On the Spanning and Routing Ratio of Directed Theta-Four[☿][♁]

Prosenjit Bose[♂] *Jean-Lou De Carufel*[♃] *Darryl Hill*[♂] *Michiel Smid*[♂]

Abstract. We present a routing algorithm for the directed Θ_4-graph, here denoted as the $\overrightarrow{\Theta_4}$-graph, that computes a path between any two vertices s and t having length at most 17 times the Euclidean distance between s and t. To compute this path, at each step, the algorithm only uses knowledge of the location of the current vertex, its (at most four) outgoing edges, the destination vertex, and one additional bit of information in order to determine the next edge to follow. This provides the first known online, local, competitive routing algorithm with constant routing ratio for the Θ_4-graph, as well as improving the best known upper bound on the spanning ratio of these graphs from 237 to 17. We also show that without this additional bit of information, the routing ratio increases to $\sqrt{290} \approx 17.03$.

1 Introduction

Finding a path in a graph is a fundamental problem in computer science. Typically, algorithms that compute paths in graphs have at their disposal knowledge of the whole graph. The problem of finding a path in a graph is more difficult in the online setting, when the routing algorithm must explore the graph as it attempts to find a path. Moreover, the situation is even more challenging if the routing algorithm only has a constant amount of working memory, i.e. it can only remember a constant size subgraph of the portion of the graph it has explored. Specifically, an online routing algorithm attempting to find a path from one vertex to another is called *local* if at each step, the only information it can use to make its forwarding decision is the location of the current vertex and its neighbouring vertices, plus a constant amount of additional information.

For a routing algorithm A and a given graph G from a class $\mathcal{G}$ of graphs, let $\mathcal{P}_G^A(s,t)$ be the path in G found by A from s to t. The class of graphs we focus on are a subclass of weighted geometric graphs. A weighted geometric graph $G = (P, E)$ is a graph whose vertex set is a set P of points in the plane and a set E of (directed or undirected) edges between pairs of points, where the weight of an edge (p,q) is equal to the Euclidean distance $L_2(p,q)$ between its endpoints (i.e., distance in the L_2-metric). For a pair of vertices s and t in P, let $\mathcal{P}_G(s,t)$ be the shortest path from s to t in G, and let $L_2(\mathcal{P}_G(s,t))$ be the length of $\mathcal{P}_G(s,t)$ with respect to the L_2-metric, i.e., the sum of the lengths of the edges of $\mathcal{P}_G(s,t)$. The spanning ratio of a graph G is the minimum value c such that $L_2(\mathcal{P}_G(s,t)) \leq c \cdot L_2(s,t)$ over all pairs of points s and t in G. A graph is called a c-*spanner*, or just a *spanner*, if its spanning ratio is at most some constant c. The routing ratio of a local online routing algorithm A on $\mathcal{G}$ is the maximum value c' such that $L_2(\mathcal{P}_G^A(s,t)) \leq c' \cdot L_2(s,t)$ for all $G \in \mathcal{G}$ and all pairs s and t in G. When c' is a constant, such an algorithm is called *competitive* on the class $\mathcal{G}$. Note that the routing ratio on a class of graphs $\mathcal{G}$ is an upper bound on the spanning ratio of $\mathcal{G}$, since the routing ratio proves the existence of a bounded-length path.

[☿]A preliminary version of this paper appeared in the proceedings of SODA 2019
[♁]This work was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).
[♂]Carleton University
[♃]University of Ottawa

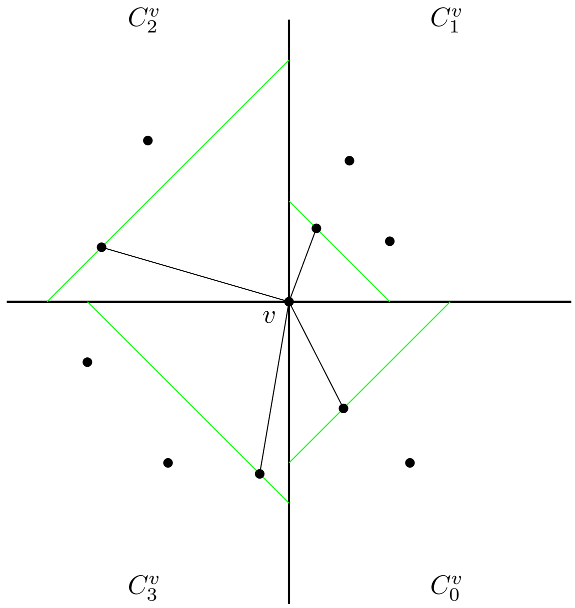


Figure 1: Neighbours of v in the Θ_4-graph.

1.1 Θ-graphs

Let $k \geq 3$ be an integer and for each i with $0 \leq i < k$, let $\mathcal{R}_i$ be the ray emanating from the origin that makes an angle of $2\pi i/k$ in the counter-clockwise direction measured from the negative y-axis. Let $\mathcal{R}_k = \mathcal{R}_0$. The Θ_k-graph of a given set P of points is the directed graph that is obtained in the following way. The vertex set is the set P. Each vertex v has at most k outgoing edges: For each i with $0 \leq i < k$, let $\mathcal{R}_i^v$ be the ray emanating from v parallel to $\mathcal{R}_i$. Let C_i^v be the cone consisting of all points in the plane that are strictly between the rays $\mathcal{R}_i^v$ and $\mathcal{R}_{i+1}^v$ or on $\mathcal{R}_{i+1}^v$. If C_i^v contains at least one point of $P \setminus \{v\}$, then let w_i be such a point whose perpendicular projection onto the bisector of C_i^v is closest to v (where closest refers to the Euclidean distance). Then the Θ_k-graph contains the directed edge (v, w_i). See Figure 1 for an example with $k = 4$. While most of the literature discussed focuses on undirected Θ_k-graphs, and thus (v, w_i) becomes the undirected edge $\{v, w_i\}$, in this paper we will study routing in the directed setting. We will hereafter refer to directed Θ_k-graphs using the $\overrightarrow{\Theta_k}$ notation.

Θ_k-graphs were introduced independently by Keil and Gutwin [21, 22], and Clarkson [20]. Both papers gave a spanning ratio of $1/(\cos\theta - \sin\theta)$, where $\theta = 2\pi/k$ is the angle defined by the cones. Observe this gives a constant spanning ratio for $k \geq 9$. Ruppert and Seidel [26] improved this to $1/(1 - 2\sin(\theta/2))$, which applies to Θ_k-graphs with $k \geq 7$. Bose et al. [9] give a tight bound of 2 for $k = 6$. In the same paper are the current best bounds on the spanning ratio of a large range of values of k. More recently Bose et al. [2] showed that $\overrightarrow{\Theta_6}$ has a spanning ratio of 7. For $k = 5$, Bose et al. [12] showed an upper bound on the spanning ratio of ≈ 5.70. The previous bound by Bose et al. [15] of ≈ 9.96 also showed a lower bound of ≈ 3.78. For $k = 4$, Barba et al. [3] showed a spanning ratio of ≈ 237, with a lower bound of 7. For $k = 3$, Aichholzer et al. [1] showed that Θ_3 is connected, but Molla [24] showed that there is no constant c for which Θ_3 is a c-spanner.

1.2 Local Routing

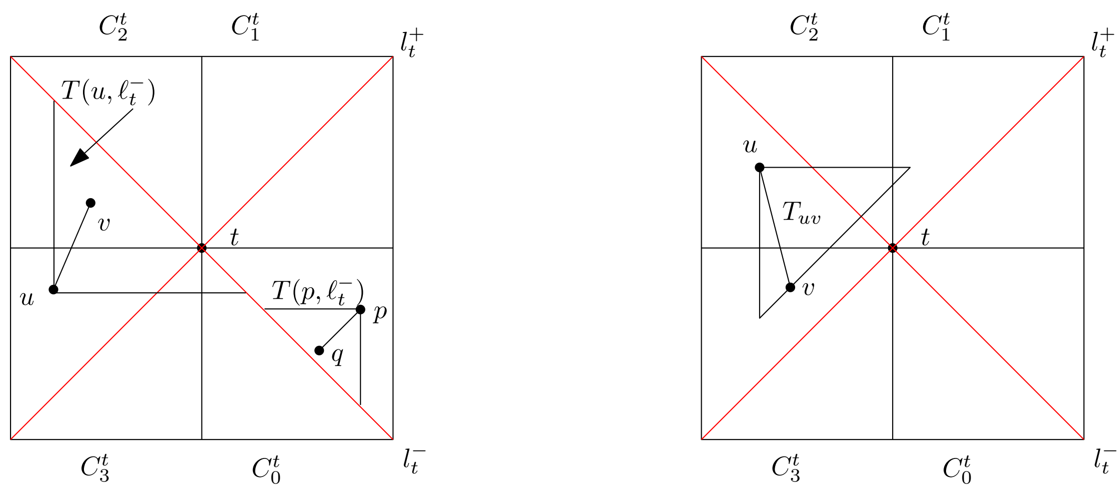
Local routing has been studied extensively in variants of the Delaunay graph as well as Θ_k-graphs (see [14, 7, 23, 18, 19, 11, 25, 17, 16]). Also, more recently there has been interest in routing on $\overrightarrow{\Theta_k}$-graphs [2]. There is an intimate connection between Θ_k-graphs and variants of the Delaunay triangulation. For example, the existence of an edge in a Θ_k-graph implies the existence of an empty triangle containing the edge (refer to Figure 1). In a Delaunay triangulation, the existence of an edge implies the existence of an empty disk containing the edge (or some empty convex shape when considering variants of the Delaunay graph). Moreover, the Delaunay graph where the empty convex shape is an equilateral triangle (this is often referred to as the TD-Delaunay graph [6]) is a subgraph of the Θ_6-graph.

Chew [18] proved that the L_1-Delaunay graph has bounded spanning ratio by providing a local routing algorithm whose routing ratio is at most $\sqrt{10}$. Bose and Morin[13] provided a competitive local routing algorithm that works on triangulations that have the *diamond property*. This includes such graphs as the L_2-Delaunay triangulation, the greedy triangulation, and the minimum weight triangulation. Bose and Morin[14] showed that there are no deterministic routing algorithms that work on any arbitrary graph. This implies that we must pair routing algorithms with particular classes of geometric graphs in order to route competitively. They also provided the first deterministic competitive routing algorithm on the L_2-Delaunay graphs. Bonichon et al. [5] showed that we could route competitively on the L_2-Delaunay triangulation with a routing ratio of around 5.9, using a generalization of Chew's [18] algorithm. This was the best known routing ratio for L_2-Delaunay triangulations until Bonichon et al. [4] gave a new algorithm with a routing ratio of 3.56, which is currently the best known. Bose et al. [10] show that the half-θ_6 graph, which is identical to the TD-Delaunay graph, has a routing ratio of $5/\sqrt{3}$, and this is shown to be tight. Since the spanning ratio of this graph is 2, it is an example where a local routing algorithm cannot necessarily find the shortest path, and we see a separation between the routing and spanning ratios in this graph.

For Θ_k-graphs, there is a simple routing algorithm called *cone-routing* or *greedy-routing* that is competitive for $k \geq 7$. To route from a vertex s to a vertex t, let C_i^s be the cone of s that contains t. Forward the packet from s to its neighbour in C_i^s, and repeat this until the destination is reached. Let $\theta = 2\pi/k$, then for $k \geq 7$, Ruppert and Seidel [26] proved that cone routing gives a routing ratio of $1/(1-2\sin(\theta/2))$. Cone routing also has the advantage of only utilizing outgoing edges, so each vertex only needs to store the location of at most k neighbours. That means these algorithms and results also apply to the $\overrightarrow{\Theta_k}$-graphs. For $k < 7$, cone-routing does not necessarily give a short path. In fact, Bose, De Carufel and Devillers [8] showed that cone-routing has unbounded routing ratios for $k \leq 6$. However, for $k = 6$, Bose et al. [10] show that a different local online routing algorithm gives a routing ratio of $\sqrt{5}/2 \approx 2.89$. More recently, Bose et al. [2] give a local online routing algorithm for the $\overrightarrow{\Theta_6}$-graph with a routing ratio of at most 14. Prior to this work, there was no known competitive routing algorithm for $k = 4$.

1.3 Our Results

In this paper we improve the upper bound of the spanning ratio of $\overrightarrow{\Theta_4}$-graphs (and, by extension, Θ_4-graphs) from 237 to 17. We do this by providing a local online routing algorithm with a routing ratio of at most 17. This is the first local routing algorithm for $\overrightarrow{\Theta_k}$-graphs or Θ_k-graphs for $k = 4$, bringing us one step closer to obtaining competitive routing strategies on all $\overrightarrow{\Theta_k}$- and



(a) Two examples of sweeping steps towards ℓ_t^-.

(b) A greedy step towards t.

Figure 2

Θ_k-graphs with $k > 3$. Our algorithm is slightly counter-intuitive since it sometimes takes a step in a cone that does not contain the destination. This is different from cone-routing that always takes a step in the cone that contains the destination. The algorithm is simple, and only uses knowledge of the destination vertex, the current vertex v, the neighbours of v, and one bit of additional information. If we forgo that bit of information, then the routing ratio increases to $\sqrt{290} \approx 17.03$. Additionally, like cone-routing, we route using only outgoing edges, so each vertex only needs to store the location of its at most 4 outgoing neighbours. For the remainder of the paper, all edges (u, v) are considered directed outgoing edges from u to v, and when we refer to the neighbour v of a vertex u in a cone C_i^u, we are referring to the outgoing edge (u, v) of u.

The rest of the paper is organized as follows. Section 2 gives the details of the routing algorithm that is used to navigate the $\overrightarrow{\Theta_4}$-graph. In Section 3 we analyze the length of the path found by the algorithm, and show an upper bound of 17 on the routing ratio. In Section 4 we give an example of a path that shows this approach cannot do any better than a routing ratio of 17. In Section 5 we show how routing with only knowledge of the destination vertex increases the routing ratio to $\sqrt{290} \approx 17.03$. Section 6 concludes the paper and gives some directions for future work.

2 Algorithm

In this section, we present our 17-competitive local online routing algorithm on $\overrightarrow{\Theta_4}$-graphs. We first introduce some concepts and notation related to the $\overrightarrow{\Theta_4}$-graph. We then define the routing model, and finally we describe the routing algorithm in detail.

2.1 Preliminaries

Let t be an arbitrary point in the plane, and let ℓ_t^- be the line through t with slope -1. Similarly let ℓ_t^+ be the line through t with slope 1. We refer to these as the *diagonals* of t. Examples can be seen in Figures 2a, 2b, and 3. To ease our analysis and avoid tedious tie-breaking, we make a general position assumption that no two vertices have the same x- or y-coordinates, and no two

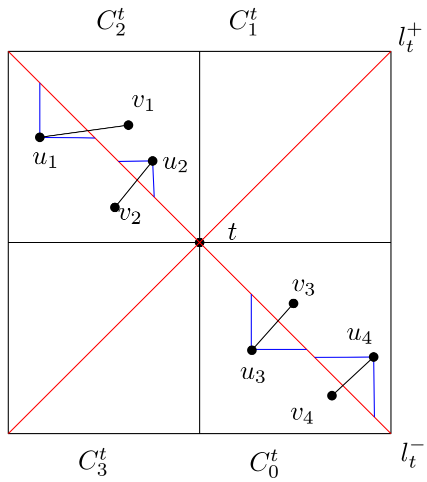


Figure 3: Vertices u_1, u_2, u_3, and u_4 are all clean with respect to ℓ_t^-.

vertices lie on a common diagonal. Let t and u be arbitrary vertices and consider a diagonal of t. Without loss of generality, we consider the diagonal ℓ_t^- and assume that u is in the half-plane below ℓ_t^-. Let $\mathcal{R}_i^u$ and $\mathcal{R}_{i+1}^u$ be the rays emanating from u that intersect ℓ_t^-. Recall that $\mathcal{R}_i^u$ and $\mathcal{R}_{i+1}^u$ delineate the cone C_i^u. Let the triangle $T(u, \ell_t^-)$ be the intersection of the halfplane of ℓ_t^- containing u and the cone C_i^u. We say that C_i^u *faces* ℓ_t^-. If $T(u, \ell_t^-)$ is empty of vertices (not including t), then we say that u is *clean with respect to* ℓ_t^-. See Figure 3. If the diagonal we are referring to is clear from the context, we simply say that u is *clean*. If u is not clean with respect to ℓ_t^-, then let v be the vertex in $T(u, \ell_t^-)$ for which (u, v) is an edge in the $\overrightarrow{\Theta_4}$-graph. We will refer to following the edge from u to v as *taking a sweeping step towards* ℓ_t^-. (See Figure 2a.) Let i be the index such that the vertex t is in the cone C_i^u. Let v be the vertex in C_i^u for which (u, v) is an edge in the $\overrightarrow{\Theta_4}$-graph. We will refer to following the edge from u to v as *taking a greedy step towards* t. (See Figure 2b.) Note that when routing with respect to ℓ_t^- (respectively ℓ_t^+) and the current vertex v is in C_3^t or C_1^t (respectively C_0^t or C_2^t), a greedy step towards t and a sweeping step towards ℓ_t^- (respectively ℓ_t^+) are the same. However, by our definition of *clean* and to disambiguate the analysis, this step is defined as a sweeping step.

2.2 Routing Model

An online local routing algorithm takes as input $u, t, N(u), m$ where u is the current vertex, t is the target vertex, $N(u)$ are the (1-hop) neighbours of u in G, and $m = \{0, 1\}^*$ is a bitstring of memory. The algorithm returns a vertex $v \in N(u)$ on the path from u to t and updates m if necessary. The maximum length of m over all steps of the algorithm represents the memory requirements of the algorithm. If the maximum length of m is 0 we say it is a *memoryless* algorithm. That is, the algorithm does not require any knowledge of the previous vertices, including the start vertex. The strongest version of this routing model uses unlimited memory, while the weakest version of this routing model is memoryless.

The local *Greedy/Sweep* routing algorithm that we define here has two versions, one that is a memoryless with a routing ratio of $\sqrt{290} \approx 17.03$ and one that uses 1 bit of memory and has a routing ratio of 17. We describe the 1 bit routing in this section. We show how to get memoryless routing with a small modification to our algorithm in Section 5.

2.3 The *Greedy/Sweep* Algorithm

We now define the 1-bit version of the *Greedy/Sweep* algorithm. Let s be the source vertex and t the target vertex. The algorithm first chooses a diagonal of t as follows: If $s \in C_0^t \cup C_2^t$, the algorithm chooses ℓ_t^-, otherwise it chooses ℓ_t^+. Intuitively the algorithm chooses the diagonal of t "closer" to s, and d encodes this information about the location of s. Our routing algorithm is then denoted by $Greedy/Sweep(u,t,N(u),d)$ where u is the current vertex, t is the target vertex, $N(u)$ are the (1-hop) neighbours of u and $d \in \{0,1\}$ is the memory representing the chosen diagonal of t. That is, $d=0$ corresponds to ℓ_t^- and $d=1$ corresponds to ℓ_t^+. For the purposes of simplification, we will express d as the diagonal directly, that is, we will assume that $d \in \{\ell_t^-, \ell_t^+\}$.

The $Greedy/Sweep(u,t,N(u),d)$ algorithm uses three other "helper" algorithms to determine the output vertex v, which we will define here. Let $v \in N(u)$ be the vertex in the cone of u that faces d. Note this implies that (u,v) is an edge in $\overrightarrow{\Theta_4}$. Then $Clean(u,t,N(u),d)$ returns True if (u,v) crosses d and False otherwise. $Sweep(u,t,N(u),d)$ returns the vertex v such that (u,v) is the edge in the cone of u facing ℓ_t^-. $Greedy(u,t,N(u))$ returns the vertex v such that (u,v) is the edge in the cone of u that contains t. Then the algorithm $Greedy/Sweep(u,t,N(u),d)$ is described in Algorithm 1.

Algorithm 1: $Greedy/Sweep(u,t,N(u),d)$

Input: u is the current vertex;
t is the target vertex;
$N(u)$ are the neighbours of u;
$d \in \{\ell_t^-, \ell_t^+\}$ is the chosen diagonal of t;

if $u = t$ **then** END;

if $Clean(u,t,N(u),d)$ **then**
  return $Greedy(u,t,N(u))$;
else
  return $Sweep(u,t,N(u),d)$;

Figure 4 gives an example of a path from s to t computed by Algorithm 1.

3 Analysis

In this section, we prove that our routing algorithm terminates and that it has a routing ratio of 17. Without loss of generality, we assume that s is in C_2^t under ℓ_t^-. Thus ℓ_t^- is the closest diagonal of t to s. For two arbitrary points u and v, let $d_x(u,v)$ and $d_y(u,v)$ be the distance between them along the x-axis and y-axis respectively. Let $L_1(u,v)$ be the L_1 distance between u and v (i.e., $L_1(u,v) = d_x(u,v) + d_y(u,v)$), and let $L_\infty(u,v)$ be the L_∞ distance from u to v (i.e., $L_\infty(u,v) = \max\{d_x(u,v), d_y(u,v)\}$). To simplify our analysis, most of our intermediate measurements will be in the L_1-metric. In the final analysis we will express the length in the L_2-metric. Let $\mathcal{P}(s,t)$ be the sequence of directed edges produced by our algorithm. For vertices u and v in $\mathcal{P}(s,t)$, with u occurring before v, let $\mathcal{P}\langle u,v\rangle$ be the subpath of $\mathcal{P}(s,t)$ from u to v.

We divide the area around t into *quadrants*. The *Northern* quadrant is the area above ℓ_t^-

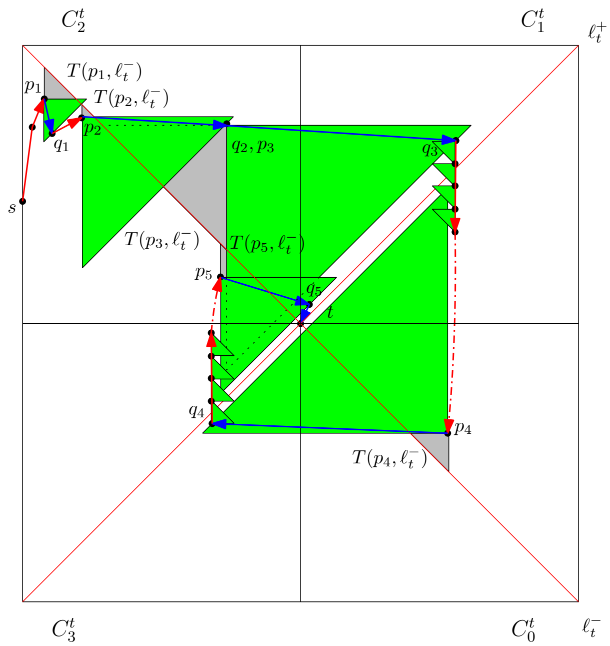


Figure 4: An example of the algorithm showing canonical triangles and cleaned triangles. Blue lines are greedy steps and red lines are sweeping steps.

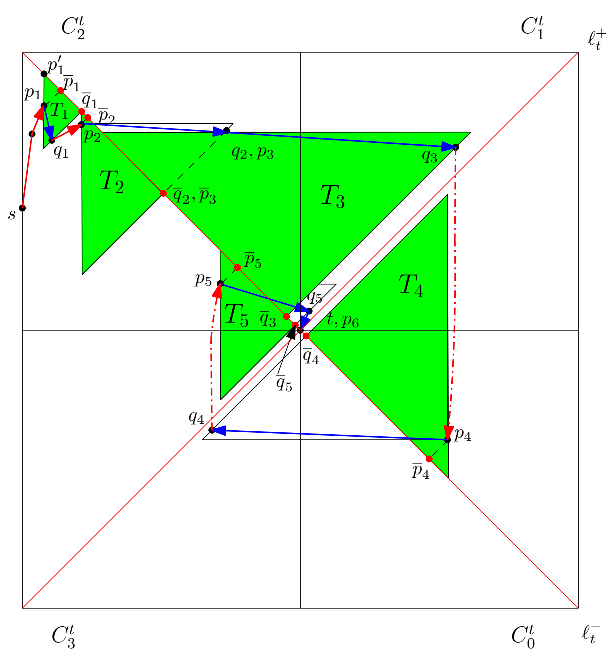


Figure 5: The bounding triangles. Blue lines are greedy steps towards t, red lines are sweeping steps towards ℓ_t^-.

and ℓ_t^+, while the *Southern* quadrant is the area below ℓ_t^- and ℓ_t^+. The *Western* quadrant is the area to the left of ℓ_t^- and ℓ_t^+, while the *Eastern* quadrant is the area to the right of ℓ_t^- and ℓ_t^+.

We note the following about the path found by the algorithm:

Lemma 1. *Let u and v be two consecutive vertices on $\mathcal{P}(s,t)$. Then $L_\infty(u,t) > L_\infty(v,t)$.*

Proof. Note that we can without loss of generality assume that u is a vertex in the Western quadrant and that we are routing with respect to ℓ_t^- by rotating the point set around t or by flipping the point set along ℓ_t^-. Since u is below ℓ_t^- and above ℓ_t^+, $L_\infty(u,t) = d_x(u,t)$. If v is in the Western quadrant, observe that for both a greedy step and a sweeping step, $d_x(u,t) > d_x(v,t)$ since we assume no two vertices have the same x-coordinate. This implies that $L_\infty(u,t) > L_\infty(v,t)$, as required.

Assume v is not in the Western quadrant. For v to be in the Eastern quadrant, (u,v) must cross both diagonals of t. Observe that a greedy step does not cross ℓ_t^+, while a sweeping step does not cross ℓ_t^-. Thus v can only be in the Northern or Southern quadrant, and $L_\infty(v,t) = d_y(v,t)$. Observe that if v is in the Southern quadrant, then (u,v) was a sweeping step, and if v is in the Northern quadrant, then (u,v) was a greedy step. In both cases, $d_y(u,t) > d_y(v,t)$ since we assume no two vertices have the same y-coordinate. Thus $L_\infty(u,t) = d_x(u,t) > d_y(u,t) > d_y(v,t) = L_\infty(v,t)$, as required. See Figs. 2a and 2b. □

Since there are a finite number of vertices, this leads to the following corollary.

Corollary 1. *The $Greedy/Sweep$ algorithm terminates, i.e., it reaches t.*

Let $((p_1,q_1),(p_2,q_2),\ldots,(p_{m-1},q_{m-1}))$ be the sequence of edges produced by greedy steps of the algorithm, with $t = p_m$. A *phase* f_i of the algorithm refers to the path from a vertex p_i to a vertex p_{i+1} consisting of $(p_i,q_i) + \mathcal{P}\langle q_i,p_{i+1}\rangle$, for $1 \le i < m$. That is, a phase consists of a single greedy step followed by a (possibly empty) sequence of sweeping steps. Note that the first phase is preceded by a (possibly empty) sequence of sweeping steps from s to p_1. Let $L_2(f_i) = L_2(p_i,q_i) + L_2(\mathcal{P}\langle q_i,p_{i+1}\rangle)$ represent the length of phase f_i. Then observe that $\mathcal{P}(s,t) = \mathcal{P}\langle s,p_1\rangle + \sum_{i=1}^{m-1} f_i$, where the + operator on paths is concatenation of the paths. Note that if each vertex on $\mathcal{P}\langle u,v\rangle$ is in the same cone $i, 0 \le i \le 3$ of all preceding vertices, then $\mathcal{P}\langle u,v\rangle$ is x- and y-monotone, and $L_2(\mathcal{P}\langle u,v\rangle) \le L_1(u,v)$. This implies that $L_2(\mathcal{P}\langle q_i,p_{i+1}\rangle) \le L_1(q_i,p_{i+1})$, for all $1 \le i < m-1$.

Let v be the neighbour of an arbitrary vertex u in the cone C_i^u. Let the *canonical triangle* T_{uv} be the triangle formed by the boundaries of C_i^u and the line through v perpendicular to the bisector of C_i^u. Note that the existence of (u,v) guarantees that T_{uv} is empty of vertices in its interior. See Figure 2b.

Definition 3.1. *Consider the edge (p_i,q_i) of f_i. If p_i is in the Northern or Southern quadrant, then let $\mathcal{L}$ be the horizontal line through p_i, otherwise $\mathcal{L}$ is the vertical line through p_i. Let the* bounding triangle T_i *be the triangle formed by the lines ℓ_t^-, $\mathcal{L}$, and $\ell_{q_i}^+$. See Figs. 6, 7, and 8.*

Lemma 2. *The bounding triangle T_i of the greedy edge (p_i,q_i) of phase f_i is empty of vertices.*

Proof. Since (p_i,q_i) is a greedy step, p_i is clean with respect to ℓ_t^-, and $T(p_i,\ell_t^-)$ and $T_{p_iq_i}$ are both empty of vertices. Observe that T_i lies completely in one of the half-planes of ℓ_t^-. If $T_{p_iq_i}$ does not intersect ℓ_t^-, then $T_{p_iq_i} \subseteq T_i$ and $T(p_i,\ell_t^-) \nsubseteq T_i$. See Figure 6. If $T_{p_iq_i}$ does intersect ℓ_t^-,

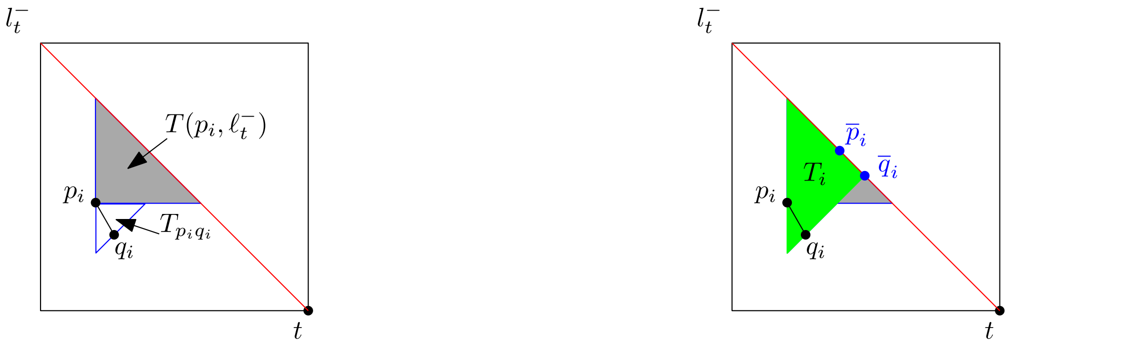


Figure 6: $T_{p_i q_i}$ does not intersect ℓ_t^-.

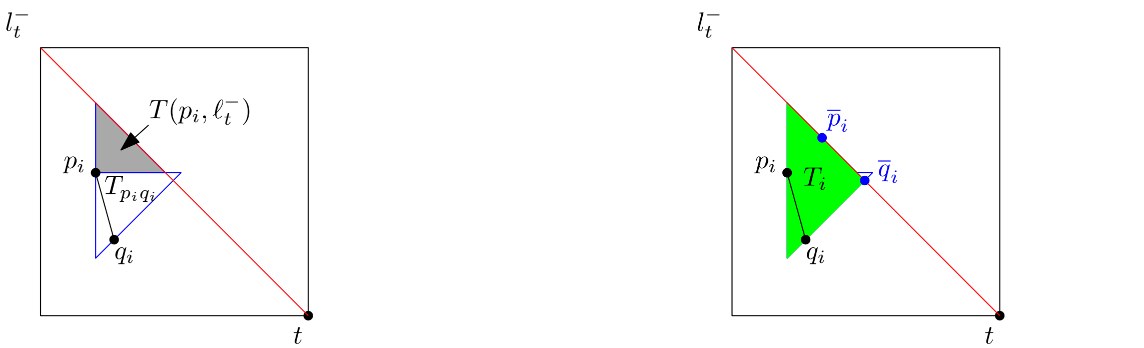


Figure 7: $T_{p_i q_i}$ intersects ℓ_t^-, and q_i lies on T_i.

then observe that $T(p_i, \ell_t^-) \subseteq T_i$ and $T_{p_i q_i} \nsubseteq T_i$. In this case, q_i can be on the same side of ℓ_t^- as p_i, and thus lie on T_i (Figure 7), or it can be on the opposite side of ℓ_t^-, and not lie on T_i (Figure 8). In all cases observe that $T_i \subseteq T_{p_i q_i} \cup T(p_i, \ell_t^-)$, and thus T_i is empty of vertices. $\square$

Notice that a bounding triangle T_i cannot be on both sides of ℓ_t^- by construction, and cannot be on both sides of ℓ_t^+ since that would imply that t is within $T_{p_i q_i}$. This implies that a bounding triangle T_i can only intersect the interior of a single quadrant.

Lemma 1 has strong implications about the positions of bounding triangles relative to one another in the same quadrant. For a vertex p, let $\bar{p}$ be the intersection of ℓ_t^- and ℓ_p^+, i.e, $\bar{p}$ is the intersection of the positive diagonal of p and the negative diagonal of t.

Lemma 3. *If T_i and T_j are two bounding triangles in the same quadrant, then $\bar{p}_j \bar{q}_j$ and $\bar{p}_i \bar{q}_i$ are disjoint segments on ℓ_t^-.*

Proof. Without loss of generality, assume that $i < j$. Note that for a point $\bar{v}$ lying on ℓ_t^-, $L_1(\bar{v}, t) = \sqrt{2} \cdot L_2(\bar{v}, t)$. That is, the L_1- and L_2-distances are proportional. Then Lemma 3 is true if $L_1(\bar{p}_i, t) > L_1(\bar{q}_i, t) > L_1(\bar{p}_j, t) > L_1(\bar{q}_j, t)$ is true. Assume without loss of generality T_i and T_j are in the Western quadrant. See Figure 9. Note that $L_1(\bar{p}_i, t) - L_1(\bar{q}_i, t) = L_1(\bar{p}_i, \bar{q}_i) = L_1(p_i, q_i)$, since q_i and t are in the same cone of p_i. Thus $L_1(\bar{p}_i, t) > L_1(\bar{q}_i, t)$ and $L_1(\bar{p}_j, t) > L_1(\bar{q}_j, t)$ are true. What remains to be shown is that $L_1(\bar{q}_i, t) > L_1(\bar{p}_j, t)$. Lemma 1 implies $L_\infty(p_i, t) > L_\infty(p_j, t)$, and both points are in the Western quadrant (by the definition of bounding triangle), thus p_j cannot be

Figure 8: $T_{p_iq_i}$ intersects ℓ_t^-, and q_i does not lie on T_i.

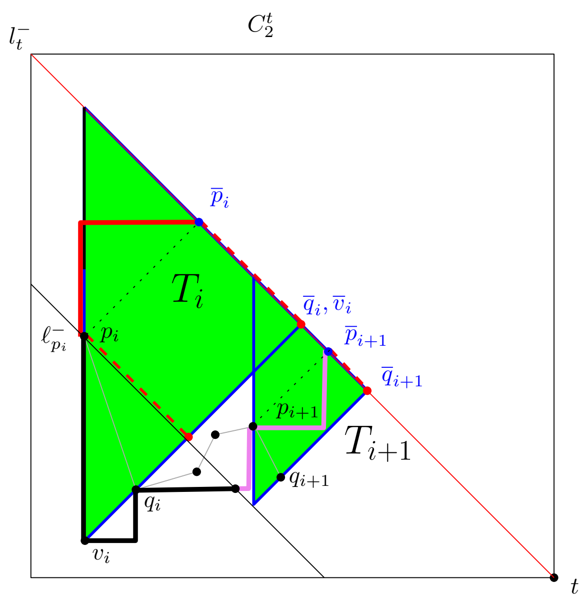


Figure 9: The red path from p_i to $\bar{p}_i$ is the same length as the violet path. Thus $\Phi(p_i, p_{i+1})$ is equal to the length of the black path.

left of p_i. This, and the fact that T_i is empty, implies p_j must be below $\ell^+_{q_i}$, which implies $\ell^+_{p_j}$ is below $\ell^+_{q_i}$, which implies $L_1(\bar{q}_i, t) > L_1(\bar{p}_j, t)$. $\square$

Figure 9 shows two consecutive bounding triangles in the Western quadrant, and the associated segments $\bar{p}_i\bar{q}_i$ and $\bar{p}_{i+1}\bar{q}_{i+1}$.

Let p_1' be the vertical projection of p_1 onto ℓ_t^-. Then the following inequality is true.

Corollary 2. $\sum_{i=1}^{m-1} L_1(\bar{p}_i, \bar{q}_i) \leq 4 \cdot L_1(p_1', t)$.

Proof. Lemma 1 implies that $L_1(p_1', t) > L_1(\bar{p}_i, t)$ for all $1 \leq i \leq m-1$. This combined with Lemma 3 and the fact that there are four quadrants implies the lemma. $\square$

Lemma 4. $L_2(f_i) \leq L_1(p_i, q_i) + L_1(q_i, p_{i+1})$.

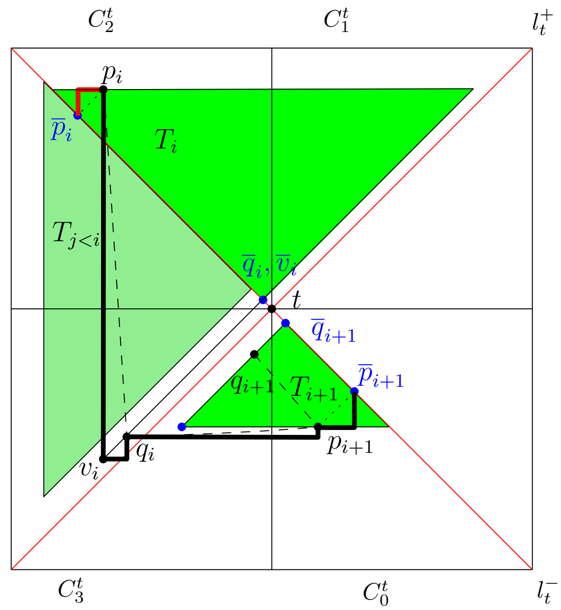


Figure 10: $L_1(q_i, \bar{p}_{i+1}) < L_1(p_i, q_i)$, since p_i is in $C_2^{q_i}$ and on the opposite side of ℓ_t^- as q_i.

Proof. This follows from the fact that (p_i, q_i) is an edge, and $\mathcal{P}\langle q_i, p_{i+1}\rangle$ is x- and y-monotone. □

Each of the bounding triangles T_i are associated with the segment $\bar{p}_i\bar{q}_i$ and the phase f_i. A natural approach is to try to bound the length of the phase $L_2(f_i)$ by the length of the segment $\bar{p}_i\bar{q}_i$. Unfortunately this approach does not quite work since the length of $\bar{p}_i, \bar{q}_i$ is proportional to the length of (p_i, q_i), but the length of the sequence of sweeping steps of f_i can be unbounded with respect to the length of (p_i, q_i). However, a relatively simple potential function reassigns the lengths of sweeping steps to either the previous phase or the next phase so that these lengths are proportional to the length of (p_i, q_i). We define the potential function $\Phi(p_i, p_{i+1}) = L_1(p_i, q_i) + L_1(q_i, \bar{p}_{i+1}) - L_1(p_i, \bar{p}_i)$ for all $1 \le i < m-1$.

Lemma 5. $\Phi(p_i, p_{i+1}) \le 2 \cdot L_1(\bar{p}_i, \bar{q}_i)$.

Proof. Without loss of generality, assume that q_i is in the Western quadrant. Since greedy steps can only originate in C_2^t and C_0^t, and a greedy step cannot cross ℓ_t^+, p_i is in C_2^t in either the Western or Northern quadrant (Figs. 9 and 10 respectively). Let v_i be the bottommost point of $T_{p_i q_i}$. Since v_i is on $\ell_{q_i}^+$, we have $\bar{v}_i = \bar{q}_i$. Since both p_i and $\bar{p}_i$ are on $\ell_{p_i}^+$, and both v_i and $\bar{v}_i$ are on $\ell_{v_i}^+$, we have $L_1(p_i, v_i) = L_1(\bar{p}_i, \bar{v}_i) = L_1(\bar{p}_i, \bar{q}_i)$. Thus it is enough to prove that $\Phi(p_i, p_{i+1}) \le 2 \cdot L_1(p_i, v_i)$. Observe that $L_1(p_i, v_i) + L_1(v_i, q_i) \ge L_1(p_i, q_i)$ by the triangle inequality.

If p_i is in the Western quadrant, we have that $L_1(v_i, \bar{p}_i) = L_1(v_i, p_i) + L_1(p_i, \bar{p}_i)$. Since both $\bar{p}_i$ and $\bar{p}_{i+1}$ lie in $C_1^{v_i}$ on ℓ_t^-, we have also have that $L_1(v_i, \bar{p}_i) = L_1(v_i, \bar{p}_{i+1})$. Thus

$$\begin{aligned}\Phi(p_i, p_{i+1}) &= L_1(p_i, q_i) + L_1(q_i, \bar{p}_{i+1}) - L_1(p_i, \bar{p}_i) \\ &\le L_1(p_i, v_i) + L_1(v_i, \bar{p}_{i+1}) - L_1(p_i, \bar{p}_i) \\ &= L_1(p_i, v_i) + L_1(v_i, \bar{p}_i) - L_1(p_i, \bar{p}_i) \\ &= 2 \cdot L_1(p_i, v_i)\end{aligned}$$

as required. Otherwise p_i is in the Northern quadrant. Observe that p_i and $\bar{p}_{i+1}$ are both in $C_1^{v_i}$, but p_i is above ℓ_t^- while $\bar{p}_{i+1}$ is on it, thus $L_1(p_i, v_i) > L_1(v_i, \bar{p}_{i+1})$. Thus

$$\begin{aligned}\Phi(p_i, p_{i+1}) &= L_1(p_i, q_i) + L_1(q_i, \bar{p}_{i+1}) - L_1(p_i, \bar{p}_i) \\ &\le L_1(p_i, v_i) + L_1(v_i, \bar{p}_{i+1}) \\ &\le 2 \cdot L_1(p_i, v_i)\end{aligned}$$

as required. □

We can now prove the main theorem.

Theorem 1. *The path produced by Algorithm 1 has length at most* $17 \cdot L_2(s,t)$.

Proof. Recall that $t = p_m$. Thus $L_1(p_m, \bar{p}_m) = 0$, and

$$\sum_{i=1}^{m-1} (L_1(p_{i+1}, \bar{p}_{i+1}) - L_1(p_i, \bar{p}_i)) = -L_1(p_1, \bar{p}_1). \tag{1}$$

Since p_1 is in C_1^s, and $\bar{p}_1$ is in $C_1^{p_1}$, we have that $\bar{p}_1$ is in C_1^s. Since we assume that ℓ_t^- is the closest diagonal to s, that gives us

$$L_1(s, \bar{p}_1) \le L_\infty(s,t) \le L_2(s,t). \tag{2}$$

Additionally, since p_1' is a point on ℓ_t^-, we have $L_1(p_1', t) = 2L_\infty(p_1', t)$. Observe that $L_\infty(s,t) > L_\infty(p_1, t) = L_\infty(p_1', t)$, thus $L_1(p_1', t) \le 2L_\infty(s,t)$, and $L_\infty(s,t) \le L_2(s,t)$. That gives us

$$L_1(p_1', t) \le 2 \cdot L_\infty(s,t) \le 2 \cdot L_2(s,t). \tag{3}$$

Thus $L_2(\mathcal{P}(s,t))$ is equal to

$$
\begin{aligned}
& L_2(\mathcal{P}\langle s, p_1\rangle) + \sum_{i=1}^{m-1} L_2(f_i) \\
&\leq L_1(s,p_1) + \sum_{i=1}^{m-1}(L_1(p_i,q_i) + L_1(q_i,p_{i+1})) \text{ (Lemma 4)} \\
&= L_1(s,p_1) + L_1(p_1,\bar{p}_1) - L_1(p_1,\bar{p}_1) + \sum_{i=1}^{m-1}(L_1(p_i,q_i) + L_1(q_i,p_{i+1})) \\
&= L_1(s,\bar{p}_1) + \sum_{i=1}^{m-1}(L_1(p_{i+1},\bar{p}_{i+1}) - L_1(p_i,\bar{p}_i)) + \sum_{i=1}^{m-1}(L_1(p_i,q_i) + L_1(q_i,p_{i+1})) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \text{(above is by (1))} \\
&= L_1(s,\bar{p}_1) + \sum_{i=1}^{m-1}(L_1(p_i,q_i) + L_1(q_i,\bar{p}_{i+1}) - L_1(p_i,\bar{p}_i)) \\
&= L_1(s,\bar{p}_1) + \sum_{i=1}^{m-1}\Phi(p_i,p_{i+1}) \\
&\leq L_1(s,\bar{p}_1) + 2\sum_{i=1}^{m-1}L_1(\bar{p}_i,\bar{q}_i) && \text{(Lemma 5)} \\
&\leq L_1(s,\bar{p}_1) + 8\cdot L_1(p_1',t) && \text{(Corollary 2)} \\
&\leq L_2(s,t) + 16\cdot L_2(s,t) && \text{(by (2) and (3))} \\
&\leq 17\cdot L_2(s,t)
\end{aligned}
$$

as required. □

4 Lower bound

In this section, we show that our analysis of the routing ratio of Algorithm 1 is tight: We will construct a set of points, together with two vertices s and t, such that the routing ratio of Algorithm 1 is arbitrarily close to 17. The construction is illustrated in Figs. 11, 12, 13, and 14. We forgo our general position assumption in order to make the demonstration of the lower bound simpler. For this particular example, if a vertex is on the boundary between two cones or two quadrants, we say that that vertex is in the *counter-clockwise* of the two cones or quadrants. Let $\epsilon > 0$ be an arbitrarily small number. Let $1 < a < b < c < d < e$ be (not necessarily consecutive) integers.

Let t be at coordinates $(0,0)$. Let s be at coordinates $(0,1)$. Vertex p_1 is at $(-1+\epsilon,1)$. See Figure 11. Place a sequence of vertices directly left of s at coordinates $(-\epsilon,1),(-2\epsilon,1),(-3\epsilon,1)...,(-1+\epsilon,1)=p_1$. This implies that s is not clean, so we take sweeping steps along this sequence of vertices (red dashed line) until we reach p_1. The path from s to p_1 has length $(1-\epsilon)$.

Directly below p_1 (and thus in $C_0^{p_1}$) there is a sequence of vertices at coordinates $(-1+\epsilon,1-\epsilon),(-1+\epsilon,1-2\epsilon),...,(-1+\epsilon,\epsilon)=p_a$, all of which are clean. Since p_1 is clean with respect to ℓ_t^-, we take greedy steps along this sequence to p_a (blue dashed line). Directly below p_a is vertex

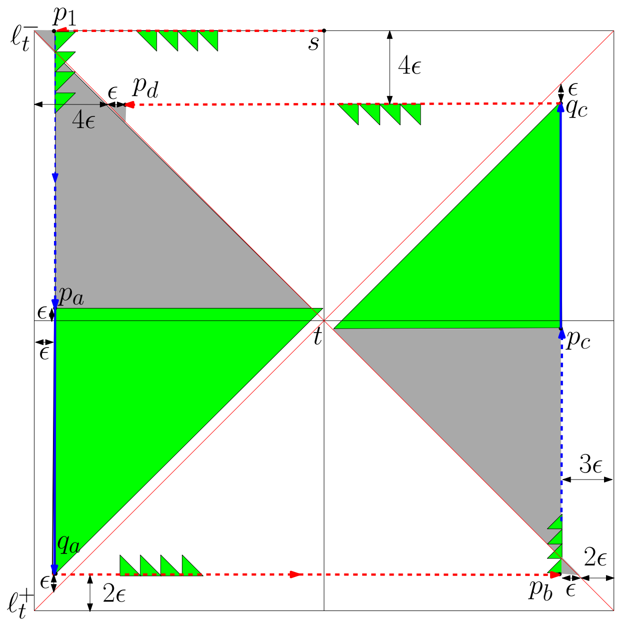


Figure 11: The path from s to p_d. Blue lines are greedy steps, red are sweeping steps towards ℓ_t^-. The grey and green regions are empty of vertices.

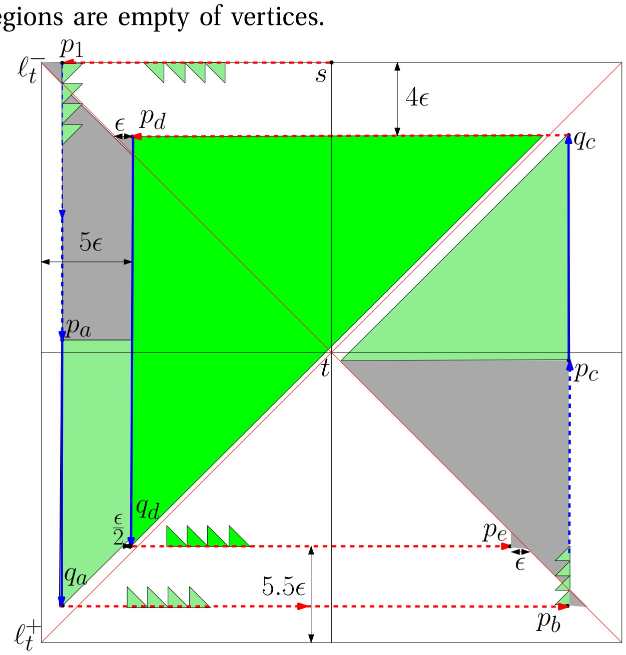


Figure 12: The path from p_d to p_e.

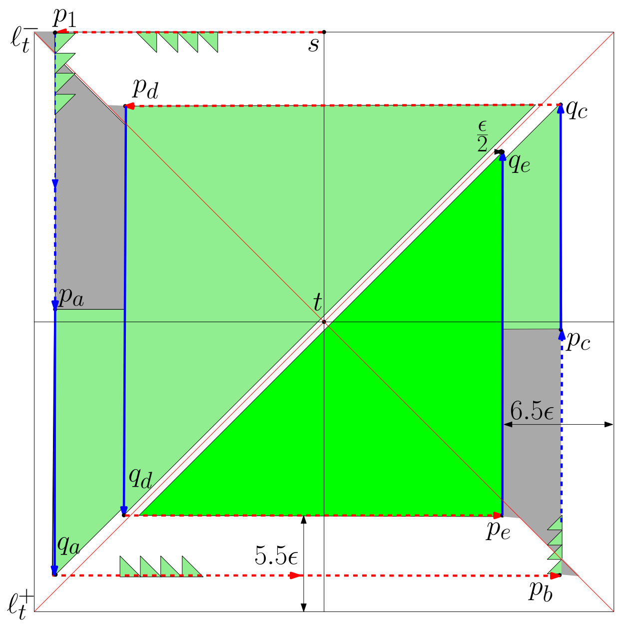


Figure 13: The last greedy edge (p_e, q_e).

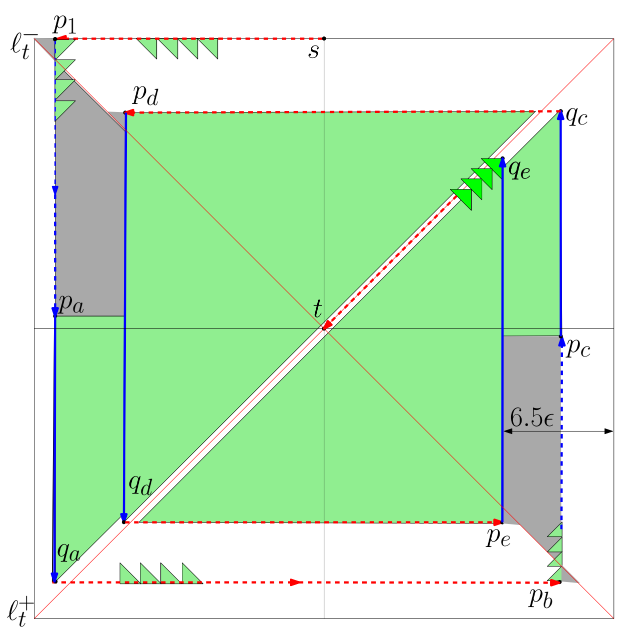


Figure 14: The final path from q_e to t.

$q_a = (-1+\epsilon, -1+2\epsilon)$. Vertex p_a is clean, so the next greedy step takes us to q_a (blue edge). The path from p_1 to q_a has length $2-2\epsilon$.

To the right of q_a is a sequence of vertices at coordinates $(-1+2\epsilon, -1+2\epsilon), (-1+3\epsilon, -1+2\epsilon), \ldots, (1-3\epsilon, -1+2\epsilon) = p_b$. The path from q_a to p_b has length $2-4\epsilon$.

There is a sequence of vertices directly above p_b at coordinates $(1-3\epsilon, -1+3\epsilon), (1-3\epsilon, -1+4\epsilon), \ldots, (1-3\epsilon, -\epsilon) = p_c$, all of which are clean. Thus we proceed along these vertices in a sequence of greedy steps from p_b to p_c (blue dashed path). From p_c we take a greedy step to $q_c = (1-3\epsilon, 1-4\epsilon)$ (blue edge). The path from p_b to q_c has length $2-6\epsilon$.

There is a sequence of vertices left of q_c at coordinates $(1-4\epsilon, 1-4\epsilon), (1-5\epsilon, 1-4\epsilon), \ldots, (-1+5\epsilon, 1-4\epsilon) = p_d$. The path from q_c to p_d has length $2-8\epsilon$.

In Figure 12 we take a greedy step from p_d to $q_d = (-1+5\epsilon, -1+5.5\epsilon)$ (blue edge). This edge has length $2-9.5\epsilon$.

To the right of q_d is a sequence of vertices at coordinates $(-1+6\epsilon, -1+5.5\epsilon), (-1+7\epsilon, -1+5.5\epsilon), \ldots, (1-6\epsilon, -1+5.5\epsilon) = p_e$. The path from q_d to p_e has length $2-11.5\epsilon$.

In Figure 13 we take a greedy step from p_e to $q_e = (1-6.5\epsilon, 1-7\epsilon)$ (blue edge). The edge (p_e, q_e) has length $2-12.5\epsilon$.

In Figs. 14 and 15 there are a sequence of vertices at $(1-7\epsilon, 1-7\epsilon), (1-7\epsilon+\epsilon', 1-7.5\epsilon), (1-7.5\epsilon, 1-7.5\epsilon), (1-7.5\epsilon+\epsilon', 1-8\epsilon) \ldots (0,0) = t$. We will define ϵ' in a moment. A sequence of clearing steps takes us from q_e to t along these vertices. Let $\delta+1$ be the number of horizontal edges in this sequence, and let γ be the number of edges with a vertical component. Let $\epsilon' = \epsilon/\delta$. Observe that $d_x(q_e, t) = 1-6.5\epsilon = (\delta+1)\cdot\epsilon/2$. The first horizontal edge has length $\epsilon/2$, and the remaining δ horizontal edges have length $\epsilon/2-\epsilon'$. Thus the total length of the horizontal edges is $1-6.5\epsilon-\delta\epsilon' = 1-6.5\epsilon-\epsilon = 1-7.5\epsilon$.

Observe that $d_y(q_e, t) = 1-7\epsilon = \gamma\cdot\epsilon/2$. Each of the γ vertical edges has length $>\epsilon/2$, since their vertical distance is $\epsilon/2$ and they are skewed from vertical, thus the total length of the vertical steps is at least $1-7\epsilon$. Thus the path from q_e to t has length at least $2-14.5\epsilon$.

The total length of these paths is at least $L_2(s,p_1)+L_2(p_1,q_a)+L_2(q_a,p_b)+L_2(p_b,q_c)+L_2(q_c,p_d)+L_2(p_d,q_d)+L_2(q_d,p_e)+L_2(p_e,q_e)+d_x(q_e,t)-\epsilon+d_y(q_e,t) = 1-\epsilon+2-2\epsilon+2-4\epsilon+2-6\epsilon+2-8\epsilon+2-9.5\epsilon+2-11.5\epsilon+2-12.5\epsilon+2-14.5\epsilon = 17-69\epsilon$. Since $L_2(s,t)=1$, by letting ϵ tend to 0 we can make the path arbitrarily close to $17\cdot L_2(s,t)$. This gives us the following theorem.

Theorem 2. *There exists a set of points such that the distance travelled by Algorithm 1 is at least $17-\epsilon$ for any $\epsilon > 0$.*

5 Removing the Diagonal-Bit

The algorithm, as presented in Section 2 uses one single bit to remember the diagonal of the destination t that is closest to the start vertex s. In this section, we show that without this single bit, the routing ratio increases to $\sqrt{290} < 17.03$.

Our modification of Algorithm 1 is to "hard code" the diagonal we route with respect to into our main and helper functions. For instance, the helper algorithms $Clean(u,t,N(u))$ and $Sweep(u,t,N(u))$ no longer require the 1-bit d as input. Instead, they always make their decisions with respect to ℓ_t^- regardless of the position of s. The routing algorithm $Greedy/Sweep(u,t,N(u))$ is now memoryless. It does not have a parameter d and it returns only a vertex $v \in N(u)$. The changes in the analysis are in Inequality (2), which becomes

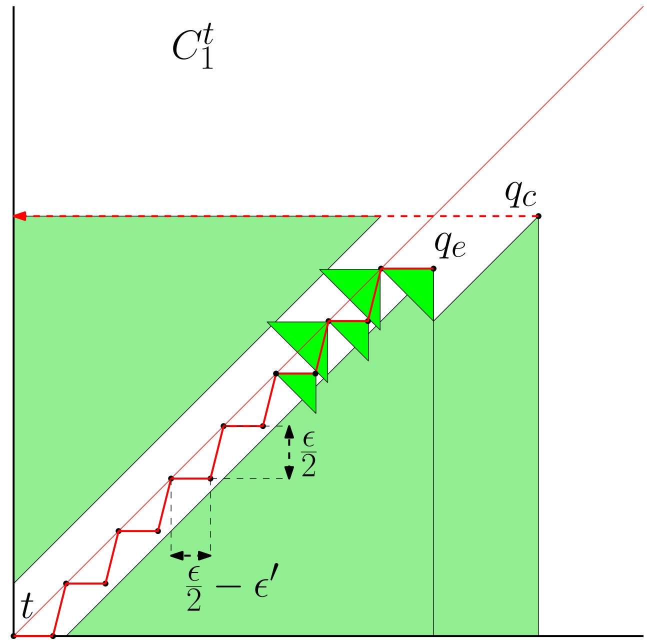


Figure 15: Details of the final path from q_e to t.

$$L_1(s, \bar{p}_1) \leq d_y(s,t) + L_\infty(s,t), \tag{4}$$

and in (3), which becomes

$$L_1(p_1', t) \leq 2 \cdot L_\infty(s,t). \tag{5}$$

If we replace (4) and (5) by (2) and (3) respectively in our proof of Theorem 1 we get $L_2(\mathcal{P}\langle s,t\rangle) \leq d_y(s,t) + 17 \cdot L_\infty(s,t)$. Let $\gamma = (d_y(s,t) + 17 \cdot L_\infty(s,t))/L_2(s,t)$. The routing ratio is thus the maximum of γ. Let u be the point at $(d_x(s), d_y(t))$, and let $\theta = \angle uts$. We can rewrite γ as $\sin\theta + 17 \cdot \cos\theta = \sqrt{17^2 + 1^2} \cdot \sin(\theta + \arctan(17))$ for $0 \leq \theta \leq \pi/4$. This is maximized at $\theta = \arctan(\frac{1}{17})$ with a value of $\sqrt{290}$. Thus we have the following theorem.

Theorem 3. *With no bits of memory, and using a fixed diagonal ℓ_t^-, Algorithm 1 outputs a path from s to t with length at most $\sqrt{290} \cdot L_2(s,t)$.*

If we refer to the lower bound proof in Section 4, we can adjust it to this new bound by moving s to the right until st forms an angle of $\arctan(\frac{1}{17})$ with the positive y-axis. Thus, in this case, we can get arbitrarily close to $\sqrt{290}$.

6 Conclusion

We have presented a simple online local routing algorithm for $\overrightarrow{\Theta_4}$-graphs that achieves a routing ratio of 17 using knowledge of the destination and one bit of information, and $\sqrt{290} < 17.03$ using only knowledge of the destination. Although we have presented the first such algorithm on $\overrightarrow{\Theta_4}$-graphs and also improved the spanning ratio of $\overrightarrow{\Theta_4}$- and Θ_4-graphs from 237 down to 17, we conjecture that this upper bound both on the routing ratio and spanning ratio is not tight. Given that 7 [3] is the best known lower bound for the spanning ratio of Θ_4, the actual spanning ratio remains unknown.

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*, 47(9):910–917, 2014.

[2] Hugo Akitaya, Ahmad Biniaz, and Prosenjit Bose. On the spanning and routing ratio of the directed θ_6-graph. In *ICCG, accepted*, 2020.

[3] Luis Barba, Prosenjit Bose, Jean-Lou De Carufel, André van Renssen, and Sander Verdonschot. On the stretch factor of the theta-4 graph. In Frank Dehne, Roberto Solis-Oba, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures - 13th International Symposium, WADS 2013, London, ON, Canada, August 12-14, 2013. Proceedings*, volume 8037 of *Lecture Notes in Computer Science*, pages 109–120. Springer, 2013.

[4] Nicolas Bonichon, Prosenjit Bose, Jean-Lou De Carufel, Vincent Despré, Darryl Hill, and Michiel H. M. Smid. Improved routing on the delaunay triangulation. In *ESA*, volume 112 of *LIPIcs*, pages 22:1–22:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[5] Nicolas Bonichon, Prosenjit Bose, Jean-Lou De Carufel, Ljubomir Perkovic, and André van Renssen. Upper and lower bounds for online routing on delaunay triangulations. *Discrete & Computational Geometry*, 58(2):482–504, 2017.

[6] Nicolas Bonichon, Cyril Gavoille, Nicolas Hanusse, and David Ilcinkas. Connections between theta-graphs, delaunay triangulations, and orthogonal surfaces. In *WG*, volume 6410 of *Lecture Notes in Computer Science*, pages 266–278, 2010.

[7] Prosenjit Bose, Paz Carmi, and Stephane Durocher. Bounding the locality of distributed routing algorithms. *Distributed Computing*, 26(1):39–58, 2013.

[8] Prosenjit Bose, Jean-Lou De Carufel, and Olivier Devillers. Expected complexity of routing in θ_6 and half-θ_6 graphs. *J. Comput. Geom.*, 11(1):212–234, 2020.

[9] 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. *Theor. Comput. Sci.*, 616:70–93, 2016.

[10] Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Competitive routing in the half-theta-6-graph. In *SODA*, pages 1319–1328. SIAM, 2012.

[11] Prosenjit Bose, Rolf Fagerberg, André van Renssen, and Sander Verdonschot. Optimal local routing on delaunay triangulations defined by empty equilateral triangles. *SIAM J. Comput.*, 44(6):1626–1649, 2015.

[12] Prosenjit Bose, Darryl Hill, and Aurélien Ooms. Improved spanning on theta-5. *CoRR*, abs/2106.01236, 2021.

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

[14] Prosenjit Bose and Pat Morin. Online routing in triangulations. *SIAM J. Comput.*, 33(4):937–951, 2004.

[15] Prosenjit Bose, Pat Morin, André van Renssen, and Sander Verdonschot. The theta-5-graph is a spanner. *CoRR*, abs/1212.0570, 2012.

[16] Nicolas Broutin, Olivier Devillers, and Ross Hemsley. Efficiently navigating a random delaunay triangulation. *Random Struct. Algorithms*, 49(1):95–136, 2016.

[17] Dan Chen, Luc Devroye, Vida Dujmovic, and Pat Morin. Memoryless routing in convex subdivisions: Random walks are optimal. *Comput. Geom.*, 45(4):178–185, 2012.

[18] Paul Chew. There is a planar graph almost as good as the complete graph. In *Symposium on Computational Geometry*, pages 169–177. ACM, 1986.

[19] Paul Chew. There are planar graphs almost as good as the complete graph. *J. Comput. Syst. Sci.*, 39(2):205–219, 1989.

[20] Kenneth L. Clarkson. Approximation algorithms for shortest path motion planning (extended abstract). In *STOC*, pages 56–65. ACM, 1987.

[21] J. Mark Keil. Approximating the complete euclidean graph. In Rolf G. Karlsson and Andrzej Lingas, editors, *SWAT 88, 1st Scandinavian Workshop on Algorithm Theory, Halmstad, Sweden, July 5-8, 1988, Proceedings*, volume 318 of *Lecture Notes in Computer Science*, pages 208–213. Springer, 1988.

[22] J. Mark Keil and Carl A. Gutwin. Classes of graphs which approximate the complete euclidean graph. *Discrete & Computational Geometry*, 7(1):13–28, 1992.

[23] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass routing on geometric networks. In *CCCG*, 1999.

[24] Nawar M. El Molla. *Yao spanners for wireless ad-hoc networks*. PhD thesis, Villanova University, Pennsylvania, USA, 2009.

[25] Stefan Ruhrup. Theory and practice of geographic routing. *Chapter 5 in Ad Hoc and Sensor Wireless Networks: Architectures, Algorithms and Protocols*, 2009.

[26] Jim Ruppert and Raimund Seidel. Approximating the d-dimensional complete Euclidean graph. In *Proceedings of the 3rd Canadian Conference on Computational Geometry*, CCCG 1991.