Average Sensitivity of Geometric Algorithms

Matthijs Ebbens
University of Cologne, Germany

Yuichi Yoshida
National Institute of Informatics, Tokyo, Japan

Abstract

In modern applications of geometric algorithms, it is often unrealistic to assume that the input representation fully captures all relevant aspects of the problem, because the input data is often large and dynamic. To address this challenge, we consider the notion of average sensitivity, which is defined as the average earth mover's distance between the output distributions of the algorithm when run on an input and the same input with one point removed, where the average is over removed points and the distance between two outputs is measured using the symmetric difference size.

We start by showing that a number of classical problems from computational geometry, in particular the convex hull, Delaunay triangulation, and Voronoi diagram problems, are "simple" from the viewpoint of average sensitivity by proving tight bounds for the average sensitivity of any algorithm for these problems. Then, we continue by constructing an algorithm with low average sensitivity that computes, for any $\epsilon > 0$, a set of $(1/3 + \epsilon)n$ guards for the art gallery problem. This is the main technical contribution of this work, which combines algorithms from computational geometry with results from the theory of local computation algorithms (LCAs) and property testing.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Average Sensitivity, Convex Hull, Delaunay Triangulation, Voronoi Diagram, Art Gallery

Digital Object Identifier 10.4230/LIPIcs.ITCS.2026.53

Funding *Matthijs Ebbens*: Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 459420781.
Yuichi Yoshida: Partly supported by JSPS KAKENHI Grant Number 24K02903.

Acknowledgements We want to thank the reviewers for their comments.

1 Introduction

In modern applications of geometric algorithms, it is often unrealistic to assume that the input representation fully captures all relevant aspects of the problem, because the input data is often large and dynamic. For example, when modeling a surface using a Delaunay triangulation of a sample point set, the input sample may not include all the points necessary to capture all important features of the surface.

For this reason, it is important to design algorithms that can give a useful solution for a problem on the "real" input set by solving it on only a subset. Here, "useful" means that the output of the algorithm when applied to the subset should be close to the output of the algorithm that we would have gotten if we had access to the real input set. The notion of *average sensitivity* was introduced to formalize this type of algorithmic stability [13, 18]. Since then many algorithms with low sensitivity have been proposed for graph problems [18], dynamic programming problems [8, 9], clustering problems [15, 19], and learning problems [5].

Average sensitivity for geometric algorithms, where the input is a point set, can be defined as follows. For a point set P and a point $p \in P$, we denote by $P - p$ the set $P \setminus \{p\}$ obtained from P by deleting p. Let $\mathcal{A}$ be a deterministic algorithm that, given a point set P, outputs a set – typically consisting of points or edges – denoted as $\mathcal{A}(P)$. Then, the *average sensitivity* of $\mathcal{A}$ is the average

17th Innovations in Theoretical Computer Science Conference (ITCS 2026).
Editor: Shubhangi Saraf; Article No. 53; pp. 53:1–53:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$$\frac{1}{|P|}\sum_{p\in P}|\mathcal{A}(P)\triangle\mathcal{A}(P-p)|, \tag{1}$$

where $\triangle$ denotes the symmetric difference. One can define the average sensitivity of randomized algorithms in a similar way. For a randomized algorithm $\mathcal{A}$ we denote by $\mathcal{A}(P)$ its output distribution on P. Let $\mathrm{EMD}(\mathcal{A}(P),\mathcal{A}(P-p))$ be the earth mover's distance between $\mathcal{A}(P)$ and $\mathcal{A}(P-p)$, where the distance between two outputs is given by the symmetric difference size. More precisely, $\mathrm{EMD}(\mathcal{A}(P),\mathcal{A}(P-p)) = \inf_{\mathcal{D}}\mathbb{E}_{(x,y)\in\mathcal{D}}|x\Delta y|$, where the infimum is taken over all joint distributions over pairs of outputs of $\mathcal{A}$ whose marginals are $\mathcal{A}(P)$ and $\mathcal{A}(P-p)$. Then, the average sensitivity of $\mathcal{A}$ is given as the average

$$\frac{1}{|P|}\sum_{p\in P}\mathrm{EMD}(\mathcal{A}(P),\mathcal{A}(P-p)). \tag{2}$$

The output specification varies for each problem, and we will review it in each section. We also note that the average sensitivity with respect to the deletion of multiple points can be bounded using the average sensitivity defined above [18].

Results

In this work, we initiate a systematic study of average sensitivity of geometric algorithms. We first consider a number of classical geometric problems and show tight bounds for the average sensitivity of any algorithm to solve them.

▶ **Theorem 1.** *The average sensitivity of any algorithm that computes*

- *the convex hull of a point set in $\mathbb{R}^d$ in general position is $\Theta_d(n^{m-1})$, where $m=\lfloor d/2\rfloor$,*
- *the Delaunay triangulation of a point set in $\mathbb{R}^d$ in general position is $\Theta_d(n^{m-1})$, where $m=\lfloor (d+1)/2\rfloor$,*
- *the Voronoi diagram of a point set in $\mathbb{R}^d$ in general position is $\Theta_d(n^{m-1})$, where $m=\lfloor (d+1)/2\rfloor$.*

These problems are frequently used as subproblems in solving more complex geometric problems, and hence proving bounds for their average sensitivity is crucial for designing algorithms with low average sensitivity for more complex problems. Theorem 1 is a combination of Theorems 3 and 5 and Corollary 6, which will be discussed in Section 2.

Next, we consider the art gallery problem and construct an algorithm with low average sensitivity for computing a small set of guards.

▶ **Theorem 2.** *For any $\varepsilon>0$, there exists a randomized algorithm for the art gallery problem that, given a polygon P of n vertices, computes the allocation of at most $(1/3+\varepsilon)n$ guards that cover P with average sensitivity* $\mathrm{poly}(\varepsilon^{-1})$.

We note that $\lfloor n/3\rfloor$ guards are sometimes necessary [1], so in general this is the best one can hope for. Theorem 2 asserts that, up to a slight approximation, this can still be achieved while guaranteeing constant average sensitivity.

2 Convex Hulls, Delaunay Triangulations and Voronoi Diagrams

In this section, we consider algorithms for classical geometric problems and show that their average sensitivity is low.

2.1 Convex Hulls

Recall that a subset S of $\mathbb{R}^d$ is called *convex* if for every $p, q \in S$ the line segment between p and q is contained in S. The convex hull problem is to compute, given a set P of points in $\mathbb{R}^d$, the *convex hull* $\mathrm{conv}(P)$ of P, which is defined as the smallest convex set containing P [2, Chapter 1]. For defining the average sensitivity of algorithms for the convex hull problem, a set P in $\mathbb{R}^d$ is taken as input and the entire combinatorial structure of the convex hull is taken as output. Hence, the size of the symmetric difference between outputs is defined as the sum of the sizes of the symmetric differences of the sets of k-faces for all $k = 0, \dots, d-1$. For convenience, we assume that the input set P is in general position, i.e., no $d+1$ points lie on a common hyperplane.

▶ **Theorem 3.** *The average sensitivity of any algorithm that computes the convex hull of a point set in $\mathbb{R}^d$ in general position is $\Theta_d(n^{m-1})$, where $m = \lfloor d/2 \rfloor$.*

Proof. We start by proving the upper bound. Let P be a point set in $\mathbb{R}^d$ in general position and let $p \in P$. Let $K = \partial\,\mathrm{conv}(P)$ be the polytope bounded by the facets of $\mathrm{conv}(P)$. Because P is in general position, K is a simplicial complex. Throughout the proof, we will use the so-called Upper Bound Theorem [12], which states that the number $f_k(P)$ of k-faces of K satisfies

$$f_k(P) = O_d\big(n^m\big), \text{ and } \sum_{k=0}^{d-1} f_k(P) = \Theta_d\big(n^m\big).$$

Let $\mathrm{star}_K(p) := \{\sigma \in K \mid p \in \sigma$ be the star of p in K and let $\mathrm{link}_K(p) := \{\sigma \in K \mid p \notin \sigma, \sigma \cup \{p\} \in K\}$ be the link of p in K. To bound the average sensitivity, we separately bound the number of *disappearing* faces, i.e., faces in $\mathrm{conv}(P) \setminus \mathrm{conv}(P-p)$, and the number of *appearing* faces, i.e., the faces in $\mathrm{conv}(P-p) \setminus \mathrm{conv}(P)$.

To bound the number of disappearing faces, observe that the faces that disappear when deleting p are exactly the faces in $\mathrm{star}_K(p)$. Summing over p and double-counting by dimension, we see that

$$\sum_{p \in P} |\,\mathrm{star}_K(p)| = \sum_{k=0}^{d-1} (k+1) f_k(P) = O_d\Big(\sum_{k=0}^{d-1} f_k(P)\Big) = O_d\big(n^m\big).$$

Now, we bound the number of appearing faces. Let $R_K(p)$ denote the set of ridges in the link of p, i.e., $(d-2)$-faces of $\mathrm{link}_K(p)$. Because K is a polytopal $(d-1)$-sphere, every ridge σ of K is contained in *exactly two* facets of K. If $\sigma \in \mathrm{link}_K(p)$, then one of those two facets is $\{p\} \cup \sigma$ (which lies in $\mathrm{star}_K(p)$) and the other facet, call it σ_{out}, is disjoint from p. Deleting $\mathrm{star}_K(p)$ turns its interior into a "hole" whose boundary is precisely $\mathrm{link}_K(p)$. New facets can only appear in this hole. Moreover, because $K' := \partial\,\mathrm{conv}(P-p)$ must be a $(d-1)$-sphere again, each ridge can be associated to only one new facet. Note that one new facet could (and usually will) be associated to multiple ridges. It follows that the number of new facets can be bounded by $|R_K(p)|$. Because P is in general position, each new facet is a $(d-1)$-simplex, so the total number of the faces contained in it is

$$s_d := \sum_{j=0}^{d-1} \binom{d}{j+1} = 2^d - 1.$$

Hence, the total number of new faces of all dimensions is at most $s_d|R_K(p)|$. Summing over p and using the fact that every ridge is contained in exactly two facets (hence in the links of exactly the two vertices that complete the ridge to those facets) we obtain

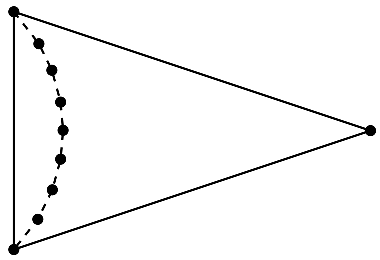

Figure 1 Assume that the number of interior vertices is $n-3$. Deleting the rightmost vertex causes $n-2$ vertex changes and n edges changes.

$$\sum_{p \in P} s_d |R_K(p)| = 2 s_d f_{d-2}(P) = O_d\big(n^m\big).$$

Because the total number of disappearing faces and appearing faces is $O_d(n^m)$, we conclude that the average sensitivity is $O_d(n^{m-1})$.

Let us now turn to the lower bound. Let P be the vertex set of a *neighborly* (e.g., cyclic) d-polytope with n vertices. Such polytopes are simplicial and satisfy $f_{d-1}(P) = \Theta_d(n^m)$ by the Upper Bound Theorem. For any vertex $p \in P$, all facets of $\partial \operatorname{conv}(P)$ that contain p must disappear after deleting p. Summing over p and dividing by n, we see that the average sensitivity is at least

$$\frac{1}{n} \sum_{p \in P} \#\{\text{facets containing } p\} = \frac{d f_{d-1}(P)}{n} = \Theta_d\big(n^{m-1}\big),$$

where the first equality follows from the fact that every facet has exactly d vertices.

This proves the matching $\Omega_d(n^{m-1})$ lower bound, and finishes the proof. ◀

▶ Remark 4. We remark that the worst-case sensitivity, i.e., the maximum size of the symmetric difference of $\mathcal{A}(P)$ and $\mathcal{A}(P-p)$ over p can be $\Omega(n)$, even when P is in $\mathbb{R}^2$. Specifically, deleting the rightmost vertex in Figure 1 causes all $n-3$ interior vertices to become extremal points. This leads to $n-2$ vertex changes and n edge changes.

2.2 Delaunay Triangulation

Next, we consider the planar Delaunay triangulation problem. Let P be a set of points in $\mathbb{R}^d$. A *triangulation* of P is a subdivision of the convex hull of P into simplices with the points of P as its vertices. A triangulation is called a *Delaunay triangulation* if the circumscribed ball of every simplex does not contain any points of P in its interior [2, Chapter 9]. The average sensitivity of an algorithm for the Delaunay triangulation problem is defined by taking as input a point set P in $\mathbb{R}^d$ and as output the set of edges in the Delaunay triangulation. Again, we assume the input point set to be in general position.

▶ **Theorem 5.** *The average sensitivity of any algorithm that computes the Delaunay triangulation of a point set in $\mathbb{R}^d$ in general position is $\Theta_d(n^{m-1})$, where $m = \lfloor (d+1)/2 \rfloor$.*

Proof. Recall that the problem of computing the Delaunay triangulation of a point set P in $\mathbb{R}^d$ can be converted to the problem of computing the convex hull of a set of points in $\mathbb{R}^{d+1}$ [2, Chapter 11]. More precisely, by lifting each point $(x_1, \ldots, x_d) \in P$ to the point

$(x_1, \ldots, x_d, x_1^2 + \ldots + x_d^2)$ on the paraboloid $x_{d+1} = x_1^2 + \ldots + x_d^2$ in $\mathbb{R}^{d+1}$, computing the convex hull of the resulting point set and projecting the lower convex hull back to $\mathbb{R}^d$, one obtains the Delaunay triangulation of P. As the average sensitivity of any algorithm that computes the convex hull in $\mathbb{R}^{d+1}$ is $\Theta_d(n^{\lfloor (d+1)/2 \rfloor - 1})$ by Theorem 3, it follows directly that the average sensitivity of any algorithm that computes the Delaunay triangulation in $\mathbb{R}^d$ is $\Theta_d(n^{\lfloor (d+1)/2 \rfloor - 1})$ as well. ◀

2.3 Voronoi Diagram

Delaunay triangulations are closely related to Voronoi diagrams [2, Chapter 7]. Given a set P of points in $\mathbb{R}^d$, the *Voronoi cell* of $p \in P$ is the set of points in $\mathbb{R}^d$ for which p is the closest point of P. The *Voronoi diagram* of P is the set of Voronoi cells for each of its points. It is well known that a Voronoi diagram is dual to the Delaunay triangulation of the same point set: vertices of the Delaunay triangulation correspond to cells in the Voronoi diagram and there is a Delaunay edge between two vertices if and only if the corresponding Voronoi cells are adjacent. The average sensitivity of an algorithm for the Voronoi diagram problem is defined by taking the set P in $\mathbb{R}^d$ as input and the set of Voronoi edges as output.

Duality between Delaunay triangulations and Voronoi diagrams allows us to conclude immediately that:

▶ **Corollary 6.** *The average sensitivity of any algorithm that computes the Voronoi diagram of a point set in $\mathbb{R}^d$ in general position is $\Theta_d(n^{m-1})$, where $m = \lfloor (d+1)/2 \rfloor$.*

3 Art Gallery Problem

In this section, we consider the art gallery problem [14, 2]. Defining the average sensitivity for the art gallery problem requires some technical care, which we defer to Section 3.1. The goal of this section is to show Theorem 2, which we restate for convenience:

▶ **Theorem 7.** *For any $\varepsilon > 0$, there exists an algorithm for the art gallery problem that, given a polygon P of n vertices, computes the allocation of at most $(1/3 + \varepsilon)n$ guards that cover P with average sensitivity* $\mathrm{poly}(\varepsilon^{-1})$.

A standard algorithm for the art gallery problem that finds $\lfloor n/3 \rfloor$ guards covering the entire polygon is as follows: first, triangulate the polygon using one of many known algorithms [3, 4, 17]. This gives a 3-colorable triangulated planar graph. Second, color the vertices of the triangulation with three colors [7]. Finally, take as set of guards all vertices of the color that occurs least. Since all triangles have exactly one vertex of the chosen color, the entire polygon is guarded.

In Section 3.2, we show that a standard triangulation algorithm has a low average sensitivity. Then in Section 3.3, we design an algorithm with low average sensitivity that finds a set of guards of size $(1/3 + \varepsilon)n$. To compute the set of guards, we first partition the triangulated graph into constant-sized parts by deleting a small fraction of edges. This can be done by applying an algorithm converted from a local computation algorithm (LCA) for the partition problem [6, 16] using the general transformation given in [18]. Then, we compute a 3-coloring for each part using an arbitrary deterministic algorithm. With a special care of the vertices incident to the deleted edges, we can compute a set of guards of size $(1/3 + \varepsilon)n$ from the 3-coloring.

In this section, we will use bold symbols to denote random variables. Also for a positive integer n, let $[n]$ denote the set $\{1, 2, \ldots, n\}$.

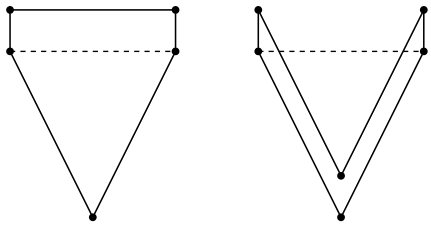

Figure 2 Left: removing the bottommost vertex introduces the dashed edge. The resulting collection of edges and vertices is still a valid polygon. Right: removing the bottommost vertex introduces the dashed edge which crosses two of the existing edges. The resulting collection of vertices and edges is no longer a polygon.

3.1 Average Sensitivity

In this section, we formally define the average sensitivity for the art gallery problem.

Usually, a *polygon* P is defined as a collection of n vertices $v_1, \ldots, v_n$ and n edges $v_1v_2, \ldots, v_nv_1$ such that no pair of non-consecutive edges intersect. Here, however, we define a polygon as the closed finite region of the plane bounded by these vertices and edges, following [2, Chapter 3]. We say that a point $p \in P$ *sees* or *covers* or *guards* a point $y \in P$ if the line segment xy is completely contained in P. The art gallery problem is to determine the minimum number of "guards" that cover P entirely. In other words, we want to find a set S of points in P of minimum size such that any point in P is covered by at least one point of S.

The average sensitivity of an algorithm for the art gallery problem is defined by taking the polygon P as input and the set of guards as output. In this case, we have to be careful for two reasons. First, the polygon $P - v_i$ obtained after deleting v_i from P should not be understood as the setwise deletion $P \setminus \{v_i\}$, but as the polygon with vertices $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n$ and edges $v_1v_2, \ldots, v_{i-1}v_{i+1}, \ldots, v_nv_1$. Second, P might contain some vertices that we cannot remove without introducing crossing edges (see Figure 2). This means that the collection of vertices and edges obtained after deleting such a vertex is no longer a polygon. Since it does not make sense to consider the art gallery problem on such collections of vertices and edges, we will only consider the removal of vertices for which the result is again a polygon. We call such vertices *admissible*. For all other vertices, we set the symmetric difference in Equation (1) or the earth mover's distance in Equation (2) to be equal to zero.

3.2 Polygon Triangulation

To start our construction of a stable-on-average algorithm for the art gallery problem, we describe an algorithm for computing a triangulation of a polygon from the literature [2, Chapter 3]. This algorithm already has a low average sensitivity, as we will show in Lemma 8, so we can use it as it is.

Many algorithms for triangulating a polygon subdivide the polygon into pieces which can be triangulated more easily [2, 3, 4, 17]. Here, the easier to triangulate pieces are typically *monotone*. A polygon is called *monotone with respect to a line* ℓ if its intersection with any line orthogonal to ℓ is connected. In particular, a polygon is called y*-monotone* if it is monotone with respect to the y-axis (see Figure 3).

To triangulate a y-monotone polygon P, we consider the vertices in order of decreasing y-coordinate [2, Chapter 3.3]. We keep a stack S of vertices that have been encountered but may still need more diagonals, ordered from lowest y-coordinate (top of the stack) to highest

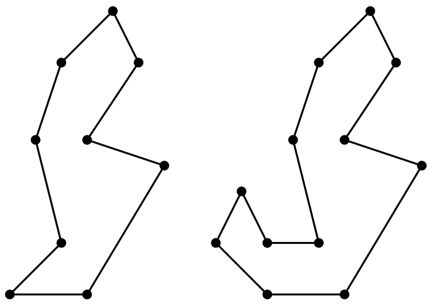

Figure 3 Left: a y-monotone polygon. Right: not a y-monotone polygon.

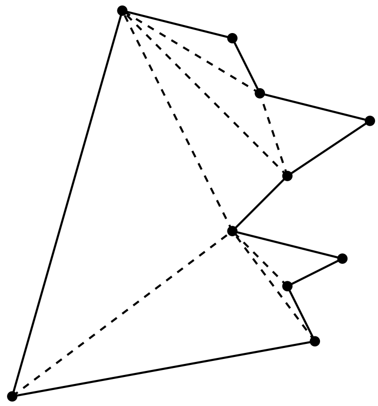

Figure 4 Triangulation of a y-monotone polygon, in which diagonals are shown dashed.

(bottom of the stack). Every time we encounter a new vertex v, we add diagonals from v to elements in S, starting at the top of the stack and continuing until we can add no more. These diagonals split of triangles and we update S accordingly (see Figure 4). For further details, we refer to the literature [2, Chapter 3.3].

To subdivide a polygon into y-monotone pieces, we use a so-called *trapezoidation* of the polygon [3, 17]. Assume that our polygon P is embedded in $\mathbb{R}^2$, so that we can use intuitive notions like left, right, horizontal, vertical and so on. Starting from each vertex of P, we draw two horizontal rays, one towards the left and one towards the right, each extending until it hits an edge of P. If a ray never hits an edge of P we just extend it indefinitely. For a vertex v we call the union of these two horizontal rays the *horizontal extension* of v. The polygon P together with the union of the horizontal extensions of its vertices forms a plane graph, which we call the *trapezoidation* of P. All faces of the trapezoidation have two horizontal sides, so we can indeed call them *trapezoids*. From the trapezoidation we obtain a subdivision of the polygon into y-monotone pieces by removing all trapezoids that do not lie in the interior of the polygon and adding diagonals between any two vertices within a trapezoid that do not lie on the same side. We note that each of the y-monotone pieces resulting from this subdivision has a special form: its boundary consists of two y-monotone polygonal chains, one of which is a single edge.

▶ **Lemma 8.** *The average sensitivity of the above algorithm for computing a triangulation of a given polygon is* $O(1)$.

Proof. Denote the above algorithm by $\mathcal{A}$ and let P be a polygon. Consider any diagonal e in $\mathcal{A}(P)$. We will show that e is not a diagonal of $\mathcal{A}(P - v)$ for $O(1)$ vertices v of P.

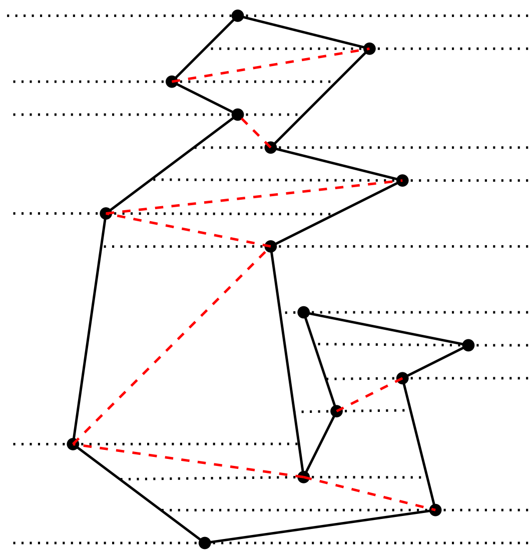

Figure 5 Subdividing a polygon into y-monotone pieces: the horizontal extensions are indicated by dotted line segments, diagonals between vertices on different sides of a trapezoid are indicated by red, dashed line segments.

First, assume that e is contained in the interior of the monotone polygon M computed in the course of running $\mathcal{A}$, i.e., it is not one of the diagonals that separates two monotone polygons. Deleting a vertex in $P \setminus M$ does not affect M, so it does not affect e either. Therefore, we only need to consider deletion of the vertices of M. Denote the vertices of M by $v_1, \ldots, v_k$ in order of decreasing y-coordinate and let $e = (v_i, v_j)$ for some $1 \leq i < j \leq k$. We claim that e is a diagonal of $\mathcal{A}(P - v_\ell)$ for $\ell \notin \{1, i, j-1, j, k\}$.

Consider any vertex $v_\ell \in M$. Clearly, deleting v_i or v_j makes it impossible for (v_i, v_j) to exist, so assume henceforth that $\ell \notin \{i, j\}$. We note that by the triangulation algorithm for monotone polygons, e is a diagonal of $\mathcal{A}(P)$ if and only if e is a valid diagonal of P, i.e., it does not cross any side of P, and v_i is in the stack $S(v_j)$ when handling v_j. This means that e is *not* a diagonal of $\mathcal{A}(P - v_\ell)$ if and only if e is not a valid diagonal of $P - v_\ell$ or v_i is not in the stack $S_\ell(v_j)$ when handling v_j. Here, $S(v_j)$ denotes the stack when handling v_j in the process of triangulating M, whereas $S_\ell(v_j)$ denotes the stack when handling v_j in the process of triangulating the monotone polygon M_ℓ containing v_i and v_j after deleting v_ℓ.

- Assume that e is not a valid diagonal of $P - v_\ell$, i.e., e crosses a side of $P - v_\ell$ or is itself a side of $P - v_\ell$. Since e is a valid diagonal of P, clearly e is also a valid diagonal of $P - v_\ell$ if $P \subseteq P - v_\ell$. Therefore, $P - v_\ell \subsetneq P$. We observe that $P \setminus (P - v_\ell)$ is a triangle (see Figure 6). The only possibly affected diagonal is $(v_{\ell-1}, v_{\ell+1})$, which is a diagonal in P, but a side of $P - v_\ell$. Hence, in this case $\ell = j - 1$.
- Assume that v_i is not in $S_\ell(v_j)$. Let v_h be the element in $S(v_j)$ below v_i, i.e., the element to which a diagonal is drawn *after* all diagonals to v_i are drawn. The element v_i is removed from the stack as soon as a diagonal is drawn to v_h. The fact that v_i is not in $S_\ell(v_j)$ implies that there exists $v \in M_\ell$ which is handled before v_j and from which there is a valid diagonal to v_i and v_h.
  It is straightforward to see that v is not some vertex of M, since otherwise v_i would also have been removed from the stack before handling v_j when triangulating M. Hence, v is a vertex in $M_\ell \setminus M$. When deleting v_ℓ from M, only the sides $(v_{\ell-1}, v_\ell)$ and $(v_\ell, v_{\ell+1})$

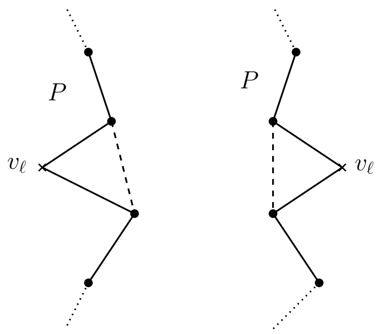

Figure 6 Deleting vertex v_ℓ removes the two sides having v_ℓ as an endpoint and introduces the dashed side $(v_{\ell-1}, v_{\ell+1})$. It is assumed that the inside of P is to the left of the drawn sides. Left: $P \subsetneq P - v_\ell$. Right: $P - v_\ell \subsetneq P$.

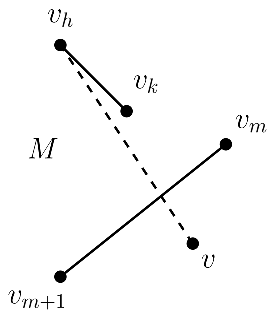

Figure 7 Diagonal (v, v_h) of M_ℓ crossing a side (v_m, v_{m+1}) of M implies that one of (v_m, v_h) and (v_{m+1}, v_h) is a diagonal of M (in this case, (v_{m+1}, v_h) is a diagonal).

(counting modulo k) are removed; the remaining sides of M are also sides of M_ℓ. Therefore, $M_\ell \setminus M$ only contains vertices with y-coordinate between the y-coordinates of $v_{\ell-1}$ and $v_{\ell+1}$. It follows that $\ell = 1$ or $\ell = k$ or $i < \ell < j$. Since the cases $\ell = 1$ and $\ell = k$ are already mentioned in the claim, we continue by assuming that $i < \ell < j$.
Since v is not contained inside M, the diagonal (v, v_h) in M_ℓ crosses some side of M, say (v_m, v_{m+1}) with $i \leq m \leq j - 1$ (see Figure 7). This means that at least one of (v_m, v_h) and (v_{m+1}, v_h) is a valid diagonal of M. However, (v_m, v_h) is not a diagonal of M for $i \leq m \leq j - 1$. Therefore, the only possibility is that (v_{m+1}, v_h) is a valid diagonal with $m = j - 1$, i.e., diagonal (v, v_h) in M_ℓ crosses side (v_{j-1}, v_j) of M. Hence, (v, v_h) is only a valid diagonal of M_ℓ if side (v_{j-1}, v_j) is removed, which happens exactly when $\ell = j - 1$. Recall that we already treated the case $\ell = j$ at the start.

Second, assume that e is a diagonal that separates two monotone polygons, say M_1 and M_2. It can be shown by a similar reasoning as before that $e \notin \mathcal{A}(P - v)$ for only a constant number of vertices v in each of M_1 and M_2. Moreover, deletion of a vertex in $P \setminus (M_1 \cup M_2)$ does not affect e. Hence, the total number of vertices v for which $e \notin \mathcal{A}(P - v)$ is $O(1)$, which is what we wanted to prove. ◀

3.3 Set of guards on 3-Colorable Triangulated Planar Graphs

Let $G = (V, E)$ be the graph obtained by triangulating the input polygon, where V consists of the vertices and E consists of the edges in the triangulation. Note that G is a 3-colorable planar graph. Standard algorithms for finding a set of at most $\lfloor n/3 \rfloor$ guards in G first compute a 3-coloring of G and then place guards on the vertices with the least occurring color [2, 7]. However, these algorithms do not have low average sensitivity, as the next example illustrates. We emphasize that throughout this example we assume that polygons are triangulated using the algorithm described in Section 3.2 and that a 3-coloring is computed using either of [2, 7]. Therefore, the lower bound for the average sensitivity might not necessarily hold when another triangulation algorithm is used, or, more generally, when a different algorithm for finding a set of at most $\lfloor n/3 \rfloor$ guards in P is used (for example one that does not rely on computing triangulations).

► **Example 9.** Consider the parallelogram containing $3n$ vertices on each of its long sides as in Figure 8. The dashed lines indicate that the pattern of 3 vertices left and right repeats in the middle and the dotted lines represent diagonals of the triangulation. The letters R (red), B (blue) and G (green) indicate the 3-coloring of the triangulation. Deleting a vertex causes the triangulation to change, which also changes the (deterministic) 3-coloring of the vertices and the resulting set of guards. We will show that the average sensitivity of the guard set is $\Omega(n)$. First, observe that in the 3-coloring before deletion of a vertex the number of red, blue and green vertices is equal. Therefore, the algorithm could have chosen any of these sets as the set of guards:

- Assume that P is guarded by the set of blue vertices. It is straightforward to show that deleting a green vertex v changes the color of all red vertices below v to green and vice versa (the blue vertices stay blue). Moreover, after deletion of v the set of green vertices becomes the smallest. As the set of guards changes from the set of blue vertices to the set of green vertices (which were originally red), the symmetric difference of the guard sets before and after deletion of v is proportional to the number of blue and red vertices below v in the original triangulation. Applying this argument to the deletion of any of the $\Omega(n)$ green vertices in the top half of the polygon leads to the desired $\Omega(n)$ bound for the average sensitivity.
- If P is guarded by the set of red vertices, we can use the same argument, but now we consider the deletion of blue vertices.
- If P is guarded by the set of green vertices, we can use the same argument. In this case, deleting vertices of any color works.

Hence, no matter which color was used for the set of initial guards, the average sensitivity is $\Omega(n)$, which is what we wanted to show.

Instead, we design in this section an algorithm for finding a set of guards in G of size arbitrarily close to $\lfloor n/3 \rfloor$ that *does* have a low average sensitivity:

► **Lemma 10.** *For any $\varepsilon > 0$, there exists a polynomial-time algorithm that, given a 3-colorable triangulated planar graph, outputs a set of guards of size at most $(1/3 + \varepsilon)n$ with average sensitivity* $\mathrm{poly}(\varepsilon^{-1})$.

Then, we can prove our main theorem of this section:

Proof of Theorem 7. The claim follows by combining Lemmas 8 and 10. ◄

In Section 3.3.1, we show that we can find a partition of a planar graph into constant-sized parts with low average sensitivity using the connection between local computation algorithms (LCAs) [16] and average sensitivity [18]. Then in Section 3.3.2, we prove Lemma 10 using the partition algorithm.

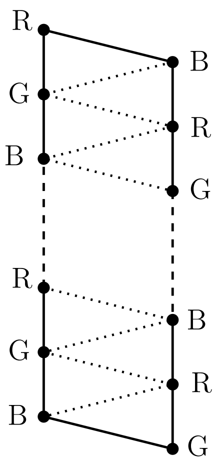

Figure 8 Polygon with $6n$ vertices, for which the standard algorithms for finding a set of guards have a high average sensitivity.

3.3.1 Graph Partitioning

We first design a stable-on-average algorithm that partitions a bounded-degree planar graph into small subsets by deleting a small number of edges crossing different subsets. To this end, we use a black-box transformation from local computation algorithms (LCAs) to stable-on-average algorithms due to [18]. The goal of an LCA for a graph problem is to, given query access to the input graph $G = (V, E)$, provide query access to a feasible solution for the problem. More formally, we assume that the input graph $G = (V, E)$ on n vertices is represented as adjacency lists, and we can access it via neighbor queries, where each query is of the form (v, i) for $v \in V$ and $i \in [n]$ and the answer is the i-th neighbor of vertex v in its adjacency list (and a special symbol if i is larger than the degree of v). Then, an LCA is defined as follows:

▶ **Definition 11** (Local Computation Algorithm (LCA) [16]). *Consider a graph problem $\mathcal{P}$, where the output to the problem is an integer labeling on vertices. Let $\delta : \mathbb{N} \to [0, 1]$ and $q, r : \mathbb{N} \to \mathbb{N}$. A (q, r, δ)-LCA for $\mathcal{P}$ is an algorithm $\mathcal{L}$ that, given query access to a graph $G = (V, E)$ on n vertices, first generates a random string $\boldsymbol{\pi} \in \{0, 1\}^{r(n)}$, and satisfies:*

- *given an input $v \in V$, the algorithm $\mathcal{L}$ makes queries to G and answers the label of v, and*
- *the answers of $\mathcal{L}$ to all possible input vertices are consistent with a single feasible labeling to $\mathcal{P}$ on G.*

For every graph G, the probability (over the choice of random string) that there exists an input vertex for which $\mathcal{L}$ makes more than $q(n)$ queries is at most $\delta(n)$.

For two integer labelings $f, \tilde{f} : V \to \mathbb{N}$, we define their distance as

$$d(f, \tilde{f}) := \#\{v \in V : f(v) \neq \tilde{f}(v)\}.$$

Then, we can naturally define the average sensitivity of an algorithm that outputs an integer labeling using this distance. It is known that we can obtain a stable-on-average algorithm from an LCA as follows:

▶ **Lemma 12** ([18]). *Consider a graph problem $\mathcal{P}$, where the output to the problem is an integer labeling on vertices.*[1] *Let $\delta : \mathbb{N} \to [0,1]$ and $q, r : \mathbb{N} \to \mathbb{N}$. If $\mathcal{P}$ has a (q, r, δ)-LCA $\mathcal{L}$, then there exists an algorithm $\mathcal{A}$ for $\mathcal{P}$, that on input $G = (V, E)$ on n vertices, has an average sensitivity of at most $q(n) + n\delta(n)$.*

To obtain a stable-on-average algorithm for graph partitioning, we will instantiate Lemma 12 with an LCA for graph partitioning, which is known as a partition oracle in the property testing literature. We first note that an integer labeling $f : V \to \mathbb{N}$ can be seen as a partition $\{f^{-1}(i)\}_{i \in \mathbb{N}}$. Then, a partition oracle is defined as follows:

▶ **Definition 13** (Partition oracle [6]). *Let $\mathcal{G}$ be a family of graphs with degree bound d and let $T : (0,1) \to \mathbb{N}$ be a function. An LCA $\mathcal{L}$ is called an $(\varepsilon, T(\varepsilon))$-partition oracle for $\mathcal{G}$ if it satisfies the following properties. Let $\boldsymbol{f} : V \to \mathbb{N}$ be the output integer labeling of $\mathcal{L}$. Note that $\boldsymbol{f}$ is a random variable.*

- *(Consistency) For each $i \in \mathbb{N}$, $\boldsymbol{f}^{-1}(i)$ is empty or an induced connected subgraph, and the subsets $\{\boldsymbol{f}^{-1}(i)\}_{i \in \mathbb{N}}$ form a partition of V.*
- *(Cut bound) With probability at least 2/3 (over the internal randomness $\boldsymbol{\pi}$ of $\mathcal{L}$), the number of edges between different sets in $\{\boldsymbol{f}^{-1}(i)\}_{i \in \mathbb{N}}$ is at most εdn.*
- *(Running time) For every $v \in V$, $\mathcal{L}(v)$ runs in time $T(\varepsilon)$.*

Note that the running time $T(\varepsilon)$ is clearly an upper bound on the size of the set $\boldsymbol{f}^{-1}(i)$ for any $i \in \mathbb{N}$.

▶ **Lemma 14** ([11]). *Let $d \geq 1$ be an integer and let $\mathcal{G}$ be the set of planar graphs with maximum degree at most d.*[2] *Then, there is an $(\varepsilon, \mathrm{poly}(\varepsilon^{-1}d))$-partition oracle for $\mathcal{P}$.*

Combining Lemmas 12 and 14, we obtain the following:

▶ **Corollary 15.** *Let $\varepsilon > 0$. There exists a randomized polynomial-time algorithm $\mathcal{L}$ that, given a planar graph $G = (V, E)$ of maximum degree at most d, outputs a function $\boldsymbol{f} : V \to \mathbb{N}$ with the following properties.*

- *(Consistency) For each $i \in \mathbb{N}$, $\boldsymbol{f}^{-1}(i)$ is empty or an induced connected subgraph, and the subsets $\{\boldsymbol{f}^{-1}(i)\}_{i \in \mathbb{N}}$ form a partition of V.*
- *(Size bound) For each $i \in \mathbb{N}$, we have $|f^{-1}(i)| = O(\mathrm{poly}(\varepsilon^{-1}d))$.*
- *(Cut bound) With probability at least 2/3 (over the internal randomness $\boldsymbol{\pi}$ of $\mathcal{L}$), the number of edges between different sets in $\{\boldsymbol{f}^{-1}(i)\}_{i \in \mathbb{N}}$ is at most εdn.*
- *The average sensitivity of $\mathcal{L}$ is $O(\mathrm{poly}(\varepsilon^{-1}d))$.*

3.3.2 Algorithm

We describe our algorithm for computing a set of guards on a 3-colorable planar graph. We first remove high-degree vertices from the input graph G and let G' be the resulting graph. We add all the removed vertices to the output set $\boldsymbol{S}$ of guards. Then, we apply Corollary 15 to $\tilde{G}$ and let $\boldsymbol{f} : V \to \mathbb{N}$ be the obtained integer labeling, which we regard as a partition of V. For notational convenience, let $\mathcal{P}_{\boldsymbol{f}} = (\boldsymbol{V}_1, \ldots, \boldsymbol{V}_{\boldsymbol{k}})$ be the partition induced by $\boldsymbol{f}$. We note that each $\boldsymbol{V}_i$ is 3-colorable and triangulated. Let $\boldsymbol{F}$ be the set of edges connecting different parts in $\mathcal{P}_{\boldsymbol{f}}$. Then, we add $V(\boldsymbol{F})$, i.e., the set of endpoints of edges in $\boldsymbol{F}$ to $\boldsymbol{S}$. Then for each $i \in [\boldsymbol{k}]$, we compute an arbitrary 3-coloring $\boldsymbol{\varphi}_i : \boldsymbol{V}_i \to \{1, 2, 3\}$ of the subgraph induced by $\boldsymbol{V}_i$ using a deterministic algorithm. Let $\boldsymbol{c}_i^* = \arg\min_{c=1,2,3} |\boldsymbol{\varphi}_i^{-1}(c)|$ be the color that occurs least, and then add $\boldsymbol{S}_i := \boldsymbol{\varphi}_i^{-1}(\boldsymbol{c}_i^*)$ to the $\boldsymbol{S}$. The details are given in Algorithm 1.

---

[1] Originally, this result was shown for graph problems where the output is a vertex set or an edge set. However, the proof can be easily generalized to our setting.

[2] Indeed, the claim holds for any minor-closed family of graphs.

Algorithm 1 Algorithm for computing a set of guards on a 3-colorable planar graph.

Input: $G = (V, E)$, $\varepsilon > 0$

$d \leftarrow 100\varepsilon^{-1}$, $H \leftarrow \{v \in V : d(v) \geq d\}$, $G' \leftarrow G[V \setminus H]$;

Apply Corollary 15 on G' with $\varepsilon_{\mathrm{PO}} := \varepsilon/10d$;

Let $\boldsymbol{f} : V \to \mathbb{N}$ be the obtained integer labeling, and let $\mathcal{P}_{\boldsymbol{f}} = (\boldsymbol{V}_1, \ldots, \boldsymbol{V}_{\boldsymbol{k}})$ be the partition of V induced by $\boldsymbol{f}$;

$\boldsymbol{F} \leftarrow$ the set of edges in G' between different parts in $\mathcal{P}_{\boldsymbol{f}}$;

$\boldsymbol{S} \leftarrow H \cup V(\boldsymbol{F})$;

for $i = 1$ *to* $\boldsymbol{k}$ **do**

  Compute a 3-coloring $\boldsymbol{\phi}_i : \boldsymbol{V}_i \to \{1, 2, 3\}$ of $G[\boldsymbol{V}_i]$ using a deterministic algorithm;

  $\boldsymbol{c}_i^* \leftarrow \arg\min_{c=1,2,3} |\boldsymbol{\phi}_i^{-1}(c)|$ and $\boldsymbol{S}_i \leftarrow \boldsymbol{\phi}_i^{-1}(\boldsymbol{c}_i^*)$;

  $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \boldsymbol{S}_i$.

return $\boldsymbol{S}$

First, we show that the output is a valid set of guards and analyze its size.

▶ **Lemma 16.** *Algorithm 1 returns a set of guards of size* $(1/3 + \varepsilon)n$ *with probability at least* $2/3$.

Proof. We first show that the output vertex set $\boldsymbol{S}$ is a valid set of guards by showing that every triangle in G has at least one vertex in $\boldsymbol{S}$. Consider a triangle uvw in G, where $u, v, w \in V$.

- If one of u, v, w is in H, then clearly at least one of them is in $\boldsymbol{S}$, because $H \subseteq \boldsymbol{S}$.
- Else if two of them, say, u and v, belong to different parts in $\mathcal{P}_{\boldsymbol{f}}$, then $u, v \in \boldsymbol{S}$, because $(u, v) \in \boldsymbol{F}$ and $V(\boldsymbol{F}) \subseteq \boldsymbol{S}$.
- Else, i.e., all of them belong to the same part, say, $\boldsymbol{V}_i$, then the color class $\phi_i^{-1}(c_i^*)$ must include one of them, and hence one of them is contained in $\boldsymbol{S}$.

Next, we analyze the size of $\boldsymbol{S}$. Note that G is planar, and hence the average degree is at most 6. Hence, by Markov's inequality, the number of vertices with degree more than $d = 100\varepsilon^{-1}$ is at most $6n/d = 6/100 \cdot \varepsilon n \leq \varepsilon n/10$. Hence, we have $|H| \leq \varepsilon n/10$.

Note that G' has maximum degree d. Hence by Corollary 15, with probability at least 2/3, we have $|\boldsymbol{F}| \leq \varepsilon_{\mathrm{PO}} dn = \varepsilon n/10$. Hence, we have $V(\boldsymbol{F}) \leq \varepsilon n/5$ with probability at least 2/3.

Also each $\boldsymbol{S}_i$ has size at most $|\boldsymbol{V}_i|/3$. Then by a union bound, we have

$$
\begin{aligned}
|\boldsymbol{S}| &\leq |H| + |\boldsymbol{F}| + \sum_{i \in [\boldsymbol{k}]} |\boldsymbol{S}_i| \leq \frac{\varepsilon n}{10} + \frac{\varepsilon n}{5} + \sum_{i \in [\boldsymbol{k}]} \frac{|\boldsymbol{V}_i|}{3} \\
&\leq \frac{\varepsilon n}{10} + \frac{\varepsilon n}{5} + \frac{n}{3} \leq \left(\frac{1}{3} + \varepsilon\right) n.
\end{aligned}
$$ ◀

To analyze the average sensitivity of the output set, it is convenient to define a distance notion for partitions. For two partitions $\mathcal{P} = \{S_1, \ldots, S_k\}$ and $\tilde{\mathcal{P}} = \{\tilde{S}_1, \ldots, \tilde{S}_{\tilde{k}}\}$ with $k \leq \tilde{k}$, we define the distance between them as

$$
d(\mathcal{P}, \tilde{\mathcal{P}}) := \min_{\pi} \sum_{i=1}^{k} \mathbf{1}[S_i \neq \tilde{S}_i] \cdot (|S_i| + |\tilde{S}_i|) + \sum_{i=k+1}^{\tilde{k}} |\tilde{S}_i|,
$$

where the minimum is over all bijections $\pi : [k] \to [k]$. Intuitively speaking, if a part in $\mathcal{P}$ does not exist in $\tilde{\mathcal{P}}$, then it contributes to the distance by its size. When $k > \tilde{k}$, we define $d(\mathcal{P}, \tilde{\mathcal{P}}) := d(\tilde{\mathcal{P}}, \mathcal{P})$. Then, we can naturally define the earth mover's distance $\mathrm{EMD}(\boldsymbol{\mathcal{P}}, \tilde{\boldsymbol{\mathcal{P}}})$ between random partitions $\boldsymbol{\mathcal{P}}$ and $\tilde{\boldsymbol{\mathcal{P}}}$. For two functions $f, \tilde{f} : V \to \mathbb{N}$, it is easy to observe that

$$d(\mathcal{P}_f, \mathcal{P}_{\tilde{f}}) \leq d(f, \tilde{f}) \cdot \max_{i \in \mathbb{N}} \left\{ |f^{-1}(i)| + |\tilde{f}^{-1}(i)| \right\} \tag{3}$$

because each $v \in V$ with $f(v) \neq \tilde{f}(v)$ contributes to the LHS by $|f^{-1}(v)| + |\tilde{f}^{-1}(v)|$. A similar inequality holds for the earth mover's distance.

▶ **Lemma 17.** *The average sensitivity of Algorithm 1 is* $\mathrm{poly}(\varepsilon^{-1})$.

Proof. Let $G = (V, E)$ be a graph, and let $G^{(e)} = (V, E - e)$ for $e \in E$. We use the superscript $^{(e)}$ to denote the variables when Algorithm 1 is applied to to $G^{(e)}$.

We first note that $|H \triangle H^{(e)}| \leq 1$. Also, G' and $(G')^{(e)}$ differ by at most one vertex of degree d, which implies they differ by at most d edges. By the average sensitivity bound of Lemma 14, we have

$$\frac{1}{m} \sum_{e \in E} \mathrm{EMD}(\boldsymbol{f}, \boldsymbol{f}^{(e)}) \leq \mathrm{poly}(\varepsilon_{\mathrm{PO}}^{-1} d) \cdot d = \mathrm{poly}(\varepsilon^{-1}).$$

This implies that

$$\frac{1}{m} \sum_{e \in E} \mathrm{EMD}(V(\boldsymbol{F}), V(\boldsymbol{F}^{(e)})) = \mathrm{poly}(\varepsilon^{-1}) \cdot d = \mathrm{poly}(\varepsilon^{-1}).$$

Next, the average sensitivity of the set $\bigcup_{i=1}^{k} \boldsymbol{S}_i$ is bounded as

$$\begin{aligned}
&\frac{1}{m} \sum_{e \in E} \mathrm{EMD}\left(\bigcup_{i=1}^{\boldsymbol{k}} \boldsymbol{S}_i, \bigcup_{i=1}^{\boldsymbol{k}^{(e)}} \boldsymbol{S}_i^{(e)} \right) \leq \frac{1}{m} \sum_{e \in E} \mathrm{EMD}(\mathcal{P}_{\boldsymbol{f}}, \mathcal{P}_{\boldsymbol{f}^{(e)}}) \\
&\leq \frac{1}{m} \sum_{e \in E} \mathrm{EMD}(\boldsymbol{f}, \boldsymbol{f}^{(e)}) \cdot \max_{i \in \mathbb{N}} \left\{ |\boldsymbol{f}^{-1}(i)| + |(\boldsymbol{f}^{(e)})^{-1}(i)| \right\} && \text{(by (3))} \\
&\leq \frac{1}{m} \sum_{e \in E} \mathrm{EMD}(\boldsymbol{f}, \boldsymbol{f}^{(e)}) \cdot \mathrm{poly}(\varepsilon^{-1} d) && \text{(by the size bound of Corollary 15)} \\
&= \mathrm{poly}(\varepsilon^{-1} d) \cdot \mathrm{poly}(\varepsilon^{-1} d) && \text{(by the average sensitivity bound of Corollary 15)} \\
&= \mathrm{poly}(\varepsilon^{-1}).
\end{aligned}$$

Recalling that $\boldsymbol{S} = H \cup V(\boldsymbol{F}) \cup \bigcup_{i=1}^{k} \boldsymbol{S}_i$, we conclude that

$$\frac{1}{m} \sum_{e \in E} \mathrm{EMD}(\boldsymbol{S}, \boldsymbol{S}^{(e)}) = \mathrm{poly}(\varepsilon^{-1}).$$ ◀

4 Future work

As, to our knowledge, this is the first work to explore average sensitivity in a geometric setting, there remain many possible directions for future research. First, one could look at other classical geometric problems, such as computing the Fréchet distance of two polygonal curves or computing the minimum enclosing disk of a point set. In both cases, it is not directly clear how the average sensitivity should be defined, as the output is a real number (the Fréchet distance) or a point (the center of the minimum enclosing disk). For the Fréchet distance, one could instead output the matching between the vertices of the two curves and define the average sensitivity via a suitable distance on matchings. For the minimum enclosing disk, it might be possible to use the recent framework of Lipschitz continuity of algorithms [10] to quantify how much a solution is affected by deleting a part of the input.

Moreover, average sensitivity could be studied for geometric search structures, such as range trees or quadtrees. Although it is not obvious how to define average sensitivity, it seems natural to use the tree-edit distance for tree-based data structures.

Finally, the concept of average sensitivity could possibly be applied to motion planning. For example, in robot reconfiguration it is assumed that the location and movement of all individual robots is known, which is not always true in practice. To account for displaced or broken robots, it might be useful to apply an algorithm with low average sensitivity.

References

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

2 Mark De Berg, Mark Van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational geometry: algorithms and applications*. Springer Science & Business Media, 2000.

3 Alain Fournier and Delfin Y Montuno. Triangulating simple polygons and equivalent problems. *ACM Transactions on Graphics*, 3(2):153–174, 1984. `doi:10.1145/357337.357341`.

4 Michael R Garey, David S Johnson, Franco P Preparata, and Robert E Tarjan. Triangulating a simple polygon. *Information Processing Letters*, 7(4):175–179, 1978. `doi:10.1016/0020-0190(78)90062-5`.

5 Satoshi Hara and Yuichi Yoshida. Average sensitivity of decision tree learning. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023.

6 Avinatan Hassidim, Jonathan A Kelner, Huy N Nguyen, and Krzysztof Onak. Local graph partitions for approximation and testing. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 22–31, 2009. `doi:10.1109/FOCS.2009.77`.

7 Ali A Kooshesh and Bernard ME Moret. Three-coloring the vertices of a triangulated simple polygon. *Pattern Recognit.*, 25(4):443, 1992. `doi:10.1016/0031-3203(92)90093-X`.

8 Soh Kumabe and Yuichi Yoshida. Average sensitivity of dynamic programming. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1925–1961, 2022. `doi:10.1137/1.9781611977073.77`.

9 Soh Kumabe and Yuichi Yoshida. Average sensitivity of the knapsack problem. In *30th Annual European Symposium on Algorithms (ESA)*, pages 75:1–75:14, 2022. `doi:10.4230/LIPIcs.ESA.2022.75`.

10 Soh Kumabe and Yuichi Yoshida. Lipschitz continuous algorithms for graph problems. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 762–797. IEEE, 2023. `doi:10.1109/FOCS57990.2023.00051`.

11 Akash Kumar, C Seshadhri, and Andrew Stolman. Random walks and forbidden minors III: poly($d\varepsilon^{-1}$)-time partition oracles for minor-free graph classes. In *Proceedings of the IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 257–268, 2022.

12 Peter McMullen. On the upper-bound conjecture for convex polytopes. *Journal of Combinatorial Theory, Series B*, 10(3):187–200, 1971.

13 Shogo Murai and Yuichi Yoshida. Sensitivity analysis of centralities on unweighted networks. In *The World Wide Web conference (WWW)*, pages 1332–1342, 2019. `doi:10.1145/3308558.3313422`.

14 Joseph O'Rourke. *Art gallery theorems and algorithms*, volume 57. Oxford University Press Oxford, 1987.

15 Pan Peng and Yuichi Yoshida. Average sensitivity of spectral clustering. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1132–1140, 2020. doi:10.1145/3394486.3403166.

16 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *Proceedings of the 1st Symposium on Innovations in Computer Science (ICS)*, pages 223–238, 2011. URL: http://conference.iiis.tsinghua.edu.cn/ICS2011/content/papers/36.html.

17 Raimund Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry*, 1(1):51–64, 1991. doi:10.1016/0925-7721(91)90012-4.

18 Nithin Varma and Yuichi Yoshida. Average sensitivity of graph algorithms. *SIAM Journal on Computing*, 52(4):1039–1081, 2023. doi:10.1137/21M1399592.

19 Yuichi Yoshida and Shinji Ito. Average sensitivity of Euclidean k-clustering. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:32487–32498, 2022.