EXPECTED COMPLEXITY OF ROUTING IN Θ_6 AND HALF-Θ_6 GRAPHS*

Prosenjit Bose,† *Jean-Lou De Carufel,*‡ *Olivier Devillers,*§

ABSTRACT. We study online routing algorithms on the Θ_6-graph and the half-Θ_6-graph (which is equivalent to a variant of the Delaunay triangulation). Given a source vertex s and a target vertex t in the Θ_6-graph (resp. half-Θ_6-graph), there exists a deterministic online routing algorithm that finds a path from s to t whose length is at most $2\|st\|$ (resp. $2.89\|st\|$) which is optimal in the worst case [Bose *et al.*, SIAM J. on Computing, 44(6)]. We propose alternative, slightly simpler routing algorithms that are optimal in the worst case and for which we provide an analysis of the average routing factor for the Θ_6-graph and half-Θ_6-graph defined on a Poisson point process.

For the Θ_6-graph, our online routing algorithm has an expected routing factor of 1.161 when s and t are random. The *routing factor* is the length of the route between s and t produced by our algorithm divided by the Euclidean distance between s and t. Moreover, our routing algorithm has a maximum expected routing factor of 1.22, where the maximum is for fixed s and t and all other points are random. This is much better than the worst-case routing ratio of 2. The *routing ratio* is the maximum routing factor among all pairs of points. For the half-Θ_6-graph, our memoryless online routing algorithm has an expected routing factor of 1.43 and a maximum expected routing factor of 1.58. Our online routing algorithm that uses a constant amount of additional memory, has an expected routing factor of 1.34 and a maximum expected routing factor of 1.40. The additional memory is only used to remember the coordinates of the starting point of the route. Both of these algorithms have an expected routing factor that is much better than their worst-case routing ratio of 2.89.

1 Introduction

A weighted geometric graph $G = (P, E)$ is a graph whose vertex set is a set P of n points in the plane, and whose edge set is a set of line segments joining pairs of points in P, with each edge weighted by the Euclidean distance between its endpoints. A graph G is a geometric δ-spanner of the complete geometric graph provided that for every pair of points $(s, t) \in P^2$ the shortest path from s to t in G has weight at most $\delta \geq 1$ times $\|st\|$ where $\|st\|$ is the Euclidean distance from s to t. The minimum δ for which G is a δ-spanner is the *spanning ratio* or *stretch factor* of G. The spanning properties of various geometric graphs have been studied extensively in the literature (see [9, 16] for a comprehensive overview of the topic).

*This work has been supported by INRIA Associated team TRIP, by grant ANR-17-CE40-0017 of the French National Research Agency (ANR project ASPAG) and by NSERC.

†Carleton University, Ottawa, Canada. jit@scs.carleton.ca

‡University of Ottawa, Ottawa, Canada. jdecaruf@uottawa.ca

§Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France. olivier.devillers@inria.fr.

A routing algorithm for a geometric graph takes as input a pair of vertices (s, t) and finds a path from s to t in the graph. When full knowledge of the graph is available to the algorithm, numerous routing algorithms exist in the literature for finding paths in these graphs such as Breadth-First Search, Depth-First Search or Dijkstra's algorithm [11, 12, 14, 15]. The problem offers different challenges in the *online* setting. By the online setting, we mean that initially, the routing algorithm has limited knowledge of the graph and needs to simultaneously explore the graph while trying to find a path from s to t. Without knowledge of the whole graph, a routing algorithm, in general, cannot identify a short path. In certain cases, depending on the information available to the routing algorithm and limitations placed on the algorithm such as how much memory it has, it may not even find a path but may end up cycling without ever reaching its destination [7]. A formal definition of our routing model is given in Section 1.3.

By definition, a δ-spanner G of the complete graph simply implies the existence of a short path in G between every pair of vertices, i.e. a path whose weight is at most δ times the Euclidean distance between the vertices. There is a difference between knowing the existence of a short path and actually finding the path, particularly in an online setting. The goal of a competitive online routing algorithm is to find a short path when one exists. We define the *routing factor* of a routing algorithm between s and t to be the ratio of the length of the path produced by the algorithm to the Euclidean length $\|st\|$. A routing algorithm is *ρ-competitive* if for any two points s and t, the routing factor is not more than ρ. The minimum value ρ for which a routing algorithm is ρ-competitive is called the *routing ratio* of the algorithm [6]. Observe that by definition, the routing ratio is an upper bound on the spanning ratio.

Both the spanning ratio and the routing ratio are fragile measures. For example, G can be the complete graph that is missing only one edge but can still have unbounded spanning and routing ratios. Instead of analyzing the routing ratio, which is the worst case routing factor for a given graph and routing algorithm, we compute the expected routing factor of a given path or the expected routing factor of a random path in a random graph. One of the key difficulties in analyzing these ratios in the probabilistic sense is that often there is a lot of dependence in the process used by a routing algorithm to select which edge to follow. To overcome these barriers, we design simple routing strategies with good worst-case behavior that can also be analyzed in the expected sense.

1.1 Contribution

This paper has two main contributions. The first contribution consists of the design of two new algorithms for routing in the half-Θ_6-graph (also known as the TD-Delaunay triangulation [2]) in the so called *negative-routing* case, which is challenging since at each step, the routing algorithm must select one from many possible edges to follow, some of which lead you astray. Our new routing algorithms come in two flavors: one is memoryless and the other uses a constant amount of memory. These new negative-routing algorithms have a worst-case optimal routing ratio but are simpler and more amenable to probabilistic analysis than the known optimal routing algorithm [6]. We also provide a new point of view on routing [6] in the half-Θ_6-graph in the *positive-routing* case, which in some sense is

identical to the optimal routing algorithm on the Θ_6-graph. This new point of view allows us to complete the probabilistic analysis both on the half-Θ_6-graph and on the Θ_6-graph.

The second contribution is the analysis of the two new negative-routing algorithms and of the positive-routing algorithm in a random setting, namely when the vertex set of the Θ_6-graph and half-Θ_6-graph is a point set that comes from an infinite Poisson point process X of intensity $\lambda > 0$. The analysis is asymptotic with λ going to infinity, and gives the expected length of the shortest path between two fixed points s and t at distance one. Our results depend on the position of t with respect to s. We express our results both by taking the worst position for t and by averaging over all possible positions for t.

The routing ratio for our memoryless negative-routing algorithm in the half-Θ_6-graph is 2.89 in the worst case which is optimal. We get a routing factor of 1.58 in the expected case for the worst position for t, and 1.43 in the expected case when averaging over all possible positions for t. For our constant memory negative-routing algorithm, we obtain a routing ratio of 2.89 in the worst case, and a routing factor of 1.40 expected for the worst position for t, and 1.34 when averaging over all possible positions for t. For the routing ratio of the positive-routing algorithm on the half-Θ_6-graph, we obtain 2 in the worst case which is optimal, and a routing factor of 1.22 expected for the worst position of t, and 1.16 when averaging over all possible positions for t. Our results on routing in the Θ_6-graph are identical to the positive-routing strategy since the Θ_6-graph is the union of two half-Θ_6-graphs and one can locally differentiate the edges between the two spanning subgraphs. Therefore, our algorithm for Θ_6-routing is memoryless.

Formal definitions of the online routing model and the different graphs on which we route are outlined in the following subsections.

1.2 The Poisson Point Process

A Poisson point process X of intensity $\lambda > 0$ in the plane is an infinite set of points with probability 1, satisfying the following properties: the expected number of points of X in a domain A is $\lambda \cdot \text{Area}(A)$ and the number of points of X in two disjoint domains are independent. The number of points in A follows a Poisson's law:

$$\mathbb{P}\left[|X \cap A| = k\right] = \frac{\lambda^k \,\text{Area}\,(A)^k}{k!} e^{-\lambda \,\text{Area}(A)}.$$

1.3 The Online Routing Model

In its weakest form, an online local geometric routing algorithm on a graph $G = (P, E)$ can be expressed as a *routing function* $f : P \times P \times \mathcal{P}(P) \to P$, where $\mathcal{P}(\cdot)$ denotes the power set, with parameters $f(u, t, N(u))$ such that $u \in P$ is the vertex for which a forwarding decision is being made (i.e., the node currently holding the message), $t \in P$ is the destination vertex (target), and $N(u) \subseteq P$ is the set of neighbours of u in G. Upon receiving a message destined for t, node u forwards the message to its neighbour $z = f(u, t, N(u)) \in N(u)$. This routing strategy is referred to as *memoryless* routing. If the routing algorithm uses constant additional memory to store some information $i \in \mathcal{I}$, then this information is

taken into consideration when computing which neighbour to forward the message to. The function then becomes $f : P \times P \times \mathcal{P}(P) \times \mathcal{I} \to P$. Such a routing strategy is referred to as *constant-memory routing*. In the remainder of the article, the additional memory is solely used to store the coordinates of the vertex from which the message originated, i.e. the source vertex.

1.4 Definition of Θ_k-graphs

For any fixed integer $k \geq 2$, the Θ_k-graph is defined as follows. For each point $p \in P$, consider a set of rays originating from p with the angle between consecutive rays being $2\pi/k$. Each consecutive pair of rays defines a cone with apex p. Orient the cones such that there is one cone, labeled C_0^p, whose bisector is a vertical ray through p pointing upwards. Label the cones in counterclockwise order around p: $C_0^p, \ldots, C_{k-1}^p$. Given two points p and q define the *canonical triangle* T_{pq} to be the triangle bounded by the sides of the cone of p that contains q and the line through q perpendicular to the bisector of that cone. An edge in Θ_k exists between two vertices p and q if q is in some cone C_i^p, and for all points $w \in C_i^p$, $\|pq'\| \leq \|pw'\|$, where q' and w' denote the orthogonal projection of q and w onto the bisector of cone C_i^p. In other words, T_{pq} contains no points of the point set P, in which case we say that T_{pq} is *empty*. The half-Θ_k-graph is defined similarly for even k but only half the cones are considered for edge inclusion. Thus, an edge exists between two vertices p and q of the half-Θ_k-graph provided that q is in some cone C_i^p where i is even, and T_{pq} is empty. The *even cones* refer to the cones with even index and the *odd cones* refer to the ones with odd index. Similarly, we say that the canonical triangle T_{pq} is *even* (resp. *odd*) if $T_{pq} \subset C_i^p$ for i even (resp. odd). In fact, the Θ_k-graph is the union of the half-Θ_k-graph defined by the even cones and the half-Θ_k-graph defined by the odd cones.

1.5 Θ_k-routing

The structure of the Θ_k-graph naturally gives rise to a simple routing algorithm known as Θ_k-*routing*. Let t be the destination vertex. The Θ_k-routing algorithm invoked at an arbitrary vertex v consists of following the edge adjacent to v in the cone of v that contains t. This process is repeated until the destination t is reached.

It is known that Θ_k-routing terminates with routing ratio $\rho = 1 + f(k)$ where $f(k) \in o(1)$ for all Θ_k-graphs with $k \geq 7$ [5, 17]. There is a gap between the best known upper bound on the spanning ratio and ρ (see [5] for a survey of the best known bounds both on the spanning ratio and ρ). For $k = 2$, the graph is just the y-monotone chain of vertices ordered vertically. In this case, Θ_2-routing works but the routing ratio is unbounded. For $k = 3$, the graph is connected but Θ_3-routing may loop as in the example of Figure 1-left [1]. For $k \in \{4, 5, 6\}$, Θ_k-routing always finds a path but its length may be unbounded (see Figures 1 and 2). Alternative routing algorithms dedicated specifically for Θ_4 and Θ_6 graphs have been designed proving that they have constant routing ratio at most 17 [4], and at most 2 [2, 6]. For Θ_5, although this graph is known to have spanning ratio at most $\sqrt{50 + 22\sqrt{5}} < 10$ [8], no online routing algorithm with constant routing ratio is known. The main issue is that there is a lack of symmetry in the Θ_5-graph that makes it difficult

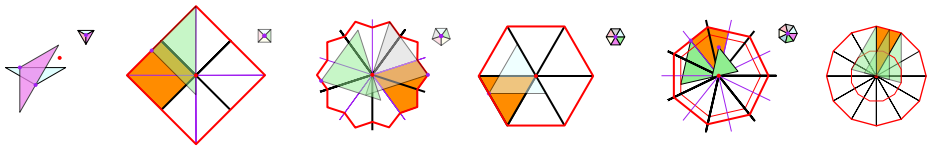

Figure 1: Routing in the Θ_k-graph for $k \in \{3, 4, 5, 6, 7, 8\}$ (from left to right). Θ_3-routing to the red point loops between the two purple points. Θ_k-routing ($k \in \{4, 5, 6\}$) from the red polygon to the red point goes strictly inside the polygon, thus decreasing the polygonal distance to the target and prove that routing will terminate on a finite set of points. For $k \in \{7, 8, \ldots\}$, one step from the boundary of the red polygon allows to quantify the decrease in polygonal distance to the target and prove that the routing ratio is bounded.

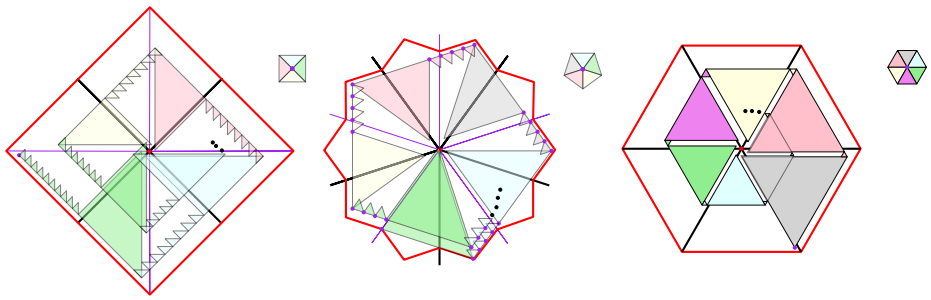

Figure 2: Routing ratio in the Θ_k-graph is unbounded for $k \in \{4, 5, 6\}$.

to route online. The proof that the spanning ratio is constant is constructive but it is an inductive proof that constructs a short path from both endpoints, thereby making it difficult to adapt some sort of local routing algorithm from the proof.

Bonichon and Marckert [3] analyze the expected length of the Θ_k-routing algorithm when the Θ_k-graph is defined on a Poisson point process with intensity λ tending to infinity. Their results are more complete since in addition to Θ_k-graphs, they address variants of Θ_k-graphs such as Yao-graphs or continuous Θ_k-graphs. But their results are limited to the standard Θ_k-routing algorithm. We focus specifically on the Θ_6-graph and the Half-Θ_6-graph and address different routing algorithms on these graphs in the remainder of the paper.

1.6 The Θ_6-graph and Half-Θ_6-graph

In the special case $k = 6$, the Θ_6-graph has some interesting properties. The six rays that define the cones around a point p make an angle of 0, $\frac{\pi}{3}$, $\frac{2\pi}{3}$, π, $\frac{4\pi}{3}$, and $\frac{5\pi}{3}$ with the horizontal axis. The triangles T_{pq} when q is in an even (resp. odd) cone C_i^p are all homothets. This is the **fundamental** property of these graphs that is not shared with any other Θ-graph. For other values of k, triangles are homothets only when they come from the same cone, i.e. with the same fixed value of i. For $k = 6$ just the parity of i matters, i.e. triangles arising

from cones where i is even (resp. odd) are homothets. Using this property, Bonichon *et al.* [2] noted that the half-Θ_6-graph is equivalent to the *TD*-Delaunay triangulation (see Figure 3 for an example of *TD*-Delaunay triangulation). The *TD*-Delaunay triangulation is a variant of the standard Delaunay triangulation where the empty disk property is replaced with an empty homothet of an equilateral triangle (*TD* is an abbreviation for Triangular Distance). Notice that the edges of the Θ_6-graph and the Half-Θ_6-graph are naturally oriented, and thus the graphs can be considered to be directed. However, as in much of the literature on these graphs, we will consider the underlying undirected graph in the design of our routing algorithms.

1.7 Θ_6-routing

As mentioned in Section 1.5, Θ_6-routing always terminates but may have an unbounded routing-ratio (see Figures 1 and 2). The analysis by Bonichon and Marckert [3] on Θ_k-routing for $k = 6$ bounds the expected cost of the Θ_6-routing algorithm by a factor of $\frac{1}{2}\ln 3 + \frac{2}{3} \approx 1.216$ on the ratio of the path length to the Euclidean length when the point set is defined on a Poisson point process with intensity λ tending to infinity. Our techniques establish the same probabilistic bound. However the worst case routing ratio of the Θ_6-routing algorithm is unbounded, see Figure 2. We focus on the probabilistic analysis of routing algorithms that are optimal in the worst-case.

Chew [10] showed that the *TD*-Delaunay triangulation (equivalently the half-Θ_6-graph) is a 2-spanner. His proof is constructive, however, it does not provide an online routing algorithm that successfully routes between every ordered pair of vertices. Without loss of generality, label the cones of the half-Θ_6-graph such that the *TD*-Delaunay triangulation is equivalent to the even half-Θ_6-graph. In this case, positive-routing refers to routing from s to t when T_{st} is even and negative-routing refers to routing from s to t when T_{st} is odd. Chew's algorithm is a positive-routing algorithm and constructs a path from s to t with a routing ratio of 2 when T_{st} is even. Since for every pair of points s and t, either T_{st} is even or T_{ts} is even, Chew's algorithm proves that the *TD*-Delaunay triangulation is a geometric 2-spanner. However, when T_{st} is odd, Chew's routing algorithm simply does not apply to find a route from s to t.

Bose *et al.* [6] addressed the negative-routing case, i.e. when T_{st} is odd, by providing a routing algorithm with routing ratio $\frac{5}{\sqrt{3}} \simeq 2.89$. Surprisingly, they proved that this ratio is optimal for any constant-memory online routing algorithm [6]. This algorithm and the one for the positive case are detailed in Sections 2.3 and 2.4. Since the worst-case optimal routing ratio is 2.89 and worst-case optimal spanning ratio is 2, this is one of the rare known separations between the spanning ratio and the routing ratio of a spanner in the online setting [6]. In essence, even though there exists a path from s to t whose length is at most $2\|st\|$, no constant memory routing algorithm can find this path in the worst-case.

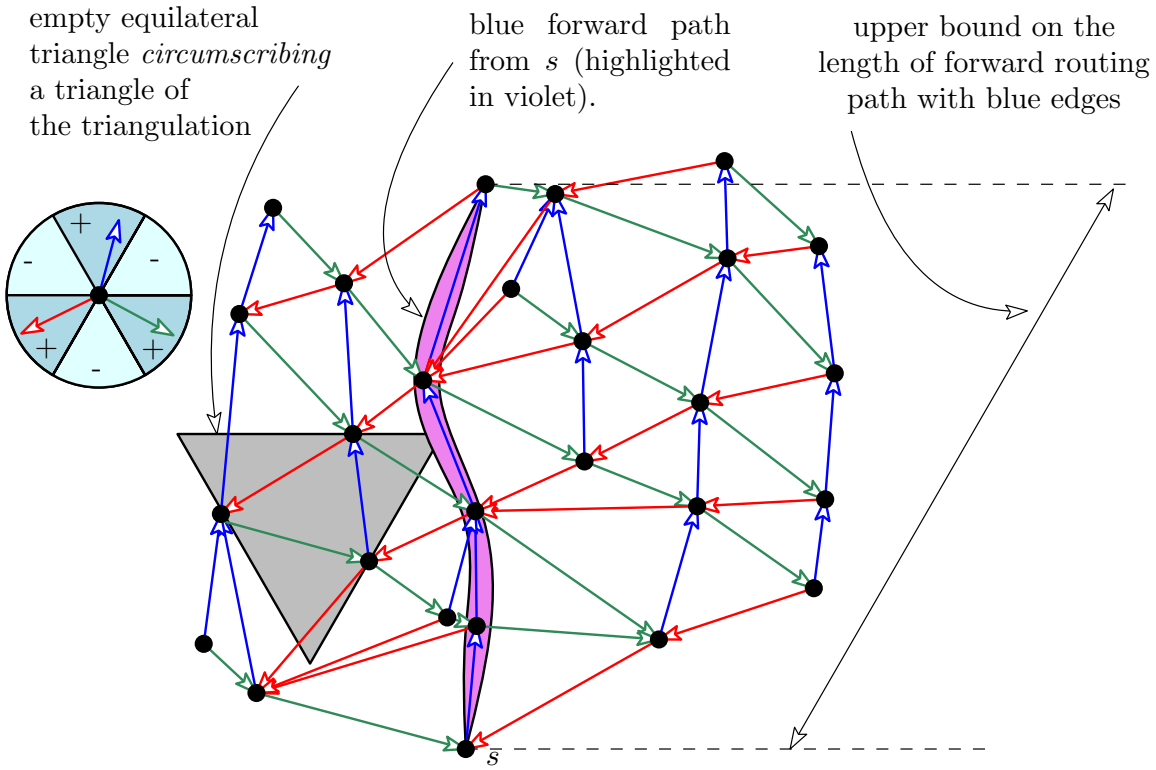

Figure 3: A *TD*-Delaunay triangulation (half-Θ_6-graph).

2 Two Basic Routing Building Blocks on the Half-Θ_6-graph

We introduce two routing modes on the half-Θ_6-graph which serve as building blocks for our routing algorithms that have optimal worst-case behavior. We consider the even half-Θ_6-graph and for ease of reference, we color code the cones C_0, C_2 and C_4 blue, red, and green respectively. The two building blocks are: the *forward-routing phase* and the *side-routing phase*. The routing algorithms proposed by Bose at al. [6] are reformulated in terms of these two building blocks.

2.1 Forward-Routing Phase

Forward-routing consists of only following edges defined by a specific type of cone (i.e., a cone with the same color) until some specified stopping condition is met. For example, suppose the specific cone selected for forward routing is the blue cone. Thus, when forward-routing is invoked at a vertex x, the edge followed is xy where y is the vertex adjacent to x in x's blue cone. If the stopping condition is not met at y, then the next edge followed is yz where z is the vertex adjacent to y in y's blue cone. This process continues until a specified stopping condition is met. A path produced by forward routing consists of edges of the same color since edges are selected from one specific cone as illustrated in Figure 3.

Lemma 1. *Suppose that forward-routing is invoked at a vertex s and ends at a vertex t where $t \in C_0^s$. The length of the path from s to t produced by forward-routing is at most the length of one side of the canonical triangle T_{st} which is $\frac{2}{\sqrt{3}}$ times the length of the orthogonal projection of st onto the bisector of C_0^s.*

Proof. This result follows from the fact that each edge along the path makes a maximum angle of $\frac{\pi}{6}$ with the cone bisector and the path is monotone in the direction of the cone bisector. □

2.2 Side-Routing Phase in the Half-Θ_6-graph

The *side-routing phase* is defined on the half-Θ_6-graph by using the fact that it is the *TD*-Delaunay triangulation, and thus planar. Consider a line ℓ parallel to one of the cone sides. Without loss of generality, we will assume ℓ is horizontal. We call the side of the line that bounds the even cones the *positive side* of ℓ. For a horizontal line, the positive side is below ℓ, and for the lines with slopes $-\sqrt{3}$ and $\sqrt{3}$, respectively, the positive side is above the line. Let $\Delta_1,\ \Delta_2,\ \Delta_3, \ldots$ be an ordered sequence of consecutive triangles of the *TD*-Delaunay triangulation intersecting ℓ. Let $j \in \mathbb{N}^\star$ and let B be the piece of the boundary of the union of the triangles $\Delta_1, \ldots, \Delta_j$ that goes from s, the bottom-left vertex of Δ_1, to t, the bottom-right vertex of Δ_j, below ℓ. Note that B is a path in the half-Θ_6-graph. Side-routing invoked at vertex s along ℓ stopping at t consists of walking from s to t along B (see Figure 4 for an example).

Lemma 2. *Side-routing on the positive side of a line ℓ parallel to a cone boundary invoked at a vertex s and stopped at a vertex t in the half-Θ_6-graph results in a path whose length is bounded by twice the length of the orthogonal projection of st on ℓ. This path only uses edges of two colors and all vertices of the path have their successor of the third color on the other side of ℓ.*

Proof. Without loss of generality, assume ℓ is horizontal and the positive side is below ℓ. Consider the triangles Δ_i, $1 \leq i \leq j$ as defined above. The empty equilateral triangle ∇_i circumscribing Δ_i has a vertex of Δ_i on each of its sides by construction (the ∇_i's are shown in blue in Figure 4 and are inverted w.r.t the canonical triangles). If Δ_i has an edge of the path (i.e., below ℓ) then the vertex on the horizontal side of ∇_i is above the line while the two others are below it. Thus, such an edge of the path goes from the left to the right side of ∇_i. Based on the slopes of the edges of ∇_i, we have the following:
–a– Each edge on the path has a length smaller than twice its horizontal projection. Therefore, summing the lengths of all the projections of the edges gives the claimed bound on the length.
–b– If the slope is negative, the path edge is green and if the slope is positive, the path edge is red.
–c– The blue successor of a vertex u of Δ_i on the lower sides of ∇_i is above ℓ since the part of C_0^u below ℓ is inside ∇_i and thus contains no other points. Therefore, blue edges do not appear on the path. □

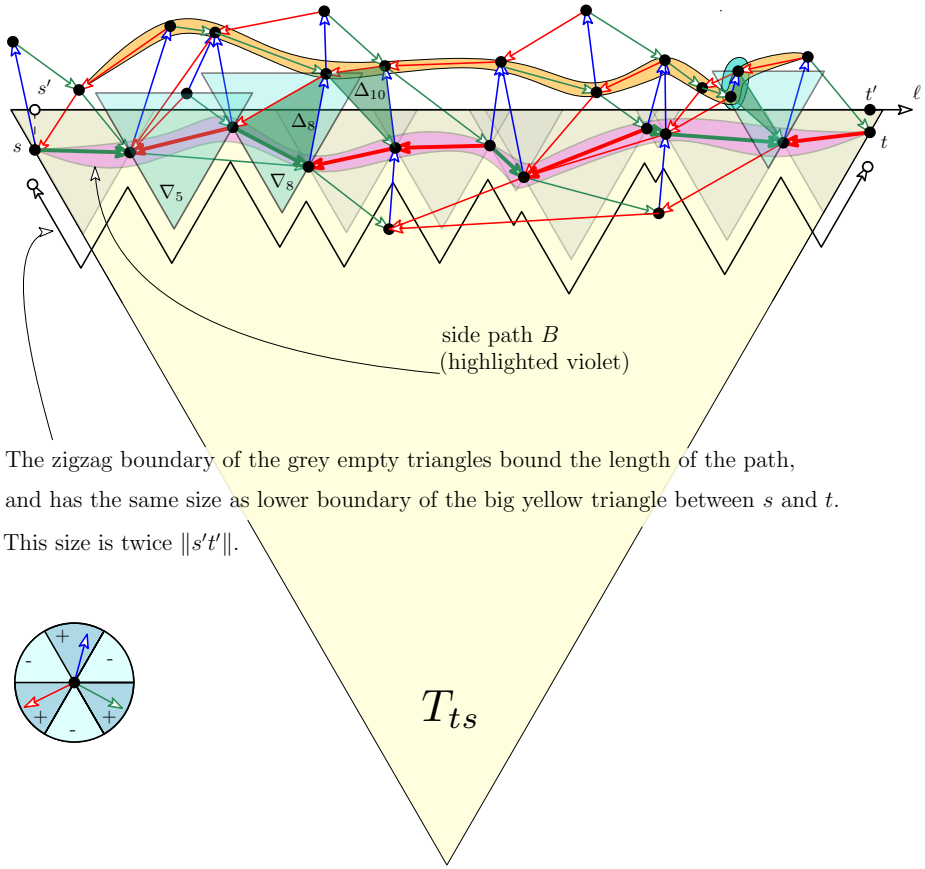

Figure 4: A side path below the horizontal line ℓ.

Notice that on the negative side of the line, we do not have the same properties. For example, the path above ℓ in Figure 4 uses at least one edge of each color (the path above is highlighted in orange, note there is one blue edge circled in blue).

2.3 Positive Routing in the Half-Θ_6-graph (and the Θ_6-graph)

If t is in a positive cone of s, Bose et al. [6] proposed a routing algorithm in the half-Θ_6-graph which they called *positive routing*. This algorithm consists of two phases: a *forward-routing* phase and a *side-routing* phase. The forward-routing phase is invoked with source s and destination t. It produces a path from s to the first vertex u outside the negative cone of t that contains s. The side-routing phase, invoked with source u and destination t, finds a path along the boundary of this negative cone.

For completeness, we give a proof of the following lemma shown in [6].

Lemma 3. *Positive routing has a worst-case routing ratio of* 2.

Proof. Without loss of generality, we assume t is in the positive cone C_0^s and that the

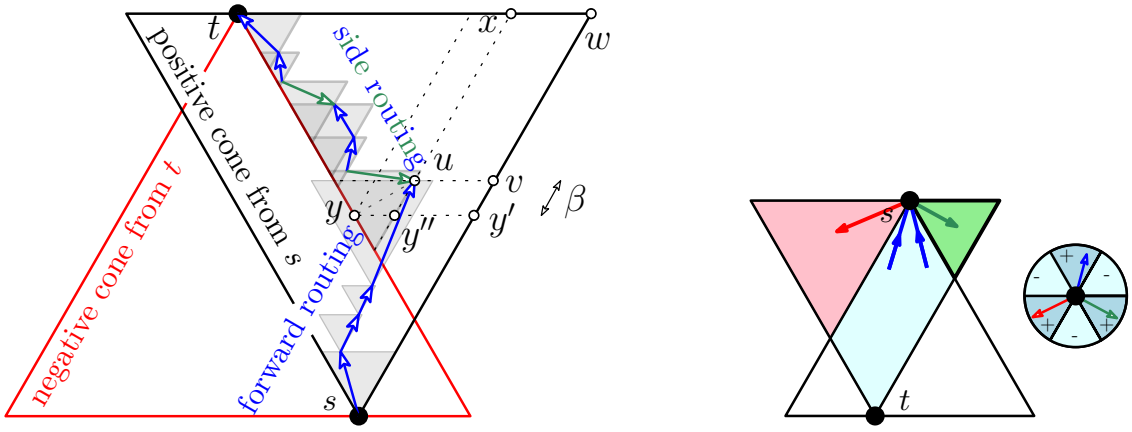

Figure 5: Positive and negative routing schemes [6].

forward routing phase leaves the negative cone from t through its right side. Let u be the last vertex on the forward routing path, and x, v the projections of u on ∂T_{st} (the boundary of T_{st}) parallel to its sides. We define the following three points: y is the perpendicular projection of u on ∂T_{ts}, y' is the horizontal projection of y on ∂T_{st}, and y'' is the horizontal projection of y on line xu (see Figure 5-left). By Lemma 1, the length of the path from s to u is bounded by $\|sv\| = \|sy'\| + \|y'v\|$ and by Lemma 2, the length of the path from u to t is bounded by $2\|yt\| = \|xy''\| + (\|tx\| - \|y''y\|)$. Since the triangle $yy''u$ is isosceles we have $\|yy''\| = \|y''u\| = \|y'v\| =: \beta$. Combining the two paths, the total length is bounded by

$$\|sy'\| + \beta + \|y''x\| + \|tx\| - \beta = \|sy'\| + \|y'w\| + \|xt\| \leq \|sw\| + \|wt\|.$$

Thus, the stretch factor is smaller than $\frac{\|sw\|+\|wt\|}{\|st\|}$. This value can be expressed as $\|sw\| f(\frac{\|wt\|}{\|sw\|})$ where $f : \xi \rightsquigarrow \frac{1+\xi}{\sqrt{\frac{3}{4}+(\xi-\frac{1}{2})^2}}$. Studying the function f and its derivative, this stretch factor is maximal and equal to 2 when $\|wt\| = \|sw\|$ with t in the upper left corner in Figure 5-left ($\xi = \frac{\|wt\|}{\|sw\|} = 1$). □

2.4 Negative Routing in the Half-Θ_6-graph

When t is in a negative cone of s, Bose et al. [6] propose a routing strategy which they call *negative routing*. Without loss of generality, assume that $t \in C_3^s$, which implies that $s \in C_0^t$ (i.e., the blue cone of t in Figure 5-right). Notice that T_{ts} is partitioned by T_{st} into three pieces, the portion contained in C_2^s (red cone of s), C_3^s and C_4^s (green cone of s). We refer to these three zones as: the red triangle, the blue region (which is the intersection of the blue cone of t with T_{st}) and the green triangle. Without loss of generality, we assume that s is to the right of t and the green triangle is smaller than the red one.

Bose et al.'s algorithm [6] can be reformulated in the following way: If neither the green nor the red triangle is empty, negative routing follows the edge from s into the smaller of the two triangles. If one of the green or the red triangle is empty, negative routing uses one step of side routing along the side of the empty triangle (if both are empty, choose the larger of the two).

This process is iterated until t is reached. The worst-case stretch factor of this process is $\frac{5}{\sqrt{3}} \simeq 2.89$ [6].

3 Alternative Negative Routing Algorithms in the Half-Θ_6-graph

In this section, we outline two alternatives to the negative routing algorithm described by Bose et al. [6]. Our algorithms are a little simpler to describe, have the same worst-case routing ratio, and are easier to analyze in the random setting. The lower bound of $\frac{5}{\sqrt{3}} \simeq 2.89$ [6] applies to our alternative negative routing algorithms.

3.1 Memoryless Routing

The first alternative routing algorithm is memoryless. We assume that the algorithm currently is at s and that the destination is t. We outline below how the algorithm decides which edge to follow out of s:

Case 1. If t is in the positive cone C_i^s (i even), take one step of forward-routing in the direction of t.

Case 2. If t is in the negative cone C_i^s (i odd) and the successor u of s in C_{i-1}^s is outside T_{ts} (red triangle empty), take one step of side-routing along the side of T_{ts} crossed by su.

Case 3. If t is in the negative cone C_i^s (i odd) and the successor u of s in C_{i+1}^s is outside T_{ts} (green triangle empty), take one step of side-routing along the side of T_{ts} crossed by su.

Case 4. If t is in the negative cone C_i^s (i odd) and both successors of s in C_{i-1}^s and C_{i+1}^s are inside T_{ts} (green and red triangle non empty), take one step of forward-routing in the direction of the side of T_{ts} incident to t closest to s (go to the green successor of s).

In essence, our strategy differs from the one of Bose et al. [6] in Case 4 where Bose et al. follows a blue edge if one exists.

We remark that when we reach Case 3, we enter a side-routing phase that will continue until t is reached since a side-routing step ensures that at the next iteration side-routing is still applicable.

The same argument holds in Case 2, unless we reach a point u with both successors outside T_{tu}, in which case we follow the other side of T_{tu}.

To summarize, if t is in a positive cone of s, this routing algorithm will produce the path described in Section 2.3 (forward routing followed by side routing) and Lemma 3

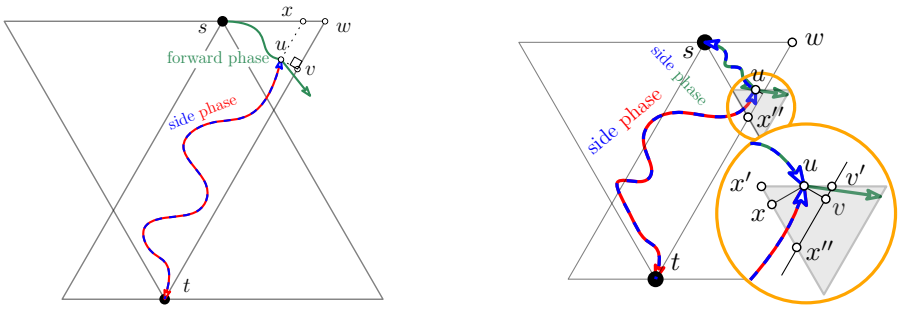

Figure 6: Alternative negative routing strategies. Left: Memoryless (Notations for Lemma 4). Right: Constant-Memory (Notations for Lemma 5).

applies. If t is in a negative cone of s we use a forward phase in the green triangle, until we reach a vertex u whose edge in the green triangle intersects T_{ts} (recall that we assume that the green triangle is the smaller one). At this point, we invoke side-routing from u to t along the boundary of T_{tu} which is the same as the boundary of T_{ts}.

Lemma 4. *Memoryless negative routing has a worst-case routing ratio of* $\frac{5}{\sqrt{3}} \simeq 2.89$.

Proof. Assume without loss of generality $s \in C_0^t$. Let w be the upper right vertex of T_{ts}, v be the orthogonal projection of u on tw and x its projection parallel to tw on sw (see Figure 6-left). By Lemma 1, the path from s to u has length bounded by $\|sx\|$ and by Lemma 2, the path from u to t has length bounded by $2\|vt\|$. Combining the two paths the length is bounded by $\|sx\| + 2\|vt\| \leq \|sw\| + 2\|wt\|$. Thus the stretch is smaller than $\frac{\|sw\|+2\|wt\|}{\|st\|}$. Studying the function $\xi \rightsquigarrow \frac{2+\xi}{\sqrt{\frac{3}{4}+\left(\xi-\frac{1}{2}\right)^2}}$ this stretch factor attains its maximum value of $\frac{5}{\sqrt{3}}$ when s and t lie on a vertical line ($\xi = \frac{\|wt\|}{\|sw\|} = \frac{1}{2}$). □

3.2 Constant-Memory Negative Routing

We propose a second negative routing algorithm that has the same worst-case routing ratio, but we will prove that it has a better average routing ratio. However, it is no longer memoryless since it needs to remember the coordinates of one vertex, namely the source s of the path.

Let x'' be the intersection between T_{ts} and T_{st} closest to s (see Figure 6-right). The idea is to use side-routing from s along sx'' and, just before exiting the green triangle, apply side-routing along $x''t$.

This routing algorithm is identical to the one in the previous subsection, except that we replace Case 4 with the following, where u is the current vertex and s is the origin of the path whose coordinates are kept in memory:

Case 4' If t is in the negative cone C_i^u (i odd) and both successors of u in C_{i-1}^u and C_{i+1}^u are inside T_{tu} (green and red triangles non-empty), take one step of side-routing along the line sx''.

Lemma 5. *Constant-memory negative routing has a worst-case routing ratio of* $\frac{5}{\sqrt{3}} \simeq 2.89$.

Proof. Assume without loss of generality $s \in C_0^t$. Let x' and x be the horizontal and orthogonal projections of u on T_{st}, respectively, and v' and v be the horizontal and orthogonal projections of u on T_{ts}, respectively (see Figure 6-right). By Lemma 2, the path from s to u has length bounded by $2\|sx\|$ and by Lemma 2 again, the path from u to t has length bounded by $2\|vt\|$. Combining the two paths the length is bounded by $2\|sx\| + 2\|vt\| \leq 2\|sx'\| + 2\|x'x\| + 2\|v't\| = 2\|wt\| + 2\|xx'\|$. Since x is the orthogonal projection of u on the side $x'x''$ of the equilateral triangle $x'x''v'$, $\|xx'\|$ is smaller than the half side of the triangle $x'x''v'$ and we get a bound on the path length of $2\|wt\| + 2\|xx'\| \leq 2\|wt\| + 2 \cdot \frac{1}{2}\|x'v'\| \leq 2\|wt\| + \|sw\|$. Therefore the result follows. □

4 Probabilistic Analysis

In this section, we develop tools to analyze the expected routing factor of the routing algorithms defined in Sections 2 and 3 from a probabilistic point of view. In Section 4.1, we analyze the expected routing factor of a forward-routing phase. In Section 4.2, we analyze the expected routing factor of a side-routing phase. Then, using these results, we will analyze in Section 5 the expected routing factor of three different routing algorithms.

4.1 Routing Factor of a Forward-Routing Phase

Let X be a Poisson point process with intensity λ and consider the half-Θ_6-graph defined on $X \cup \{s\}$, where s is the origin. Let $p_0 = s$ and p_{i+1} be the successor of p_i in the half-Θ_6-graph using the cone $C_0^{p_i}$. We define $\tau_1 := \frac{\sqrt{3}}{12}(3\ln 3 + 4)$.

Lemma 6. *Let* $A > 0$ *and* $\alpha = 2\sqrt{A}\lambda^{-\frac{1}{4}}\sqrt{\log(2A\sqrt{\lambda})}$. *Consider a forward-routing phase in the upward direction, starting at the origin until it crosses the line* $y = A$. *The expected routing factor of this phase is* $\tau_1 + O\left(\frac{\alpha}{A}\right)$. *With probability greater than* $1 - \frac{17}{5}A^{-\frac{1}{2}}\lambda^{-\frac{1}{4}}$, *the endpoint of this phase lies in* $[-\alpha, \alpha] \times [A + 2\alpha]$.

Proof. Let $p_0 = s, p_1, p_2, \ldots, p_n$ denote the vertices of the forward path. We consider the following random variables: L_i is the Euclidean length of $p_{i-1}p_i$, $L_{x,i}$ is the signed length of the horizontal projection of $p_{i-1}p_i$, and $L_{y,i}$ is the length of the vertical projection of $p_{i-1}p_i$.

Since $C_0^{p_i}$ does not intersect $T_{p_{i-1}p_i}$, for different values of i, these variables are independent and identically distributed. Thus, when there is no ambiguity we denote them by L, L_x, and L_y, respectively.

If p is the upward successor of s in the half-Θ_6-graph, then the canonical triangle T_{sp} is empty. Using polar coordinates where $p = r(\cos\phi, \sin\phi)$, the area of T_{sp}, denoted by $\mathrm{Area}(T_{sp})$, is $\frac{1}{\sqrt{3}}r^2\sin^2\phi$. Thus $\mathbb{P}[T_{sp} \text{ is empty}] = e^{-\lambda \mathrm{Area}(\mathcal{T})} = e^{-\lambda\frac{1}{\sqrt{3}}r^2\sin^2\phi}$, from which the following bound follows[1] (see Figure 7).

[1] Integral computations are available as a Maple worksheet with this paper.

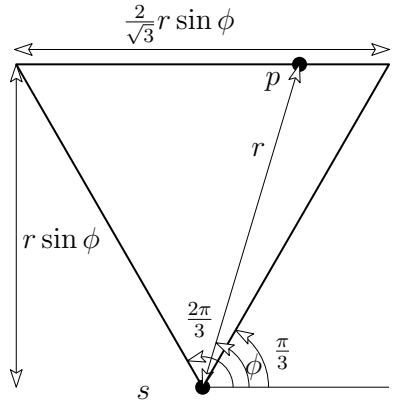

Figure 7: Illustration of Lemma 6.

$$\begin{aligned}
\mathbb{E}[L] = \mathbb{E}[L_1] &= \lambda \int_{p \in BlueCone} \mathbb{P}[T_{sp} \cap X = \emptyset] \|p\| \mathrm{d}p \\
&= \lambda \int_0^{\infty} \int_{\frac{\pi}{3}}^{\frac{2\pi}{3}} e^{-\lambda \frac{1}{\sqrt{3}} r^2 \sin^2\phi} r^2 \mathrm{d}\phi \mathrm{d}r \\
&= \frac{1}{\sqrt{\lambda}} \frac{1}{2} 3^{-\frac{1}{4}} \sqrt{\pi} (1 + \tfrac{3}{4} \ln 3) \simeq \frac{1.228}{\sqrt{\lambda}}.
\end{aligned}$$

We define $\mu := \frac{1}{2} 3^{-\frac{1}{4}} \sqrt{\pi}(1 + \frac{3}{4} \ln 3) \simeq 1.228$. By symmetry, the expected length of the horizontal projection of an edge is $\mathbb{E}[L_x] = 0$. For the vertical projection, we get

$$\begin{aligned}
\mathbb{E}[L_y] &= \lambda \int_{p \in BlueCone} \mathbb{P}[T_{sp} \cap X = \emptyset] y_p \mathrm{d}p \\
&= \lambda \int_0^{\infty} \int_{\frac{\pi}{3}}^{\frac{2\pi}{3}} e^{-\lambda \frac{1}{\sqrt{3}} r^2 \sin^2\phi} r^2 \sin\phi \ \mathrm{d}\phi \mathrm{d}r \\
&= \frac{1}{\sqrt{\lambda}} \frac{1}{2} 3^{\frac{1}{4}} \sqrt{\pi} \simeq \frac{1.166}{\sqrt{\lambda}}.
\end{aligned}$$

We define $\mu_y := \frac{1}{2} 3^{\frac{1}{4}} \sqrt{\pi} \simeq 1.166$. We prove that after n steps, the length of the path from p_0 to p_n is approximately $n\,\mathbb{E}[L] = \frac{n\mu}{\sqrt{\lambda}}$, while the position of p_n is close to $(n\,\mathbb{E}[L_x], n\,\mathbb{E}[L_y]) = (0, \frac{n\mu_y}{\sqrt{\lambda}})$. This gives a routing factor around $\frac{\mu}{\mu_y}$. Formally, let us first compute the higher order moments and define as follows the constants σ, σ_x, σ_y, ρ, ρ_x, and ρ_y.

$$\begin{aligned}
\mathbb{E}\left[L^2\right] &= \lambda\int_0^\infty\int_{\frac{\pi}{3}}^{\frac{2\pi}{3}} e^{-\lambda\frac{1}{\sqrt{3}}r^2\sin^2\phi}r^3\mathrm{d}\phi\mathrm{d}r = \tfrac{10\sqrt{3}}{9\lambda} = \tfrac{\sigma^2}{\lambda},\\
\mathbb{E}\left[L_x^2\right] &= \lambda\int_0^\infty\int_{\frac{\pi}{3}}^{\frac{2\pi}{3}} e^{-\lambda\frac{1}{\sqrt{3}}r^2\sin^2\phi}r^3\cos^2\phi\ \mathrm{d}\phi\mathrm{d}r = \tfrac{\sqrt{3}}{9\lambda} = \tfrac{\sigma_x^2}{\lambda},\\
\mathbb{E}\left[L_y^2\right] &= \lambda\int_0^\infty\int_{\frac{\pi}{3}}^{\frac{2\pi}{3}} e^{-\lambda\frac{1}{\sqrt{3}}r^2\sin^2\phi}r^3\sin^2\phi\ \mathrm{d}\phi\mathrm{d}r = \tfrac{\sqrt{3}}{\lambda} = \tfrac{\sigma_y^2}{\lambda},\\
\mathbb{E}\left[L^3\right] &= \lambda\int_0^\infty\int_{\frac{\pi}{3}}^{\frac{2\pi}{3}} e^{-\lambda\frac{1}{\sqrt{3}}r^2\sin^2\phi}r^4\mathrm{d}\phi\mathrm{d}r = \tfrac{(27\ln 3+68)\sqrt{\pi\sqrt{3}}}{64\lambda\sqrt{\lambda}} = \tfrac{\rho}{\lambda\sqrt{\lambda}},\\
\mathbb{E}\left[|L_x|^3\right] &= \lambda\int_0^\infty\int_{\frac{\pi}{3}}^{\frac{2\pi}{3}} e^{-\lambda\frac{1}{\sqrt{3}}r^2\sin^2\phi}r^4|\cos^3\phi|\mathrm{d}\phi\mathrm{d}r = \tfrac{3^{\frac{1}{4}}\sqrt{\pi}}{16\lambda\sqrt{\lambda}} = \tfrac{\rho_x}{\lambda\sqrt{\lambda}},\\
\mathbb{E}\left[L_y^3\right] &= \lambda\int_0^\infty\int_{\frac{\pi}{3}}^{\frac{2\pi}{3}} e^{-\lambda\frac{1}{\sqrt{3}}r^2\sin^2\phi}r^4\sin^3\phi\ \mathrm{d}\phi\mathrm{d}r = \tfrac{3^{\frac{7}{4}}\sqrt{\pi}}{4\lambda\sqrt{\lambda}} = \tfrac{\rho_y}{\lambda\sqrt{\lambda}}.
\end{aligned}$$

For identically independently distributed variables X_i, each of which has expected value $\mu_\star$, standard deviation $\sigma_\star$, and third moment $\rho_\star$, the *central limit theorem* states that

$$\lim_{n\to\infty}\mathbb{P}\left[\left|\sum_{i=1}^n X_i - n\mu_\star\right| \geq a\sigma_\star\sqrt{n}\right] = \left(1-\operatorname{erf}\left(\tfrac{a}{\sqrt{2}}\right)\right),$$

when n tends to infinity, where *erf* is the error function $erf(x) := \frac{1}{\sqrt{\pi}}\int_{-x}^{x} e^{-t^2}dt \in [-1,1]$. Then, the *Berry-Esseen inequality* specifies the rate of convergence [13]:

$$\left|\mathbb{P}\left[\left|\sum_{i=1}^n X_i - n\mu_\star\right| \geq a\sigma_\star\sqrt{n}\right] - \left(1-\operatorname{erf}\left(\tfrac{a}{\sqrt{2}}\right)\right)\right| \leq 0.4785\frac{\rho_\star}{\sigma_\star^3\sqrt{n}} \leq \frac{\rho_\star}{2\sigma_\star^3\sqrt{n}}. \tag{1}$$

Applying Equation (1) to L_y with $a=\sqrt{\log n}$ and using $\operatorname{erf}(t) \geq 1-\frac{e^{-t^2}}{\sqrt{\pi}t}$ for $t\geq 0$, we get

$$\begin{aligned}
\mathbb{P}\left[\left|\sum_{i=1}^n L_{y,i} - n\tfrac{\mu_y}{\sqrt{\lambda}}\right| \geq \tfrac{\sigma_y}{\sqrt{\lambda}}\sqrt{n\log n}\right] &\leq \left(1-\operatorname{erf}\left(\sqrt{\tfrac{\log n}{2}}\right)\right) + \frac{\frac{\rho_y}{\lambda\sqrt{\lambda}}}{2\left(\frac{\sigma_y}{\sqrt{\lambda}}\right)^3\sqrt{n}}\\
&= \left(1-\operatorname{erf}\left(\sqrt{\tfrac{\log n}{2}}\right)\right) + \frac{\rho_y}{2\sigma_y^3\sqrt{n}}.\\
&\leq \left(1-\left(1-\frac{\frac{1}{\sqrt{n}}}{\sqrt{\pi}\sqrt{\frac{\log n}{2}}}\right)\right) + \frac{\rho_y}{2\sigma_y^3\sqrt{n}}\\
&= \frac{\sqrt{2}}{\sqrt{\pi}\sqrt{n\log n}} + \frac{\rho_y}{2\sigma_y^3\sqrt{n}}.
\end{aligned} \tag{2}$$

Let $n_A = \frac{(A+\alpha)\sqrt{\lambda}}{\mu_y}$. For λ big enough we have $\alpha = 2\sqrt{A}\lambda^{-\frac{1}{4}}\sqrt{\log(2A\sqrt{\lambda})}$ smaller than A, from which the following bound follows

$$\begin{aligned}
\tfrac{\sigma_y}{\sqrt{\lambda}}\sqrt{n_A \log n_A} &= \tfrac{\sigma_y}{\sqrt{\lambda}}\sqrt{\frac{(A+\alpha)\sqrt{\lambda}}{\mu_y}\log\frac{(A+\alpha)\sqrt{\lambda}}{\mu_y}} \\
&\leq \tfrac{\sigma_y}{\sqrt{\lambda}}\sqrt{\frac{2A\sqrt{\lambda}}{\mu_y}\log\frac{2A\sqrt{\lambda}}{\mu_y}} \\
&= \sigma_y\sqrt{\tfrac{2}{\mu_y}}\sqrt{A}\lambda^{-\frac{1}{4}}\sqrt{\log(\tfrac{2}{\mu_y}A\sqrt{\lambda})} \\
&\leq 3^{\frac{1}{4}}\sqrt{\frac{2}{\frac{1}{2}3^{\frac{1}{4}}\sqrt{\pi}}}\sqrt{A}\lambda^{-\frac{1}{4}}\sqrt{\log(2A\sqrt{\lambda})} \\
&\leq 1.73\sqrt{A}\lambda^{-\frac{1}{4}}\sqrt{\log(2A\sqrt{\lambda})} \\
&\leq 2\sqrt{A}\lambda^{-\frac{1}{4}}\sqrt{\log(2A\sqrt{\lambda})} = \alpha.
\end{aligned}$$

Combining this with Equation (2), we get:

$$\begin{aligned}
\mathbb{P}\left[\left|\sum_{i=1}^{n_A} L_{y,i} - A - \alpha\right| \geq \alpha\right] &\leq \mathbb{P}\left[\left|\sum_{i=1}^{n_A} L_{y,i} - A - \alpha\right| \geq \tfrac{\sigma_y}{\sqrt{\lambda}}\sqrt{n_A \log n_A}\right] \\
&\leq \frac{\sqrt{2}}{\sqrt{\pi}\sqrt{n_A \log n_A}} + \frac{\rho_y}{2\sigma_y^3\sqrt{n_A}} \\
&\leq \tfrac{1}{\sqrt{n_A}}\left(\sqrt{\tfrac{2}{\pi}} + \frac{\rho_y}{2\sigma_y^3}\right) \\
&\leq \sqrt{\tfrac{\mu_y}{A\sqrt{\lambda}}}\left(\sqrt{\tfrac{2}{\pi}} + \frac{\rho_y}{2\sigma_y^3}\right).
\end{aligned}$$

Now we look at the x-coordinate of p_{n_A}, which is $\sum_{i=1}^{n_A} L_{x,i}$. Using Equation (1) again, we have

$$\mathbb{P}\left[\left|\sum_{i=1}^{n_A} L_{x,i}\right| \geq \tfrac{\sigma_x}{\sqrt{\lambda}}\sqrt{n_A \log n_A}\right] \leq \left(1 - \operatorname{erf}\left(\sqrt{\tfrac{\log n_A}{2}}\right)\right) + \frac{\rho_x}{2\sigma_x^3\sqrt{n_A}}.$$

Substituting for n_A and since $\alpha \geq \frac{\sigma_x}{\sqrt{\lambda}}\sqrt{n_A \log n_A}$ (for λ big enough), we get

$$\mathbb{P}\left[\left|\sum_{i=1}^{n_A} L_{x,i}\right| \geq \alpha\right] \leq \sqrt{\tfrac{\mu_y}{A\sqrt{\lambda}}}\left(\sqrt{\tfrac{2}{\pi}} + \frac{\rho_x}{2\sigma_x^3}\right).$$

Thus,

$$\begin{aligned}\mathbb{P}\left[p_{n_A} \in [-\alpha,\alpha]\times[A,A+2\alpha]\right] &\leq \sqrt{\tfrac{\mu_y}{A\sqrt{\lambda}}}\left(2\sqrt{\tfrac{2}{\pi}}+\frac{\rho_x}{2\sigma_x^3}+\frac{\rho_y}{2\sigma_y^3}\right)\\ &= \sqrt{\tfrac{1}{2}3^{\frac{1}{4}}\sqrt{\pi}}\left(2\sqrt{\tfrac{2}{\pi}}+\frac{\frac{3^{\frac{1}{4}}\sqrt{\pi}}{16}}{2(3^{-\frac{3}{4}})^3}+\frac{\frac{3^{\frac{7}{4}}\sqrt{\pi}}{4}}{2(3^{\frac{1}{4}})^3}\right)A^{-\frac{1}{2}}\lambda^{-\frac{1}{4}}\\ &\leq 3.4A^{-\frac{1}{2}}\lambda^{-\frac{1}{4}}.\end{aligned}$$

Let i_A be the smallest i such that $y_{p_i} \geq A$. If $i_A \leq n_A$, then $y_{p_{n_A}} \geq y_{p_{i_A}} \geq A$. We can bound $\|p_0p_{i_A}\| > A$ and the path length $\sum_{i=1}^{i_A} L_i \leq \sum_{i=1}^{n_A} L_i \leq n_A\,\mathbb{E}[L]$. If $i_A > n_A$, we can bound the routing factor by $\frac{2}{\sqrt{3}}$ using Lemma 1 (worst-case analysis of forward-routing). We get:

$$\begin{aligned}\mathbb{E}\left[\frac{\sum_{i=1}^{i_A} L_i}{\|p_0p_{i_A}\|}\right] &\leq \mathbb{P}[i_A \leq n_A]\,\mathbb{E}\left[\frac{\sum_{i=1}^{i_A} L_i}{\|p_0p_{i_A}\|}\right] + \mathbb{P}[i_A > n_A]\,\frac{2}{\sqrt{3}}\\ &\leq 1\cdot\frac{n_A\cdot\mathbb{E}[L]}{A}+\sqrt{\tfrac{\mu_y}{A\sqrt{\lambda}}}\left(\sqrt{\tfrac{2}{\pi}}+\frac{\rho_y}{2\sigma_y^3}\right)\frac{2}{\sqrt{3}}\\ &\leq \frac{\frac{(A+\alpha)\sqrt{\lambda}}{\mu_y}\frac{\mu}{\sqrt{\lambda}}}{A}+\sqrt{\tfrac{4\mu_y}{3}}\left(\sqrt{\tfrac{2}{\pi}}+\frac{\rho_y}{2\sigma_y^3}\right)\frac{1}{A\sqrt{\lambda}}\\ &= \frac{\mu}{\mu_y}+O\left(A^{-\frac{1}{2}}\lambda^{-\frac{1}{4}}\sqrt{\log(2A\sqrt{\lambda})}\right)+O\left(\frac{1}{A\sqrt{\lambda}}\right)\\ &= \frac{\sqrt{3}}{12}(3\ln 3+4)+O\left(A^{-\frac{1}{2}}\lambda^{-\frac{1}{4}}\sqrt{\log(2A\sqrt{\lambda})}\right).\end{aligned}$$

□

Remark 7. Θ_6*-routing in a dense point set runs in two phases, while the target is* really inside *a cone, it has a similar behavior to forward-routing. When it reaches the boundary of a cone, it switches between two cones (one odd and one even) and a similar analysis can be done. However, measuring the progress along the cone boundary instead of the cone bisector gives an expected routing factor of* $\tau_2 := \frac{2}{\sqrt{3}}\tau_1 = \frac{1}{6}(3\ln 3+4)$.

4.2 Routing Factor of a Side-Routing Phase

To analyze the expected routing factor of a side-routing phase, we consider the sum of the lengths of all of its edges. Then, we use the *Slivnyak-Mecke formula* (refer to [18, Corollary 3.2.3]) to transform this sum into an integral, from which we get the following lemma.

Lemma 8. *Consider a side-routing phase in the horizontal direction, starting at the origin until it reaches the line* $x = A$. *The expected routing factor of this phase is* $\tau_2 + O\left(A^{-1}\lambda^{-\frac{1}{2}}\right)$ *with* $\tau_2 := \frac{2}{\sqrt{3}}\tau_1$.

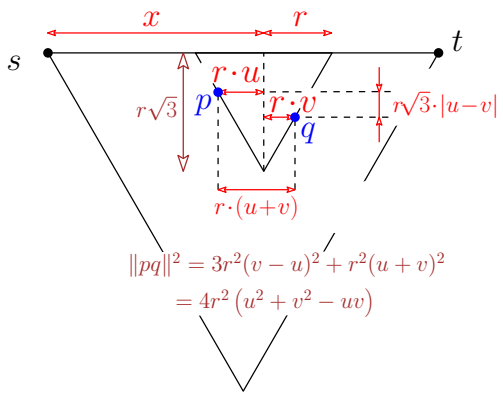

Figure 8: Illustration of Lemma 8.

Proof. Up to a scaling factor of A on the lengths and A^2 on the density, we can assume without loss of generality that $A = 1$. Let $\nabla(p, q)$ denote the equilateral triangle with p on its left side, q on its right side and its horizontal side supported by the x-axis (the triangle is below the x-axis). Let $s = (0, 0)$ and $t = (1, 0)$. The following analysis sums up the lengths of the edges pq of the half-Θ_6-graph of X, where $p, q \in \nabla(s, t)$ and $\nabla(p, q)$ is empty. Depending on the situation, we may have to add an edge that connects the path to s and/or to t if these points have been added to the point set. We may also have to add an edge that crosses the boundary of $\nabla(s, t)$. In any case, the expected length of these edges is $O\left(\frac{1}{\sqrt{\lambda}}\right)$, which is negligible. Using Slivnyak-Mecke formula, we have

$$\begin{aligned}\mathbb{E}\left[\text{length}\right] &= \mathbb{E}\left[\sum_{p,q\in X\cap\nabla(s,t)} \mathbb{1}_{[\nabla(p,q)\cap X=\emptyset]}\|pq\|\right] + O\left(\frac{1}{\sqrt{\lambda}}\right)\\ &= \lambda^2 \iint_{p,q\in\nabla(s,t)} e^{-\lambda \operatorname{Area}(\nabla(p,q))}\|pq\|\mathrm{d}q\mathrm{d}p + O\left(\frac{1}{\sqrt{\lambda}}\right).\end{aligned}$$

To solve this integral, we consider a variable substitution, where points p and q, instead of being defined by their Cartesian coordinates, are defined by their triangle $\nabla(p, q)$ (described by x and r) and parameters to place p and q on the relevant sides of $\nabla(p, q)$. More formally, the variable substitution Φ is define as (see Figure 8):

$$\begin{aligned}&\Phi : \mathbb{R} \times \mathbb{R}_{\geq 0} \times [0, 1]^2 \to (\mathbb{R}^2)^2\\ &(x, r, u, v) \rightsquigarrow \left(\begin{pmatrix} x - ru \\ -\sqrt{3}r(1-u)\end{pmatrix}, \begin{pmatrix} x + rv \\ -\sqrt{3}r(1-v)\end{pmatrix}\right).\end{aligned}$$

The Jacobian of Φ is

$$\det(J_\Phi) = \begin{vmatrix} 1 & -u & -r & 0 \\ 0 & -\sqrt{3}(1-u) & \sqrt{3}r & 0 \\ 1 & v & 0 & r \\ 0 & -\sqrt{3}(1-v) & 0 & \sqrt{3}r \end{vmatrix} = -6r^2.$$

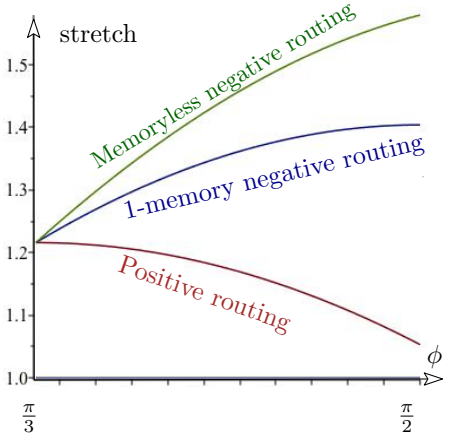

Figure 9: Illustration of Theorem 9. The expected routing factor of the different algorithms as a function of the angle of st with the x-axis.

This variable substitution yields

$$\begin{aligned}
\mathbb{E}\left[\text{length}\right] &= \lambda^2 \int_0^1 \int_0^{\min(x,1-x)} \int_0^1 \int_0^1 e^{-\lambda r^2\sqrt{3}} r 2\sqrt{u^2+v^2-uv} \cdot \left|\det(J_\Phi)\right| \mathrm{d}v\mathrm{d}u\mathrm{d}r\mathrm{d}x \\
&= \lambda^2 \left(\int_0^1 \int_0^{\min(x,1-x)} e^{-\lambda r^2\sqrt{3}} r \cdot 6 \cdot r^2 \mathrm{d}r\mathrm{d}x \right) \left(2\int_0^1 \int_0^1 \sqrt{u^2+v^2-uv}\,\mathrm{d}v\mathrm{d}u \right) \\
&= \lambda^2 \left(2 \cdot \int_0^{\frac{1}{2}} \int_0^x 6e^{-\lambda r^2\sqrt{3}} \cdot r^3 \mathrm{d}r\mathrm{d}x \right) \left(4\int_0^1 \int_0^u \sqrt{u^2+v^2-uv}\,\mathrm{d}v\mathrm{d}u \right) \\
&= 8\lambda^2 \cdot \frac{1}{4\lambda^2} \left(2 + 3^{\frac{3}{4}}\sqrt{\pi}\ \mathrm{erf}\left(\tfrac{1}{2}\sqrt{\lambda}\sqrt[4]{3}\right) \tfrac{1}{\sqrt{\lambda}} + e^{-\frac{1}{4}\lambda\sqrt{3}} \right) \left(\tfrac{1}{6} + \tfrac{\ln 3}{8} \right) \\
&= \tfrac{1}{6}(3\ln 3 + 4) + O\left(\tfrac{1}{\sqrt{\lambda}}\right) = \tau_2 + O\left(\tfrac{1}{\sqrt{\lambda}}\right) \simeq 1.2160 + O\left(\tfrac{1}{\sqrt{\lambda}}\right).
\end{aligned}$$

□

Observe that side-routing and Θ_6-routing in the neighbourhood of a cone boundary have the same expected routing factor. This means that in side-routing, the expected progress made by an edge of the path behaves as if it was independent from the previous edges, although this is not the case a priori.

5 Wrap Up

In this section, we analyze the routing algorithms defined in Sections 2 and 3 using the tools from Section 4. Recall that these algorithms are made up of two phases. The analysis is based on the following two ingredients. First, the splitting point between the two phases belongs to a small region of the plane with high probability. Second, the two phases of each routing algorithm can be analyzed separately.

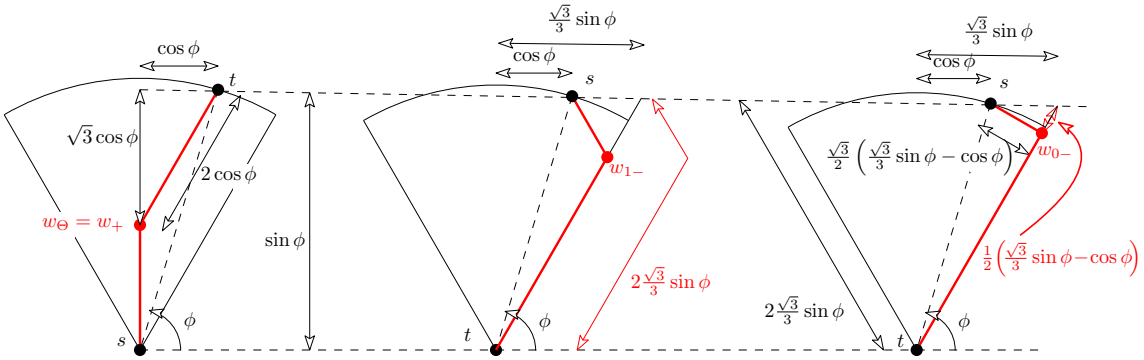

Figure 10: Illustration of the proof of Theorem 9.

Theorem 9. *Let X be a Poisson point process with intensity λ. Consider the positive and the two alternative negative routing algorithms on the half-Θ_6-graph defined on $X \cup \{s,t\}$, where s and t are two points at unit distance. Figure 9 presents a graph of the expected routing factors of the different routing algorithms as λ tends to ∞ in terms of ϕ the angle of st with the horizontal axis. The following table shows the expected stretch for a given ϕ and also the maximum for all ϕ and average on ϕ.*

Routing	$\mathbb{E}$ [routing factor] (ϕ)	$\max_{s,t} \mathbb{E}$ [routing factor]	$\mathbb{E}_{s,t}[\mathbb{E}$ [routing factor]]
Positive routing	$\tau_1 \left(\sin\phi + \frac{1}{\sqrt{3}}\cos\phi\right)$	$\frac{2}{\sqrt{3}}\tau_1 \simeq 1.2160$	$\frac{2\sqrt{3}}{\pi}\tau_1 \simeq 1.1612$
Constant-memory	$\frac{4}{3}\tau_1 \sin\phi$	$\frac{4}{3}\tau_1 \simeq 1.4041$	$\frac{4}{\pi}\tau_1 \simeq 1.3408$
Memoryless	$\tau_1 \left(\frac{3}{2}\sin\phi - \frac{\sqrt{3}}{6}\cos\phi\right)$	$\frac{3}{2}\tau_1 \simeq 1.5800$	$\frac{6-\sqrt{3}}{\pi}\tau_1 \simeq 1.4306$

$$\tau_1 := \frac{1}{4\sqrt{3}}(3\ln 3 + 4)$$

Proof. Each of the three routing algorithms we consider is the combination of two phases, each of which can be a forward-routing phase or a side-routing phase. These two phases articulate at a splitting point $w \in X$, where the routing algorithm changes from one phase to the next. The actual splitting point is close to an *ideal splitting point* $w_\star$ (which does not belong to X) that depends only on s, t and the routing algorithm. Therefore, the analysis of each of the three routing algorithms follows the same scheme: let the expected routing factor of the two phases be τ and τ', respectively. Since the point w is close to $w_\star$ with high probability, the expected routing factor of the routing algorithm is $\frac{\|sw_\star\|\tau + \|w_\star t\|\tau'}{\|st\|}$ (see Figure 10-left).

Positive routing consists of a forward-routing phase from s to w followed by a side phase from w to t (see Section 2.3). If we fix $s = (0,0)$ and $t = (\cos\phi, \sin\phi)$ for $\phi \in [\frac{\pi}{3}, \frac{\pi}{2}]$, the splitting point becomes $w_+ = \left(0, 1 - \sqrt{3}\sin\phi\right)$. With probability less than $4\lambda^{-\frac{1}{4}}$ we bound the routing factor by the worst case bounds, i.e., between 1 and 2. Otherwise, we use the fact that the splitting point w is close to w_+. Let $A = 1 - \sqrt{3}\cos\phi$ and $\alpha = 2\sqrt{A}\lambda^{-\frac{1}{4}}\sqrt{\log(2A\sqrt{\lambda})}$. We have $\|w_+ w\| < 3\alpha$ with probability greater than $1 - 4\lambda^{-\frac{1}{4}}$, by Lemma 6. Moreover, by Lemmas 6 and 8, $\tau = \tau_1$ and $\tau' = \tau_2$. Thus, the expected

routing factor $\psi := \mathbb{E}\left[\frac{\tau_1\|sw\|+\tau_2\|wt\|}{\|st\|}\right]$ satisfies

$$\begin{aligned}
\psi &\leq \mathbb{P}\left[\|w_+w\| \geq \alpha\right] 2 + (1 - \mathbb{P}\left[\|w_+w\| \geq \alpha\right]) \left(\frac{\tau_1\|sw_+\| + \tau_2\|w_+t\|}{\|st\|} + \frac{(\tau_1+\tau_2)\|ww_+\|}{\|st\|}\right) \\
\psi &\geq \mathbb{P}\left[\|w_+w\| \geq \alpha\right] 1 + (1 - \mathbb{P}\left[\|w_+w\| \geq \alpha\right]) \left(\frac{\tau_1\|sw_+\| + \tau_2\|w_+t\|}{\|st\|} - \frac{(\tau_1+\tau_2)\|ww_+\|}{\|st\|}\right) \\
\psi &= \frac{\tau_1\|sw_+\| + \tau_2\|w_+t\|}{\|st\|} + O(\lambda^{-\frac{1}{4}}\sqrt{\log\lambda}).
\end{aligned}$$

And we get as a limit when $\lambda \to \infty$:

$$\frac{\tau_1\|sw_+\| + \tau_2\|w_+t\|}{\|st\|} = \tau_1\left(\sin\phi - \sqrt{3}\cos\phi\right) + \tau_2 2\cos\phi = \tau_1\left(\sin\phi + \tfrac{1}{\sqrt{3}}\cos\phi\right),$$

whose maximum value is $\frac{2}{\sqrt{3}}\tau_1 \simeq 1.2160$, obtained for $\phi = \frac{\pi}{3}$. Considering any direction in a positive cone equally likely, we average on ϕ to get an expected value. By symmetry, we only average on $\phi \in [\frac{\pi}{3}, \frac{\pi}{2}]$ and get:

$$\frac{6}{\pi}\int_{\frac{\pi}{3}}^{\frac{\pi}{2}} \tau_1\left(\sin\phi + \tfrac{1}{\sqrt{3}}\cos\phi\right) \mathrm{d}\phi = \tfrac{2\sqrt{3}}{\pi}\tau_1 \simeq 1.1612.$$

Remark 10. *By Remark 7, a similar analysis can be done for Θ_6-routing with the same splitting point, giving the same expected factor as with positive routing.*

Constant-memory routing consists of a side-routing phase from s to w, followed by a side-routing phase from w to t (see Section 3.2). If we fix $t = (0,0)$ and $s = (\cos\phi, \sin\phi)$ for $\phi \in [\frac{\pi}{3}, \frac{\pi}{2}]$, the splitting point becomes $w_{1-} = \left(\frac{\sqrt{3}}{3}\sin\phi - \cos\phi\right)(1, \frac{\sqrt{3}}{3})$. By Lemma 8, $\|w_{1-}w\| < O\left(\lambda^{-\frac{1}{2}}\right)$ with high probability. Moreover, by Lemma 8, $\tau = \tau' = \tau_2$. Thus, the expected routing factor when $\lambda \to \infty$ is (see Figure 10-center).

$$\tau_2\frac{\|sw_{1-}\| + \|w_{1-}t\|}{\|st\|} = \tau_2 2\frac{\sqrt{3}}{3}\sin\phi = \tfrac{4}{3}\tau_1\sin\phi,$$

whose maximum is $\frac{4}{3}\tau_1 \simeq 1.4041$, obtained for $\phi = \frac{\pi}{2}$. Averaging on ϕ yields an expected value of

$$\frac{6}{\pi}\int_{\frac{\pi}{3}}^{\frac{\pi}{2}} \tfrac{4}{3}\tau_1\sin\phi \,\mathrm{d}\phi = \tfrac{4}{\pi}\tau_1 \simeq 1.3408.$$

Memoryless negative routing is made of a forward-routing phase from s to w, followed by a side phase from w to t (see Section 3.1). If we fix $t = (0,0)$ and $s = (\cos\phi, \sin\phi)$ for $\phi \in [\frac{\pi}{3}, \frac{\pi}{2}]$, the splitting point becomes w_{0-} the orthogonal projection of s on the side of T_{ts} (see Figure 10-right). By Lemma 6, $\|w_{0-}w\| < O\left(\lambda^{-\frac{1}{4}}\sqrt{\log\lambda}\right)$ with probability greater than $1 - O(\lambda^{-\frac{1}{4}})$. Moreover, by Lemmas 6 and 8, $\tau = \tau_1$ and $\tau' = \tau_2$ Thus, the limit of the

expected routing factor is

$$\begin{aligned}\frac{\tau_1\|sw_{0-}\|+\tau_2\|w_{0-}t\|}{\|st\|} &= \tau_1\left(\tfrac{\sqrt{3}}{2}\left(\frac{\sqrt{3}}{3}\sin\phi-\cos\phi\right)\right)+\tau_2\left(2\frac{\sqrt{3}}{3}\sin\phi-\tfrac{1}{2}\left(\frac{\sqrt{3}}{3}\sin\phi-\cos\phi\right)\right)\\ &= \tau_1\left(\tfrac{3}{2}\sin\phi-\tfrac{\sqrt{3}}{6}\cos\phi\right),\end{aligned}$$

whose maximum is $\frac{3}{2}\tau_1 \simeq 1.5800$, obtained for $\phi = \frac{\pi}{2}$. Averaging on ϕ yields an expected value of

$$\frac{6}{\pi}\int_{\frac{\pi}{3}}^{\frac{\pi}{2}} \tau_1\left(\tfrac{3}{2}\sin\phi-\tfrac{\sqrt{3}}{6}\cos\phi\right)\mathrm{d}\phi = \tfrac{6-\sqrt{3}}{\pi}\tau_1 \simeq 1.4306.$$

Figure 9 depicts the expected routing factors as functions of ϕ. □

Acknowledgements

The authors would like to thank Nicolas Chenavier for interesting discussions related to this paper.

References

[1] Oswin Aichholzer, Sang Won Bae, Luis Barba, Prosenjit Bose, Matias Korman, André van Renssen, Perouz Taslakian, and Sander Verdonschot. Theta-3 is connected. *Computational Geometry: Theory and Applications*, 47(9):910–917, 2014. doi:10.1016/j.comgeo.2014.05.001.

[2] Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and David Ilcinkas. Connections between Theta-graphs, Delaunay triangulations, and orthogonal surfaces. In *Proceedings of the 36th International Conference on Graph Theoretic Concepts in Computer Science (WG 2010)*, pages 266–278, 2010. URL: https://hal.archives-ouvertes.fr/hal-00454565.

[3] Nicolas Bonichon and Jean-François Marckert. Asymptotics of geometrical navigation on a random set of points in the plane. *Advances in Applied Probability*, 43(4):899–942, 2011. doi:10.1239/aap/1324045692.

[4] Prosenjit Bose, Jean-Lou De Carufel, Darryl Hill, and Michiel H. M. Smid. On the spanning and routing ratio of Theta-four. In *Symposium on Discrete Algorithms*, pages 2361–2370. SIAM, 2019. doi:10.1137/1.9781611975482.144.

[5] Prosenjit Bose, Jean-Lou De Carufel, Pat Morin, André van Renssen, and Sander Verdonschot. Towards tight bounds on theta-graphs: More is not always better. *Theoretical Computer Science*, 616:70–93, 2016. doi:10.1016/j.tcs.2015.12.017.

[6] Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Optimal local routing on Delaunay triangulations defined by empty equilateral triangles. *SIAM Journal on Computing*, 44(6):1626–1649, 2015. doi:10.1137/140988103.

[7] Prosenjit Bose and Pat Morin. Online routing in triangulations. *SIAM Journal on Computing*, 33(4):937–951, 2004. doi:10.1137/S0097539700369387.

[8] Prosenjit Bose, Pat Morin, André van Renssen, and Sander Verdonschot. The Θ_5-graph is a spanner. *Computational Geometry: Theory and Applications*, 48(2):108 – 119, 2015. doi:10.1016/j.comgeo.2014.08.005.

[9] Prosenjit Bose and Michiel Smid. On plane geometric spanners: A survey and open problems. *Computational Geometry: Theory and Applications*, 46(7):818–830, 2013. doi:10.1016/j.comgeo.2013.04.002.

[10] Paul Chew. There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2):205–219, 1989.

[11] Edsger Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[12] John Hopcroft and Robert Tarjan. Algorithm 447: Efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973. doi:10.1145/362248.362272.

[13] Victor Korolev and Irina Shevtsova. An improvement of the berry-esseen inequality with applications to poisson and mixed poisson random sums. *Scandinavian Actuarial Journal*, pages 1–25, 2010. URL: https://arxiv.org/abs/0912.2795, doi:10.1080/03461238.2010.485370.

[14] C. Y. Lee. An algorithm for path connection and its applications. *IRE Transaction on Electronic Computers*, EC-10(3):346–365, 1961. doi:10.1109/TEC.1961.5219222.

[15] Edward Moore. The shortest path through a maze. In *Proceedings of the International Symposium on the Theory of Switching*, pages 285–292, 1959.

[16] Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, 2007.

[17] Jim Ruppert and Raimund Seidel. Approximating the d-dimensional complete Euclidean graph. In *Proceedings of the 3rd Canadian Conference on Computational Geometry (CCCG 1991)*, pages 207–210, 1991.

[18] Rolf Schneider and Wolfgang Weil. *Stochastic and Integral Geometry*. Probability and Its Applications. Springer Berlin Heidelberg, 2008.