, prove any constant upper bound on the V2V conflict-free chromatic guarding of simple polygons.

References

1. Z. Abel, V. Alvarez, E. D. Demaine, S. P. Fekete, A. Gour, A. Hesterberg, P. Keldenich, and C. Scheffer. Three colors suffice: Conflict-free coloring of planar graphs. In SODA, pages 1951–1963. SIAM, 2017.
2. M. Abrahamsen, A. Adamaszek, and T. Miltzow. The art gallery problem is $\exists \mathbb{R}$-complete. pages 65–73, 2018.
3. P. Ashok, F. V. Fomin, S. Kolay, S. Saurabh, and M. Zehavi. Exact algorithms for terrain guarding. ACM Trans. Algorithms, 14(2):25:1–25:20, 2018.
4. A. Bärtschi, S. K. Ghosh, M. Mihaláň, T. Tschager, and P. Widmayer. Improved bounds for the conflict-free chromatic art gallery problem. Proceedings of the thirtieth annual Symposium on Computational Geometry, pages 144–153, 2014.
5. A. Bärtschi and S. Suri. Conflict-free chromatic art gallery coverage. Algorithmica, 68(1):265–283, 2014.
6. B. Ben-Moshe, M. J. Katz, and J. S. B. Mitchell. A constant-factor approximation algorithm for optimal 1.5d terrain guarding. SIAM J. Comput., 36(6):1631–1647, 2007.
7. P. Bhattacharya, S. K. Ghosh, and B. Roy. Approximability of guarding weak visibility polygons. Discrete Applied Mathematics, 228:109–129, 2017.
8. N. Biggs. Perfect codes in graphs. Journal of Combinatorial Theory, Series B, 15(3):289 – 296, 1973.
9. É. Bonnet and P. Giannopoulos. Orthogonal terrain guarding is np-complete. pages 11:1–11:15, 2018.
10. É. Bonnet and T. Miltzow. Parameterized hardness of art gallery problems. In 24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark, pages 19:1–19:17, 2016.
É. Bonnet and T. Miltzow. An approximation algorithm for the art gallery problem. pages 20:1–20:15, 2017.

Onur Çağırıcı, Petr Hliněný, and Bodhayan Roy. On colourability of polygon visibility graphs. In 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India, volume 93 of LIPIcs, pages 21:1–21:14, 2018. doi:10.4230/LIPIcs.FSTTCS.2017.21.

V. Chvátal. A combinatorial theorem in plane geometry. Journal of Combinatorial Theory, Series B, 18:39–41, 1975.

M. de Berg, O. Cheong, M. Kreveld, and M. Overmars. Computational Geometry, Algorithms and Applications. Springer-Verlag, 3rd edition, 2008.

A. Efrat and S. Har-Peled. Guarding galleries and terrains. Inf. Process. Lett., 100(6):238–245, 2006.

S. Eidenbenz, C. Stamm, and P. Widmayer. Inapproximability results for guarding polygons and terrains. Algorithmica, 31(1):79–113, 2001.

L. H. Erickson and S. M. LaValle. An art gallery approach to ensuring that landmarks are distinguishable. 2011.

S. K. Ghosh. Visibility Algorithms in the Plane. Cambridge University Press, 2007.

S. K. Ghosh. Approximation algorithms for art gallery problems in polygons. Discrete Applied Mathematics, 158:718–722, 2010.

R. K. Guy. Unsolved Problems in Number Theory. Springer-Verlag, 1994.

F. Hoffmann, K. Kriegel, S. Suri, K. Verbeek, and M. Willert. Tight bounds for conflict-free chromatic guarding of orthogonal art galleries. Comput. Geom., 73:24–34, 2018.

M. J. Katz. A PTAS for vertex guarding weakly-visible polygons - an extended abstract. CoRR, abs/1803.02160, 2018.

M. J. Katz and G. Morgenstern. Guarding orthogonal art galleries with sliding cameras. Int. J. Comput. Geometry Appl., 21(2):241–250, 2011.

J. King and D. G. Kirkpatrick. Improved approximation for guarding simple galleries from the perimeter. Discrete & Computational Geometry, 46(2):252–269, 2011.

J. Kratochvíl and M. Krivánek. On the computational complexity of codes in graphs. In Mathematical Foundations of Computer Science 1988, MFCS’88, Carlsbad, Czechoslovakia, August 29 - September 2, 1988, Proceedings, volume 324 of Lecture Notes in Computer Science, pages 396–404. Springer, 1988.

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

J. O’Rourke. Art Gallery Theorems and Algorithms. Oxford University Press, New York, 1987.

J. Pach and G. Tardos. Conflict-free colourings of graphs and hypergraphs. Combinatorics, Probability & Computing, 18(5):819–834, 2009.

S. Suri. On some link distance problems in a simple polygon. IEEE Journal on Robotics and Automation, 6:108–113, 1990.