Efficiently Navigating a Random Delaunay Triangulation

Nicolas Broutin, Olivier Devillers, Ross Hemsley

Efficiently Navigating a Random Delaunay Triangulation

Nicolas Broutin [1] [†] and Olivier Devillers [2] [‡] and Ross Hemsley [2] [§]

[1] *Inria, 23 avenue d'Italie, 75013 Paris* [2] *Inria, 2004 route des Lucioles - BP 93, 06902 Sophia Antipolis*

Abstract. Planar graph navigation is an important problem with significant implications to both point location in geometric data structures and routing in networks. Whilst many algorithms have been proposed, very little theoretical analysis is available for the properties of the paths generated or the computational resources required to generate them. In this work, we propose and analyse a new planar navigation algorithm for the Delaunay triangulation. We then demonstrate a number of strong theoretical guarantees for the algorithm when it is applied to a random set of points in a convex region.

Keywords: Routing, Point Location, Randomised Analysis, Delaunay Triangulation

1 Introduction

Given a planar embedding of a graph $G = (V, E)$, a source node $z \in V$ and a destination point $q \in \mathbb{R}^2$, we consider the *planar graph navigation* problem of finding a route in G from z to the nearest neighbour of q in V. In particular, we assume that any vertex $v \in V$ may access its coordinates in $\mathbb{R}^2$ with a constant time query. The importance of this problem is twofold. On the one hand, finding a short path between two nodes in a network is currently a very active area of research in the context of routing in networks [1, 19, 22]. On the other, the problem of locating a face containing a point in a convex subdivision (point location) is an important sub-routine in many algorithms manipulating geometric data structures [10, 11, 17, 20]. A number of algorithms have been proposed within each of these fields, many of which are in fact equivalent. It seems the majority of the literature in these areas is concerned with the existence of algorithms which always succeed under different types of constraints, such as the *competitiveness* of the algorithm, or the class of network. Apart from worst-case bounds, very little is known concerning the properties of the path lengths and running times for these algorithms under random distribution hypotheses for the input vertices. We aim to bridge the gap between these two fields by giving and analysing an algorithm that is provably efficient within both of these contexts when the underlying graph is the Delaunay triangulation. More precisely, we show that even for the worst possible pair of source and destination chosen from a uniformly random input, our algorithm remains efficient in an asymptotic sense. To the best of our knowledge, our results provide the first analysis for a *localised*[(i)]

[†] Projet RAP, INRIA Paris - Rocquencourt
[‡] Projet Geometrica, INRIA Sophia Antipolis - Méditerranée
[§] Projet Geometrica, INRIA Sophia Antipolis - Méditerranée

[(i)] We call a routing algorithm *localised* if it eventually 'forgets' where it came from in a technical sense that we define in Section 1.3.

routing algorithm on a randomly generated planar graph. In addition, we take particular care when dealing with boundary effects that occur when the routing algorithm approaches the edge of the domain, achieving bounds that remain asymptotically the same irrespective of where in the process the input points are chosen from. Finally, we believe our results help provide convincing evidence that similar asymptotic results should hold for other localised routing algorithms such as *greedy routing* (which we define in Section 1.2).

In this extended abstract we will outline the main ingredients towards the theorems given in Section 1.3, and give a proof that the number of vertices accessed by the algorithm is proportional to the length of the walk, for a fixed source and destination that are sufficiently far apart. For the full technical details, we refer the reader to [8].

1.1 Definitions

In the following, we define the *competitiveness* of an algorithm to be the worst case ratio between the length of the path generated by the algorithm and the Euclidean distance between the source and the destination. Thus competitiveness may depend on the class of graphs one allows, but not the pair of source and destination. Let $\mathbf{N}_d(v)$ denote the set of neighbours of v within d hops of v. We shall sometimes refer to the d'th neighbourhood of a set X, to denote the set of all sites that can be accessed from a site in X with fewer than d hops. We call an algorithm *c-memoryless* if at each step in the navigation, it only has access to the destination q, the current vertex v and $\mathbf{N}_c(v)$. Some authors use the term *online* to refer to an algorithm that only has access to q, the current vertex v, $\mathbf{N}_1(v)$ and $O(1)$ words of memory which may be used to store information about the history of the navigation. Finally, an algorithm may be either deterministic or randomised. We define a randomised algorithm to be an algorithm that has access to a random oracle at each step.

1.2 Previous results

Graph Navigation for Point location. The problem of point location is most often studied in the context of triangulations and the algorithms are referred to as *walking algorithms* [11]. A walking algorithm may work by following edges or by following incidences between neighbouring faces, which is equivalent to a navigation in the dual graph. There are three main algorithms that have received attention in the literature: *straight walk*, which is a walk that visits all triangles crossed by the line segment zq; *greedy vertex walk*, which always chooses the vertex in $\mathbf{N}_1(v)$ which is closest to q and *visibility walk* which walks to an adjacent triangle if and only if it shares the same half-space as q relative to the shared edge. It is known that these algorithms always terminate if the underlying triangulation is Delaunay [11].

The aim is generally to analyse the expected number of steps that the algorithm requires to reach the destination under a given distribution hypothesis. Such an analysis has only been provided by Devroye et al. [13] who succeeded in showing that straight walk reaches the destination after $O(\|zq\|\sqrt{n}\,)$ steps in expectation, for n random points in the unit square.[(ii)] The analysis in this case is facilitated since it is possible to compute the probability that a triangle is part of the walk without looking at the other vertices. Straight walk is online, but not memoryless since at every step the algorithm must know the location of

[(ii)] Zhu also provides a tentative $O\left(\sqrt{n\log n}\,\right)$ bound for visibility walk [23]. This is a proof by induction for "a random edge at distance d". It considers the next edge in the walk and applies an induction hypothesis to try and bound the progress. Unfortunately, the new edge cannot be considered as random: each edge that is visited has been chosen by the algorithm and the edges do not all have the same probability to be chosen at each step in the walk. Restarting the walk from a given edge is not possible either (as done in [12]), since the knowledge that an edge is a Delaunay edge influences the local point distribution.

the source point, z. It is also rarely used in practice since it is usually outperformed empirically by one of the remaining two algorithms, *visibility walk* or *greedy vertex* walk, which are both 1-memoryless [11]. The complex dependence between the steps of the algorithm in these cases makes the analysis difficult, and it remains an important open question to provide an analysis for either of these two algorithms.

Graph Navigation for Routing. In the context of packet routing in a network, each vertex represents a node which knows its approximate location and can communicate with a selected set of neighbouring nodes. One example is in wireless networks where a node communicates with all devices within its communication range. In such cases, it is often convenient for the nodes to agree on a communication protocol such that the graph of directly communicating nodes is planar, since this can make routing more efficient. Delaunay triangulations have been used in this context due to their ability to act as spanners (the length of shortest paths in the graph, seen as curves in $\mathbb{R}^2$, do not exceed the Euclidean distance by more than a constant factor), and methods exist to locally construct the full Delaunay triangulation, given some conditions on the point distribution [16, 18, 21].

Commonly referenced algorithms in this field are: *greedy routing*, which is the same algorithm as *greedy vertex walk*, given in the context of point location; *compass routing* which is similar to greedy, except that instead of choosing the point in $\mathbf{N}_1(v)$ minimising the distance to q, it chooses the point in $x \in \mathbf{N}_1(v)$ minimising the angle $\angle q, v, x$, and also *face routing* which is a generalisation of straight walk that can be applied to any planar graph. In this context, overall computation time is usually considered less important than trying to construct algorithms that find short paths in a given network topology under certain memory constraints. We give a brief overview of results relating to this work.

Bose et al. [7] demonstrated that it is not possible to construct a deterministic online algorithm that finds a path with constant competitiveness in an *arbitrary* triangulation. They also demonstrated by counter example that neither greedy routing, nor compass routing is $O(1)$-competitive on the Delaunay triangulation [5]. Bose and Morin [6] went on to show that there does, however, exist an online c-competitive algorithm that works on any graph satisfying a property they refer to as the 'diamond property', which is satisfied by Delaunay triangulations. They show this by providing an algorithm which is essentially a modified version of the straight walk. Bose and Morin also show that there is no algorithm that is competitive for the Delaunay triangulation under the link length (the link length is the number of edges visited by the algorithm) [7].

In terms of time analysis, it appears the only relevant results are those by Chen et al. [9], who show that no memoryless routing algorithm is asymptotically better than a random walk when the underlying graph is an arbitrary convex subdivision.

Navigation in the Plane. We briefly remark that for the related problem of navigation in the plane, several probabilistic results exist; for example [2] and [3]. In this context, the input is a set of vertices in the plane along with an oracle that can compute the next step given the current step and the destination in $O(1)$ time. Although the steps are also dependent in these cases, the case of Delaunay triangulations we treat here is more delicate because of the geometry of the region of dependence implied by the Delaunay property.

1.3 Contributions

In this paper we give a new deterministic planar graph navigation algorithm which we call *cone walk* that succeeds on any Delaunay triangulation and produces a path which is 3.7-competitive. We briefly

underline the fact that our algorithm has been designed for theoretical demonstration, and we do not claim that it would be faster in a practical sense than, for example, *greedy routing* or *face routing*. On the other hand, direct comparisons would perhaps be unfair, since *greedy routing* is not $O(1)$-competitive on the Delunay triangulation [5] whereas we prove that cone walk is; and *face routing* is not memoryless in any sense (since it must always remember the initial vertex), whilst cone walk is localised in the sense given by Theorem 1. In the theorems that follow, we characterise the asymptotic properties of the cone walk algorithm applied to a random input.

Let $\mathcal{D}$ be a smooth convex domain of the plane with area 1, and write $\mathcal{D}_n = \sqrt{n}\mathcal{D}$ for its scaling to area n. For $x, y \in \mathcal{D}$, let $\|xy\|$ denote the Euclidean distance between x and y. Under the hypothesis that the input is the Delaunay triangulation of n points uniformly distributed in a convex domain of unit area, we prove that, for any $\varepsilon > 0$, our algorithm is $O(\log^{3+\varepsilon} n)$-memoryless with probability tending to one. In the case of cone walk, this is equivalent to bounding the number of neighbourhoods that might be accessed during a step, which we deal with in the following theorem.

Theorem 1 *Let $\mathbf{X}_n := \{X_1, X_2, \dots, X_n\}$ be a collection of n independent uniformly random points in $\mathcal{D}_n$. For $z \in \mathbf{X}_n$ and $q \in \mathcal{D}_n$, let $M(z, q)$ be the maximum number of neighbourhoods needed to compute any step of the walk. Then, for every $\varepsilon > 0$,*

$$\mathbb{P}\Big(\exists z \in \mathbf{X}_n,\, q \in \mathcal{D}_n : M(z, q) > \log^{3+\varepsilon} n\Big) \leq \frac{1}{n}. \tag{1}$$

In particular, as $n \to \infty$, $\mathbb{E}[\sup_{z\in\mathbf{X}_n, q\in\mathcal{D}_n} M(z, q)] = O(\log^{3+\varepsilon} n)$, for every $\varepsilon > 0$.

Also with probability close to one, we show that the path length, the number of edges and the number of vertices accessed are $O(\,\|zq\| + \log^6 n\,)$ for any pair of points in the domain. We formalise these properties in the following theorem.

Theorem 2 *Let $\mathbf{X}_n := \{X_1, X_2, \dots, X_n\}$ be a collection of n independent uniformly random points in $\mathcal{D}_n$. Let $\Gamma(z, q)$ denote either the Euclidean length of the path generated by the cone walk from $z \in \mathbf{X}_n$ to $q \in \mathcal{D}_n$, its number of edges, or the number of vertices accessed by the algorithm when generating it. Then there exist constants $C_{\Gamma,\mathcal{D}}$ depending only on Γ and on the shape of $\mathcal{D}$ such that, for all n large enough,*

$$\mathbb{P}\left(\exists z \in \mathbf{X}_n,\, q \in \mathcal{D}_n \;:\; \Gamma(z, q) > C_{\Gamma,\mathcal{D}} \cdot \|zq\| + 4\Big(1 + \sqrt{\|zq\|}\Big)\log^6 n\right) \leq \frac{1}{n}. \tag{2}$$

In particular, as $n \to \infty$,

$$\mathbb{E}\left[\sup_{z\in\mathbf{X}_n,\, q\in\mathcal{D}_n} \Gamma(z, q)\right] = O(\sqrt{n}\,). \tag{3}$$

Finally, we bound the computational complexity of the algorithm, $T(z, q)$.

Theorem 3 *Let $\mathbf{X}_n := \{X_1, X_2, \dots, X_n\}$ be a collection of n independent uniformly random points in $\mathcal{D}_n$. Then in the RAM model of computation, there exists a constant C depending only on the shape of $\mathcal{D}$ and the particular implementation of the algorithm such that for all n large enough,*

$$\mathbb{P}\left(\exists z \in \mathbf{X}_n,\, q \in \mathcal{D}_n \;:\; T(z, q) > C \cdot \|zq\| \log\log n + \Big(1 + \sqrt{\|zq\|}\Big)\log^6 n\right) \leq \frac{1}{n}. \tag{4}$$

In particular, as $n \to \infty$,

$$\mathbb{E}\Big[\sup_{z\in\mathbf{X}_n,\, q\in\mathcal{D}_n} T(z,q)\Big] = O(\sqrt{n}\log\log n\,). \tag{5}$$

Remark 1 *We conjecture that the factor of* $\log\log(n)$ *in Theorem 3 can be removed. However, we were unable to demonstrate this due to the complex dependancy structure between the steps.*

Remark 2 *The choice of the initial vertex is never discussed. However, previous results show that choosing this point carefully can result in an expected asymptotic speed up for any graph navigation algorithm [20].*

2 The Cone Walk Algorithm

Consider the finite set of sites in general position, $\mathbf{X} \subset \mathbb{R}^2$ contained within a compact convex domain $\mathcal{D} \subset \mathbb{R}^2$. Let $\mathrm{DT}(\mathbf{X})$ be the Delaunay triangulation of $\mathbf{X}$, which is the graph in which three sites $x, y, z \in \mathbf{X}$ form a triangle if and only if the disc with x, y and z on its boundary does not contain any site in $\mathbf{X}$. Given two points $z, q \in \mathbb{R}^2$ and a number $r \in \mathbb{R}$ we define $\mathrm{Disc}(z, q, r)$ to be the closed disc whose diameter spans z and the point at a distance $2r$ from z on the ray zq. Finally, we define $\mathrm{Cone}(z, q, r)$ to be the sub-region of $\mathrm{Disc}(z, q, r)$ contained within a closed cone of apex z, axis zq and half angle $\frac{\pi}{8}$ (see Figure 2).

Given a site $z \in \mathbf{X}$ and a destination point $q \in \mathcal{D}$, we define one *step* of the cone walk algorithm by growing the region $\mathrm{Cone}(z, q, r)$ anchored at z from $r = 0$ until the first point $z' \in \mathbf{X}$ is found such that the region is non-empty. Once z' has been determined, we refer to it as the *stopper*. We call the region $\mathrm{Cone}(z, q, r)$ for the given r a *search cone*, and we call the associated disc $\mathrm{Disc}(z, q, r)$ the *search disc* (see Figure 2). The point z' is then selected as the anchor of a new search cone $\mathrm{Cone}(z', q, \cdot)$ and the next step of the walk begins. See Figure 1 for an example run of the algorithm.

To find the stopper using only neighbour incidences in the Delaunay triangulation, we need only access vertices in a well-defined local neighbourhood of the search disc. Define the points $\mathbf{X} \cap \mathrm{Disc}(z, q, r) \setminus \{z, z'\}$ to be the *intermediate vertices*. The algorithm finds the stopper at each step by gradually growing a disc anchored at z in the direction of the destination, adding the neighbours of all vertices in $\mathbf{X}$ intersected along the way. This is achieved in practice by maintaining a series of candidate vertices initialised to the neighbours of z and selecting amongst them the vertex defining the smallest search disc at each iteration. Each time we find a new vertex intersecting this disc, we check to see if it is contained within $\mathrm{Cone}(z, q, \infty)$. If it is, this point is the next stopper and this step is finished. Otherwise the point must be an intermediate vertex and we add its neighbours to the list of candidate vertices. This procedure works because the intermediate vertex defining the next largest disc is always a neighbour of one of the intermediate vertices that we have already visited during the current step.

We terminate the algorithm when the destination q is contained within the current search disc for a given step. At this point we know that one of the points contained within $\mathrm{Disc}(z, q, r)$ is a Delaunay neighbour of q in $\mathrm{DT}(\mathbf{X} \cup \{q\})$. We can further compute the triangle of $\mathrm{DT}(\mathbf{X})$ containing the query point q (point location) or find the nearest neighbour of q in $\mathrm{DT}(\mathbf{X})$ by simulating the insertion of the point q into $\mathrm{DT}(\mathbf{X})$ and performing an exhaustive search on the neighbours of q in $\mathrm{DT}(\mathbf{X} \cup \{q\})$.

We will distinguish between the *visited* vertices, which we take to be the set of all sites contained within the search discs for each step, and the *accessed* vertices, which we define to be the visited vertices along

with their 1-hop neighbourhood. This distinction will be important, since the sequence of steps in the walk depends only on the vertices visited, but the cone walk algorithm must access the set of 'vertices accessed' in order to compute the sequence of steps efficiently using only local information.

The pseudo-code below gives a detailed algorithmic description of the CONE-WALK algorithm. We take as input some $z \in \mathbf{X}$, $q \in \mathcal{D}$ and return a Delaunay neighbour of q in $\mathrm{DT}(\mathbf{X} \cup \{q\})$. Recalling that $\mathbf{N}_1(v)$ refers to the Delaunay neighbours of $v \in \mathrm{DT}(\mathbf{X})$ and additionally defining NEXT-VERTEX(S, z, q) to be the procedure that returns the vertex in S with the smallest r such that $\mathrm{Disc}(z, q, r)$ touches a vertex in S and IN-CONE(z, q, y) to be `true` when $y \in \mathrm{Cone}(z, q, \infty)$.

```
CONE-WALK(z, q)
 1  Substeps = {z}
 2  Candidates = N_1(z)
 3  while true
 4      y = NEXT-VERTEX(Candidates ∪ {q}, z, q)
 5      if IN-CONE(z, q, y)
 6          if y = q
 7              // Destination reached.
 8              return NEXT-VERTEX(Substeps, q, z)
 9          // y is a stopper
10          z = y
11          Substeps = {z}
12          Candidates = N_1(z)
13      else
14          // y is an intermediate vertex.
15          Substeps = Substeps ∪ {y}
16          Candidates = Candidates ∪ N_1(y) \ Substeps
```

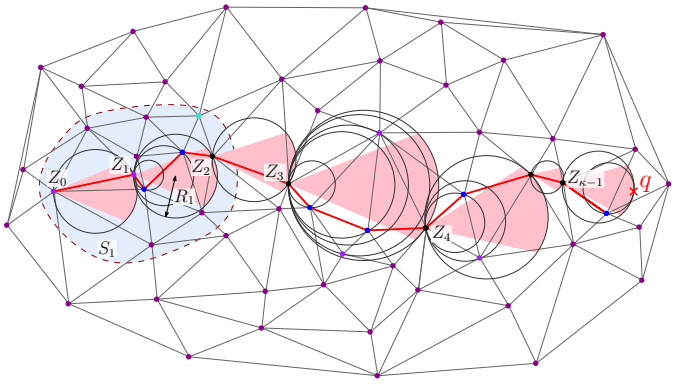

Fig. 1: An example of cone walk.

We note that the order in which the vertices are discovered during the walk does not necessarily define a path in $\mathrm{DT}(\mathbf{X})$. If we only wish to find a point of the triangulation that is close to the destination (for example, in point location), this is not a problem. However, in the case of routing, a path in the triangulation is required to provide a route for data packets. To this end, we provide two options that we shall refer to as SIMPLE-PATH and COMPETITIVE-PATH. SIMPLE-PATH is a simple heuristic that can quickly generate a path that is provably short on average. We conjecture that SIMPLE-PATH is indeed competitive, however we were unable to prove this. COMPETITIVE-PATH is slightly more complex from an implementation point of view, however we show that for any possible input the algorithm will always generate a path of constant competitiveness whilst still maintaining the same asymptotic behaviour as SIMPLE-PATH.

3 Asymptotic Analysis for Cone Walk

We now focus on the asymptotic properties of cone walk applied to points drawn from a randomly generated input. We will derive all of our results using a spatial Poisson process, since it has useful independence properties that are not available when one simply samples n random points in a fixed region. The theorems in Section 1.3 then follow by 'de-poissonising' (details given in [8]). Let $\mathbf{\Phi}_n$ be a planar Poisson process contained within $\mathcal{D}_n$, with intensity measure given by the standard two-dimensional Lebesgue measure λ. We have the following properties, for A, B Borel subsets of $\mathcal{D}_n$.

1. $\mathbb{E}\,|A \cap \mathbf{\Phi}_n| = \lambda(A)$, so in particular $\mathbb{E}\,|\mathbf{\Phi}_n| = n$;
2. if $A \cap B = \varnothing$ then $\mathbb{P}(|A \cap \mathbf{\Phi}_n| = k \mid |B \cap \mathbf{\Phi}_n| = t) = \mathbb{P}(|A \cap \mathbf{\Phi}_n| = k)$;

3. $\mathbb{P}(|A \cap \mathbf{\Phi}_n| = \varnothing) = \exp(-\lambda(A))$;
4. for all $k > e^2 \cdot \lambda(A)$ we have $\mathbb{P}(|A \cap \mathbf{\Phi}_n| \geq k) \leq \exp(-k)$.

In this extended abstract, we limit ourselves to studying the number of vertices that may be accessed by the walk for a fixed pair of randomly chosen start and destination points, since we believe this is one of the key properties of the cone walk algorithm. The results given here may then be extended to bound the number of steps accessed for any run of the algorithm by showing that it is possible to generate a 'small enough' sample containing every possible instance of cone walk on $\mathbf{\Phi}_n \times \mathcal{D}_n$ with high probability. This proof is quite technical, and we omit it here. In the following, we shall denote the sequence of stoppers visited by the walk process between fixed z and q as $(Z_i)_{i>0}$ (taking $Z_0 := z$), and the sequence of radii of the discs $(R_i)_{i>0}$. We also write $D_i := \mathrm{Disc}(Z_i, q, R_i)$, the i'th disc and let $W := \cup_{i>0} D_i$.

3.1 The number of sites accessed

Let $K := |\mathbf{\Phi}_n \cap W|$ be the number of *visited* vertices for an instance of cone walk. Recall that if we now wish to count the number of vertices that would be *accessed* by the cone walk algorithm to compute this walk, we will need to take K and add the number of vertices of $\mathbf{\Phi}_n$ that are within the 1-hop neighbourhood of the *visited* vertices. The difficulty in bounding this quantity comes from the fact that the vertices outside of the walk discs may be visited multiple times, introducing complex dependency relationships and making the analysis more subtle.

We begin by noting that any vertex outside of W is accessed by the cone walk algorithm exactly once for each edge leaving that vertex and crossing the boundary of W. We call the union of all such edges for every accessed vertex the *crossing edges*. To give a bound on the number of accessed vertices, we now require Proposition 4 given below. Its proof follows from the fact that, given some conditions, the contents of the cones (and discs) for each step behave as independent and identically distributed random variables. These technical details are dealt with in [8]. In the following, let ω_n be a sequence satisfying $\omega_n \geq \log n$.

Proposition 4 *Let K be the number of vertices visited by cone walk starting from a given site z with $L = \|zq\|$. Then, for $c_1 > 0$ a constant defined in [8] and for all n large enough,*

$$\mathbb{P}\left(K \geq c_1 L + \sqrt{L}\,\omega_n^4 + \omega_n^3\right) \leq 7 \exp\left(-\omega_n^{3/2}\right). \tag{6}$$

The following proposition now bounds the number of vertices accessed.

Proposition 5 *Let $A = A(z, q)$ be the number of sites in $\mathbf{\Phi}_n$ accessed by the cone walk algorithm (with multiplicity) when walking towards q from z, and $L = \|zq\|$. Then there exists a constant $c_2 > 0$ such that (for n sufficiently large)*

$$\mathbb{P}\left(A(z,q) > c_2\, L + 4\left(\sqrt{L} + 1\right)\omega_n^6\right) \leq 3 \exp\left(-\omega_n^{5/4}\right). \tag{7}$$

Proof: In order to bound the number of such edges, we adapt the concept of the *border point* introduced by Bose and Devroye [4] to bound the stabbing number of a random Delaunay triangulation. For $B \subseteq \mathcal{D}_n$ and a point $x \in \mathcal{D}_n$, let $\|xB\| := \inf\{\|xy\| : y \in B\}$ denote the distance from x to B.

We consider the walk from z to q in $\mathcal{D}_n$, letting

$$W = \bigcup_{i=1}^{\kappa} D_i \qquad \text{and} \qquad W^{\odot} := b(\mathbf{w}, 2\max\{\|zq\|, \omega_n^5\}), \tag{8}$$

where $\mathbf{w}$ denotes the centroid of the segment zq and we recall that $b(\mathbf{x}, r)$ denotes the closed ball centred at $\mathbf{x}$ of radius r. Then, for $x \in W^{\odot}$, let C be the disc centered at x and with radius $\min\{\|xW\|, \|x\partial W^{\odot}\|\}$ (where ∂A is used to denote the boundary of A). Partition the disc C into eight cone-shaped sectors (such that one of the separation lines is vertical, say) truncated to a radius of $\sqrt{3}/2$ times that of the outer disc (see Figure 3). We now say that x is a *border point* if one of the eight cones does not contain any points in $\mathbf{\Phi}_n$. If $x \in W^{\odot}$ is not a border point then there is no Delaunay edge between x and a point lying outside C, since a circle trough x and $y \notin C \subset W^{\odot} \setminus W$ must entirely enclose at least one sector of C (see dotted circle in Figure 3). Thus if x has a Delaunay edge with extremity in W, then x must be a border point.

The connection between border points and the number of crossing edges can be made via Euler's relation, since it follows that a crossing edge is an edge of the (planar) subgraph of $\mathrm{DT}(\mathbf{\Phi})$ induced by the points which either lie inside W, are border points, or lie outside of $W^{\odot}$ and have a neighbour in W. Let B_W denote of set of border points, E_W the collection of crossing edges, and Y_W the collection of points lying outside of $W^{\odot}$ and having a Delaunay neighbor within W. Then

$$A(z, q) \leq |E_W| \leq 3(|W \cap \mathbf{\Phi}| + |B_W| + |Y_W|). \tag{9}$$

Proposition 4 bounds $|W \cap \mathbf{\Phi}|$, as this is exactly the set of *visited vertices*. Lemmas 6 and 7 bounding $|B_W|$ and $|Y_W|$ complete the proof. □

Lemma 6 *For all n large enough, we have*

$$\mathbb{P}\Big(|Y_W| \geq 10\max\{L, \omega_n^5\}\Big) \leq 2\exp\left(-\omega_n^{5/4}\right). \tag{10}$$

Proof: Heuristically, our proof will follow from the fact that, with high probability, a Delaunay edge away from the boundary of the domain is not long enough to span the distance between a point within the walk, and a point outside of $W^{\odot}$. Unfortunately our proof is complicated by points on the walk which are very close to the boundary of the domain, since in this case, those points might have 'bad' edges which are long enough to escape $W^{\odot}$. To deal with this, we will take all points in the walk that are close to the border, and imagine that every Delaunay edge touching one of these points is such a 'bad' edge. The total number of these edges will be bounded by the maximum degree.

To begin, we give the first case. Consider an arbitrary point $x \in W \cap \mathbf{\Phi}$ that is at least ω_n^{-3} away from the boundary of $\mathcal{D}_n$. Suppose this point has a neighbour outside of $W^{\odot}$, then its circumcircle implicitly overlaps an unconditioned region of $\mathcal{D}_n$ with area at least $c\,\omega_n^{-3}\,\omega_n^5 = c\,\omega_n^2$ (for $c > 0$ a constant depending on the shape of the domain). The probability that this happens for x is thus at most $\exp\left(-\omega_n^2\right)$. Now note that there are at most $2n$ points in $\mathbf{\Phi}$ with probability bounded by $\exp\left(-\omega_n^2\right)$ and at most $4n^2$ edges between points of $x \in W \cap \mathbf{\Phi}$ and $x \in \{W^{\odot}\}^c \cap \mathbf{\Phi}$. By the union bound, the probability that any such edge exists is at most

$$(4n^2)\exp\left(-c\omega_n^2\right) + \exp\left(-\omega_n^2\right) \leq \exp\left(-\omega_n^{3/2}\right). \tag{11}$$

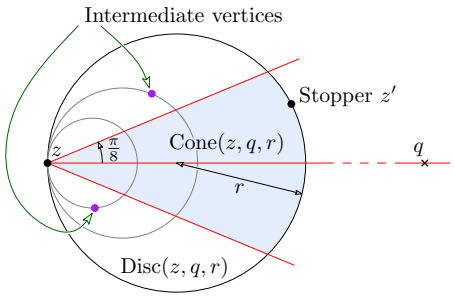

Fig. 2: One step of the cone-walk algorithm.

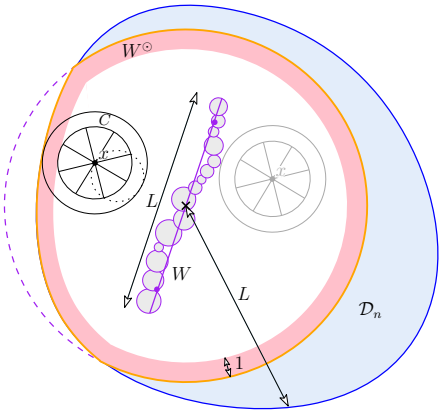

Fig. 3: For the proof of Proposition 5.

For the second case, we count the number of points within ω_n^{-3} of the boundary of the domain. Using standard arguments, we have that there are no more than $10\max\{L,\omega_n^5\}\cdot\omega_n^{-3}$ such points, with probability at least $\exp\left(-\omega_n^2\right)$. Each of these has at most $\Delta_{\mathbf{\Phi}}$ edges that could exit $W^{\odot}$, where $\Delta_{\mathbf{\Phi}}$ is the maximum degree of any vertex in $\mathrm{DT}(\mathbf{\Phi})$, which is bounded in Proposition 30 of [8]. Thus, the number of such bad edges is at most $10\max\{L,\omega_n^5\}\omega_n^{-3}\cdot\omega_n^3$ with probability at least $\exp(-\omega_n^2)+\exp(-\omega_n^{5/4})$ □

Lemma 7 *For all n large enough, and universal constant $C>0$,*

$$\mathbb{P}\Big(|B_W|\geq C\max\{L,\omega_n^7\}\Big)\leq 2\exp\left(-\omega_n^{3/2}\right). \tag{12}$$

Proof: Since $|B_W|$ is a sum of indicator random variables, it can be bounded using (a version of) Chernoff–Hoeffding's method. The only slight annoyance is that the indicators $\mathbf{1}_{\{x\in B_W\}}, x\in\mathbf{\Phi}_n\cap W^{\odot}$ are not independent. Note however that $\mathbf{1}_{\{x\in B_W\}}$ and $\mathbf{1}_{\{y\in B_W\}}$ are only dependent if the discs used to define membership to B_W for x and y intersect. There is a priori no bound on the radius of these discs, and so we shall first discard the points $x\in\mathbf{\Phi}_n$ lying far away from $\partial W^{\odot}$ and W. More precisely, let $B_W^{\star}$ denote the set of border points lying within distance ω_n of either W or $\partial W^{\odot}$, and $B_W^{\bullet}=B_W\setminus B_W^{\star}$. We now bound $|B_W^{\bullet}|$ by noting that the probability that there exists an empty circle of radius greater than $c\cdot\omega_n$ (for any $c>0$) in $\mathcal{D}_n$ is at most $\exp(-\omega_n^{3/2})$; and that each cone in the border point construction implicitly contains an empty circle (this is formally dealt with in Lemma 10 in [8]). Thus for sufficiently large n,

$$\mathbb{P}\big(|B_W^{\bullet}|\neq 0\big)\leq\exp\left(-\omega_n^{3/2}\right). \tag{13}$$

Bounding $|B_W^{\star}|$ is now easy since the amount of dependence in the family $\mathbf{1}_{\{x\in B_W\}}, x\in\mathbf{\Phi}_n\setminus W$ is controlled and we can use the inequality by Janson [14, 15]. We start by bounding the expected value $\mathbb{E}\,|B_W^{\star}|$. Note that for a single point $x\in\mathbf{\Phi}_n$, by definition the disc used to define whether x is a border point does not intersect W and stays entirely within $\mathcal{D}_n$, so

$$\begin{aligned}\mathbb{P}_x(x\in B_W^{\star})=\mathbb{P}(x\in B_W^{\star}) &\leq 24\exp\left(-\frac{\pi}{32}\min\{\|xW\|,\|x\partial W^{\odot}\|\}^2\right)\\ &\leq 24\exp\left(-\frac{\pi}{32}\|xW\|^2\right)+24\exp\left(-\frac{\pi}{32}\|x\partial W^{\odot}\|^2\right)\end{aligned} \tag{14}$$

and $\mathbf{\Phi}_n$ is unconditioned in $\mathcal{D} \setminus W$. Partition $\mathcal{D} \setminus W$ into disjoint sets as follows:

$$\mathcal{D} \setminus W = \bigcup_{i=0}^{\infty} U_i \tag{15}$$

where $U_i := \{x \in \mathcal{D} \ : \ i \leq \|xW\| < i+1\}$. Similarly, the sets $U_i' := \{x \in W^{\odot} : i \leq \|x\partial W^{\odot}\| < i+1\}$ form a similar partition for $W^{\odot}$. Using (14) above, we may apply the Slivnyak-Mecke formula to get

$$\begin{aligned}
\mathbb{E}|B_W^{\star}| &= \mathbb{E}\left[\sum_{x\in\mathbf{\Phi}_n} \mathbf{1}_{\{x\in W^{\odot}\setminus W\}}\ \mathbf{1}_{\{x\in B_W^{\star}\}}\right] && (16)\\
&= \int_{W^{\odot}\setminus W} \mathbb{P}_x(x \in B_W^{\star})\, \lambda(dx) && (17)\\
&\leq \sum_{i=0}^{\infty} \int_{U_i} 24\exp\left(-\pi i^2/32\right)\ \lambda(dx) + \sum_{i=0}^{\infty}\int_{U_i'} 24\exp\left(-\pi i^2/32\right)\ \lambda(dx) && (18)\\
&= 24\sum_{i=0}^{\infty}(\lambda(U_i)+\lambda(U_i'))\ \exp\left(-\pi i^2/32\right). && (19)
\end{aligned}$$

We may now bound $\lambda(U_i)$ and $\lambda(U_i')$ as follows. Recall that W is a union of discs $W = \cup_i D_i$. We clearly have that

$$U_i \subseteq \bigcup_{j=0}^{\kappa-1} \big\{x \in \mathcal{D} \ : \ i \leq \|xD_j\| < i+1\big\}. \tag{20}$$

Note that

$$\begin{aligned}
\lambda\big(\big\{x \in \mathcal{D} \ : \ i \leq \|xD_j\| < i+1\big\}\big) &\leq \pi((R_j+i+1)^2 - (R_j+i)^2) && (21)\\
&= \pi(2(R_j+i)+1). && (22)
\end{aligned}$$

So, assuming there are κ steps in the walk we get

$$\lambda(U_i) \ \leq \ \sum_{j=0}^{\kappa-1}\pi(2(R_j+i)+1) \ = \ 2\pi\sum_{j=0}^{\kappa-1} R_j + \pi(i+1)\kappa. \tag{23}$$

Regarding $\lambda(U_i')$, note first that $W^{\odot}$ is convex for it is the intersection of two convex regions. It follows that its perimeter is bounded by $4\pi\max\{\|zq\|,\omega_n\}$, so that $\lambda(U_i') \leq 4\pi\max\{\|zq\|,\omega_n\}$ for every $i \geq 0$. It now follows easily that there exist universal constants C, C' such that

$$\mathbb{E}\,|B_W^{\star}| = \mathbb{E}\,\mathbb{E}\big[|B_W^{\star}| \ \big|\ R_i, i \geq 0\big] \leq C\,\mathbb{E}\left[\sum_{j=0}^{\kappa-1} R_j + \kappa + \max\{\|zq\|,\omega_n\}\right] \leq C'\max\{\|zq\|,\omega_n^6\}. \tag{24}$$

For the concentration, we use the fact that if $\|xW\|, \|x\partial W^{\odot}\| \leq \omega_n$ then the chromatic number χ of the dependence graph of the family $\mathbf{1}_{\{x \in B^{\star}_W\}}$ is bounded by the maximum number of points of $\mathbf{\Phi}_n$ contained in a disc of radius $2\omega_n$. Let $\mathrm{Po}(\lambda)$ denote a Poisson distributed random variable of rate λ. We have

$$\mathbb{P}(\chi \geq 8\pi\omega_n^2) \leq \mathbb{P}(\exists x \in \mathcal{D} : \mathbf{\Phi}_n \cap b(x, 2\omega_n)) \tag{25}$$

$$\leq \mathbb{E}\left[\sum_{x \in \mathbf{\Phi}_n} \mathbb{P}_x(|b(x, 2\omega_n) \cap \mathbf{\Phi}_n| \geq 8\pi\omega_n^2)\right] \tag{26}$$

$$\leq \mathbb{E}\left[\sum_{x \in \mathbf{\Phi}_n} \mathbb{P}(\mathrm{Po}(4\pi\omega_n^2) \geq 8\pi\omega_n^2)\right] \tag{27}$$

$$\leq \exp\left(-\omega_n^2\right), \tag{28}$$

for all n large enough. Let

$$W^{\partial} := \left\{x \in W^{\odot} \;\middle|\; \max\{\|xW\|, \|x\partial W^{\odot}\|\} \leq \omega_n\right\}.$$

Following Equation (21) and by the convexity of $W^{\odot}$, there exists a universal constant C'' such that

$$\mathbb{P}\left(|\mathbf{\Phi}_n \cap W^{\partial}| \geq C'' \max\{L, \omega_n^5\} \cdot \omega_n^2\right) \leq \exp\left(-\omega_n^2\right).$$

We thus obtain for $t > 0$ and by Theorem 3.2 of [14],

$$\mathbb{P}(|B^{\star}_W| \geq \mathbb{E}\,|B^{\star}_W| + t) \leq \mathbb{E}\left[\exp\left(-\frac{2t^2}{\chi \cdot |\mathbf{\Phi}_n \cap W^{\odot}|}\right)\right] \tag{29}$$

$$\leq \exp\left(-\frac{t^2}{8\pi\, C'' \max\{L, \omega_n^5\} \cdot \omega_n^5}\right) + \mathbb{P}(\chi \geq 8\pi\omega_n^2) + \mathbb{P}(|\mathbf{\Phi}_n \cap W^{\partial}| > C'' \max\{L, \omega_n^5\} \cdot \omega_n^4) \tag{30}$$

$$\leq \exp\left(-\frac{t^2}{8\pi\, C'' \max\{L, \omega_n^5\} \cdot \omega_n^4}\right) + 2\exp\left(-\omega_n^2\right). \tag{31}$$

And the claimed result follows for n sufficiently large by choosing $t := C''\left(L + \omega_n^6\right)$. □

Acknowledgements We would like to thank Marc Glisse, Mordecai Golin, Jean-François Marckert, and Andrea Sportiello for fruitful discussions during the Presage workshop on geometry and probability.

References

[1] F. Baccelli and B. Blaszczyszyn. *Stochastic Geometry and Wireless Networks*. NOW, 2009.

[2] N. Bonichon and J.-F. Marckert. Asymptotics of geometrical navigation on a random set of points in the plane. *Adv. in Applied Probability*, 43:899–942, 2011.

[3] C. Bordenave. Navigation on a Poisson point process. *The Annals of Applied Probability*, 18:708–746, 2008.

[4] P. Bose and L. Devroye. On the stabbing number of a random Delaunay triangulation. *Computational Geometry: Theory and Applications*, 36:89–105, 2006.

[5] P. Bose and P. Morin. Online routing in triangulations. *SIAM journal on computing*, 33:937–951, 2004.

[6] P. Bose and P. Morin. Competitive online routing in geometric graphs. *Theor. Comput. Sci.*, 324(2-3):273–288, 2004.

[7] P. Bose, A. Brodnik, S. Carlsson, E. Demaine, R. Fleischer, A. López-Ortiz, P. Morin, and J. Munro. Online routing in convex subdivisions. *International Journal of Computational Geometry & Applications*, 12:283–295, 2002.

[8] N. Broutin, O. Devillers, and R. Hemsley. Efficiently Navigating a Random Delaunay Triangulation. Rapport de recherche, Feb. 2014. URL `http://hal.inria.fr/hal-00940743`.

[9] D. Chen, L. Devroye, V. Dujmović, and P. Morin. Memoryless routing in convex subdivisions: Random walks are optimal. *Computational Geometry*, 45(4):178 – 185, 2012. ISSN 0925-7721. doi: 10.1016/j.comgeo.2011.12.005.

[10] P. M. M. de Castro and O. Devillers. Simple and efficient distribution-sensitive point location, in triangulations. In *ALENEX*, pages 127–138, 2011.

[11] O. Devillers, S. Pion, and M. Teillaud. Walking in a triangulation. *Int. J. Found. Comput. Sci.*, 13:181–199, 2002.

[12] L. Devroye, E. Mücke, and B. Zhu. A note on point location in Delaunay triangulations of random points. *Algorithmica*, 22: 477–482, 1998.

[13] L. Devroye, C. Lemaire, and J.-M. Moreau. Expected time analysis for Delaunay point location. *Comput. Geom. Theory Appl.*, 29:61–89, 2004.

[14] D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, Cambridge, UK, 2009.

[15] S. Janson. Large deviation for sums of partially dependent random variables. *Random Structures and Algorithms*, 24(3): 234–248, 2004.

[16] G. Kozma, Z. Lotker, and M. Sharir. Geometrically aware communication in random wireless networks. *ACM PODC*, 2004.

[17] C. L. Lawson. Software for C^1 surface interpolation. In J. R. Rice, editor, *Math. Software III*, pages 161–194. Academic Press, New York, NY, 1977.

[18] X.-Y. Li, G. Calinescu, P.-J. Wan, and Y. Wang. Localized delaunay triangulation with application in ad hoc wireless networks. *IEEE Trans. Parallel Distrib. Syst.*, 14(10):1035–1047, Oct. 2003. ISSN 1045-9219. doi: 10.1109/TPDS.2003.1239871.

[19] S. Misra, I. Woungang, and S. Misra. *Guide to wireless sensor networks*. Springer, 2009.

[20] E. P. Mücke, I. Saias, and B. Zhu. Fast randomized point location without preprocessing in two- and three-dimensional delaunay triangulations. *IN PROC. 12TH ANNU. ACM SYMPOS. COMPUT. GEOM*, 12:274–283, 1996. doi: 10.1.1.6.6917.

[21] Y. Wang and X.-Y. Li. Efficient delaunay-based localized routing for wireless sensor networks: Research articles. *Int. J. Commun. Syst.*, 20(7):767–789, July 2007. ISSN 1074-5351. doi: 10.1002/dac.v20:7.

[22] F. Zhao and L. Guibas. *Wireless Sensor Networks. An Information Processing Approach*. Morgan Kaufmann, 2004.

[23] B. Zhu. On Lawson's oriented walk in random Delaunay triangulations. In *Fundamentals of Computation Theory*, volume 2751 of *Lecture Notes Comput. Sci.*, pages 222–233. Springer-Verlag, 2003.