Memory Requirements for Local Geometric Routing and Traversal in Digraphs

M. Fraser [*] E. Kranakis [†] J. Urrutia [‡]

Abstract

Local route discovery in geometric, connected, *undirected* plane graphs is guaranteed by the Face Routing algorithm. The algorithm is local and geometric in the sense that it is executed by an agent moving along a network and using at each node only information about the current node (incl. its position) and a finite number of others (independent of graph size). Local geometric traversal algorithms also exist for undirected plane graphs. In this paper we show that no comparable routing or traversal algorithms exist for the class of strongly connected plane *directed* graphs (digraphs). We construct a class of digraphs embedded in the plane for which either local routing or local traversal requires $\Omega(n)$ memory bits, where n is the order of the graph. We discuss these results in light of finding a suitable model for mobile ad hoc networks with uni-directional edges, showing in the extended version of this paper that digraphs for which the $\Omega(n)$ lower bound holds occur even in the class of embedded digraphs arising out of a very conservative model.

1 Introduction

Face Routing was proposed in [9] as an algorithm that accomplishes discovery of routes locally in geometrically embedded, connected, undirected planar graphs. It guarantees delivery in time $O(n)$ where n is the order of the graph. Since its introduction in 1999, there have been various studies extending the class of graphs over which variations of this algorithm can be applied. In the 2D *undirected* case, these are enough to handle most graphs occuring as models of 2D mobile adhoc networks: Quasi-Planar graphs, Unit Disk Graphs (see survey in [11]) and so called Quasi-Unit Disk Graphs (with radial coefficient $d > \frac{1}{\sqrt{2}}$).

Traversal algorithms have also been well studied for undirected graphs in the plane. For example, for planar embedded graphs [4] and quasi-planar subdvisions [3] local geometric algorithms are known.

In the 3D case relatively little has been done until now. Recently M. Fraser extended Face Routing to undirected graphs embedded on known surfaces of higher genus [8] and Durocher et al. [5] extended it to Unit Ball Graphs (UBG's) with nodes bounded within a relatively thin slab of space (of thickness at most $1/\sqrt{2}$). Also a recent paper by R. Wattenhofer [6] proposes a partially randomized routing algorithm for 3D ad hoc networks.

There has also been little progress on the 2D *directed* case. Some special classes of 2D directed graphs for which local geometric routing algorithms are known include Eulerian and Outerplanar *digraphs* [2]. The question of existence of local geometric algorithms for general geometric digraphs has remained open.

It is important to note that there are several variations in the use of the term **"local"** in the ad-hoc community. In this paper, by a *local algorithm* we mean a deterministic algorithm in which an agent moving along the network uses at each node only information stored at that node concerning the node itself and its neighbours (including current position) together with a certain amount M of memory which is carried as overhead in the message. No further information regarding the rest of the network is available to the agent, e.g. neither the number of nodes, where they are located, the topology of the network, nor any other global information. The agent is not allowed to alter the state of a node.

By *local routing algorithm*, we mean a local algorithm as defined above where the agent should reach a node t started from a node s, given the position or ID of t.

In Face Routing, messages are allowed to carry with them the ID's (or typically the positions) of sender and destination as well as two other nodes and a number representing a distance. This means M has typically been restricted to $O(\log n)$ as part of the definition of "local". Upon arrival to a node, a message can use the local information stored at a node plus the information it carries along.

We show that planar embedded directed graphs (digraphs), unlike their undirected counterparts, do not admit local geometric routing *or* traversal algorithms

[*]Mathematics Dept., University of Colima, Colima, MEXICO. fraser.maia@gmail.com

[†]Schoool of Computer Science, Carleton University, Ottawa, CANADA. Research supported in part by NSERC and MITACS grants. kranakis@scs.carleton.ca

[‡]Inst. of Mathematics, National Autonomous University of Mexico (UNAM), Mexico City, MEXICO, Research supported in part by MTM2006-03909 (Spain) and CONACYT of Mexico, Proyecto SEP-2004-Co1-45876. urrutia@matem.unam.mx

with carried memory $M \in O(\log n)$ (for non-geometric digraphs the corresponding result for traversal was shown by Fraigniaud and Ilcinkis [7]). We show this by constructing a class $\mathcal{C}$ of embedded digraphs for which either geometric traversal or route discovery requires $\Omega(n)$ memory bits.

It is important to notice that in practice, many wireless and ad-hoc networks *do have* oriented links. If a network has transmitters with different broadcast powers, it may happen that a node can receive messages from a node to which it cannot send.

In the extended version of this paper we describe a Face-Routing-like routing algorithm which is correct for plane digraphs and is executed by an agent carrying $O(n)$ memory.

Our main objective in this paper is to prove the following result:

Theorem 1 *There is a class of bounded degree (i.e., both in-degree and out-degree) strongly connected embedded digraphs in the plane for which no deterministic local traversal or routing algorithm (even a geometric one) exists. In particular, if n is the number of vertices of a graph in this class then every traversal or routing algorithm on the graph requires $\Omega(n)$ bits of memory.*

Here we use the term *traversal* in its weakest sense: namely requiring only that all nodes must be visited and making no assumption about return to start.

The result of Fraigniaud and Ilcinkas [7] addressed only *non-geometric* graphs and so hope remained that the use of geometry might make possible an $O(\log n)$ algorithm for at least planar embedded digraphs. We settle this issue in the negative.

We remark that in the absence of planar geometry the routing problem is, loosely speaking, equivalent to that of traversal since an adversary may connect a destination node to any other graph node; however, this is no longer the case for planar embedded graphs, since not all nodes are reachable from a given destination position without introducing edge-crossings.

We also note that while Face Routing provides a linear time, $O(\log n)$ memory local routing algorithm for planar embedded, undirected graphs, when it was introduced in 1999 there was by contrast no known $O(\log n)$ local routing algorithm for abstract undirected graphs. In 2005, one was established by Reingold (using universal exploration sequences, [10]). But, not only is this method much more complicated than Face Routing, its time still remains polynomial with high exponent. Planar geometry thus has a significant impact in reducing the complexity of the routing problem in the undirected case. The point of the present paper is to show that the benefit of geometry is *insuficient* to reduce memory requirements of local routing to logarithmic in the directed case.

2 Main Result

To construct a class with the properties stated in Theorem 1, we define for each $n \in \mathbb{N}$ an embedded digraph, called a *random lock*, which is a "randomized" embedding of a variation on the combination lock (combination locks have unbounded in-degree, i.e., $\Theta(n)$, see [7], and this variation just bounds the degree by merging wigs, albeit in a random way). To this end we first define a kinked embedded lock.

Definition 2.1 [Kinked Embedded Lock] *Let $x = x_1 x_2 \cdots x_{n-1}$ be any bit string of length $n-1$ (notational warning: we start with index 1). Suppose the 1 bits of x occur for indices $i_0, \ldots, i_k$ and the 0 bits for indices $j_0, \ldots, j_K$. A kinked embedded lock determined by x is an embedded digraph $\varphi(H_x)$ where $H_x = (\bar{V}_x, \bar{E}_x \cup \{e\})$ is an abstract digraph with*

- *set of vertices $\bar{V}_x = \{u_0, u_1, u_2, \ldots, u_{n-1}, u_n, w, v_1, v_2, \ldots, v_{n-1}\}$,*
- *set of directed edges $\bar{E}_x = E'_n \cup E''_n$ where, $E'_n = \{(u_i, u_{i+1}) : 0 \leq i \leq n-1\} \cup \{(u_i, v_i) : 1 \leq i \leq n-1\}$ and $E''_n = \{(u_n, v_{i_k}), (u_n, v_{j_K})\} \cup \{(v_{i_q}, v_{i_{q-1}}) : 1 \leq q \leq k\} \cup \{(v_{j_q}, v_{j_{q-1}}) : 1 \leq q \leq K\} \cup \{(v_{i_0}, u_0), (v_{j_0}, u_0)\}$[1],*
- *and one bi-directional edge $e = (u_n, w)$,*

and φ is any embedding $\varphi : H_x \rightarrow \mathbb{R}^2$ satisfying the following recursive relation:

- *$\varphi(u_0) = (0,0), \varphi(u_1) = (1,0)$ and*
- *for $1 \leq i \leq n-1$ if u_i has coordinates (a,b) then:*

  $\varphi(u_{i+1}) = (a+1, b+1), \varphi(v_i) = (a+1, b-1)$ for $x_i = 0$, and

  $\varphi(u_{i+1}) = (a+1, b-1), \varphi(v_i) = (a+1, b+1)$ for $x_i = 1$,

and $\varphi(w) = (n+1, 0)$, with all edges being straight line segments. We denote by K_x the kinked embedded lock determined by x.

Note that the order of the kinked embedded lock K_x determined by an $(n-1)$-bit string x is $|K_x| = 2n+1$. It is easily checked that φ as described is an embedding (i.e. that there are no edge crossings).

An example of the requirements on φ for the string $x = 0111$ is depicted in Figure 1. This figure includes only the edges $\varphi(E'_n)$.

The full embedded digraph $\varphi(H_x)$ is obtained from the one shown in Figure 1 by adding embedded directed edges from $\varphi(E''_n)$, namely adding edges "all along the top" and "all along the bottom" (to connect subsequent

[1] where undefined elements are assumed omitted

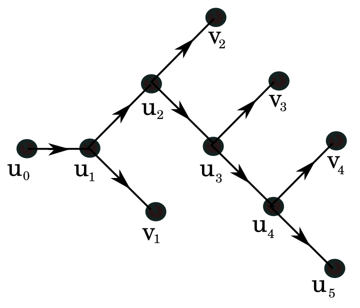

Figure 1: An example of the requirements on $\varphi(E'_n)$ for the string $x = 0111$.

upper vertices v_{i_q} and subsequent lower vertices v_{j_q} as given by E''_n) and then adding the bi-directional edge e connecting w to u_n. This is depicted in Figure 2. Note that the position of $\varphi(w)$ does not depend on the actual bit string x, just on n.

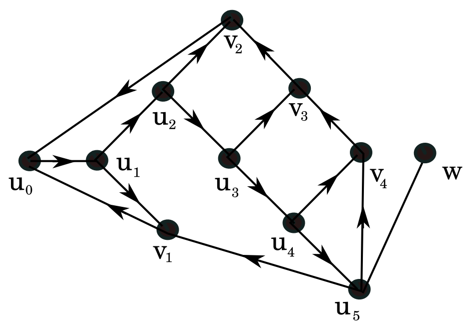

Figure 2: The kinked embedded lock K_x determined by the bit string $x = 0111$.

We will now show that if we use random bit strings to construct kinked embedded locks we obtain a class of geometrically embedded digraphs – random locks – which satisfy the requirements of the theorem. The idea is that any traversal or routing algorithm, $\mathcal{A}$, on such graphs must be able to reach w from u_0 knowing at most the positions of these two points and local information. To do so, it must be able to reach u_n using only this data. This allows one to generate the underlying random bit string and the desired lower bound follows.

Proof. *(of Theorem 1)* Let x be an infinite Kolmogorov random (i.e. algorithmically random) bit string $x = x_1x_2\dots$ (notational warning: we start with index 1). Then for any $n \in \mathbb{N}$, $x^n := x_1x_2\cdots x_n$ has Kolmogorov complexity at least n. In other words, any algorithm which can generate this string has space complexity at least n.

Let $\mathcal{C} = \{K_{x^n} : n \in \mathbb{N}\}$ be the set of kinked embedded graphs determined by all finite initial substrings of x. $\mathcal{C}$ consists of bounded-degree embedded digraphs. We call these *random locks*. Recall that $|K_{x^{n-1}}| = 2n + 1$, so the order of these graphs is linear in n.

Assume that $\mathcal{A}$ is a correct geometric traversal or routing algorithm for $\mathcal{C}$ using memory $M(n)$ bits of (transported and working) memory for the random lock $K_{x^{n-1}} \in \mathcal{C}$ determined by the $(n-1)$-bit string x^{n-1}. Since the size of these graphs is linear in n we need only show $M(n) \in \Omega(n)$.

Starting at u_0, $\mathcal{A}$ must in particular be able to reach w using only local geometric data (coordinates of current node and its neighbours) *plus* $M(n)$ bits of memory, which may include local geometric data gathered along its route (coordinates of vertices visited and their neighbours) or even coordinates of w (in the case of a routing algorithm) . Note that knowing the coordinates of w just corresponds to knowing $n + 1$.

To reach w from u_0, $\mathcal{A}$ must eventually – i.e. at some time T – be at u_1 and then follow exactly the path $u_1, u_2, \dots u_n, w$. To be precise, let T be the time just before computing an exit vertex at u_1 and let S_T be the state of $\mathcal{A}$ at this time, i.e. the stored memory at this moment including the current line number L_T that has just been executed at this time in the program P encoding $\mathcal{A}$. Storing S_T requires memory at most $M(n)$.

Let $\mathcal{G}$ be an efficient algorithm which given a pair of positive integers (x, y) and a bit b calculates $(x, y), (x+1, y+1-2b), (x+2, y+2-2b), (x+2, y-2b)$. We may assume $\mathcal{G}$ has space complexity linear in the size of (x, y). If we assume the numbers $x, y \leq n$, then the space complexity of $\mathcal{G}$ is $O(\log n)$. Note that $\mathcal{G}$ is essentially a *geometry simulator* for $\mathcal{A}$ as long as $\mathcal{A}$ travels along a path consisting of only u_i's. More precisely, whenever $\mathcal{A}$ is at some u_i for which it will next advance to u_{i+1}, if it passes to $\mathcal{G}$ the coordinates of u_i and the value 0 or 1 indicating its decision at u_i (to turn respectively up or down) then $\mathcal{G}$ will output the coordinates of u_{i+1} and of all its neighbours. This is in fact all the geometric data that the algorithm $\mathcal{A}$ can use to make its decision at u_{i+1}.

Using $\mathcal{A}$ and $\mathcal{G}$, we now define an algorithm $\mathcal{B}$ of space complexity $M(n) + O(\log n)$ which generates the bit string x^n. This includes the constant overhead to store instructions for all three algorithms (incl. passing of data between them, conversion of left/right instructions to bits etc...).

The algorithm $\mathcal{B}$ takes input $\mathtt{S_T}$, $\mathtt{F} = n - 1$. Its working data will include the coordinates of four points in the plane: $\mathtt{u_{curr}}$ and $\mathtt{u_{left}}$, $\mathtt{u_{rightUp}}$, $\mathtt{u_{rightDn}}$. It initializes these to $(1, 0), (0, 0), (2, 1), (2, -1)$ respectively, these being the coordinates of u_1, u_0 and the upper then lower of u_2, v_1. The algorithm then proceeds as follows:

1. Set $i = 1$. Place $\mathcal{A}$ in state $\mathtt{S_T}$.

2. Run $\mathcal{A}$ for a single iteration giving it local pseudo-geometric data $u_{\texttt{curr}}$, $u_{\texttt{left}}$, $u_{\texttt{rightUp}}$, $u_{\texttt{rightDn}}$ in order to generate a bit b encoding whether it will turn upwards or downwards ($b = 0$ or 1 respectively). This corresponds to deciding which of the two neighbours to the right is labelled u_2.

3. Output b. If $i \geq F$, exit; else increment i by one.

4. Call $\mathcal{G}$ with input $u_{\texttt{curr}}$ and b. Let its output be used to set $u_{\texttt{left}}$, $u_{\texttt{curr}}$, $u_{\texttt{rightUp}}$, $u_{\texttt{rightDn}}$ respectively. Go to 2.

Recall that when the algorithm $\mathcal{A}$ is started in state S_T using local geometric data from u_1, it performs exactly the run $u_1, u_2, \ldots, u_{n-1}$. Thus the output of $\mathcal{B}$ will be exactly the bit string x^n. Since the complexity of $\mathcal{B}$ is $M(n) + O(\log n)$, and x^n is Kolmogorov random we must have $M(n) \in \Omega(n)$. $\square$

k-Locality We remark that a k-local version of Theorem 1 also holds: namely, even a geometric routing or traversal algorithm with access to geometric data k hops away from its current location (k, any fixed positive integer) requires $\Omega(n)$ memory bits. This is proved by inserting k extra vertices on each of the edges $\{(u_0, u_1), (u_i, v_i), (u_i, u_{i+1}) : i \in \mathbb{N}, 1 \leq i \leq n-1\}$ (i.e. k-subdividing these edges) in the random locks above and adjusting the geometry simulator accordingly so as to generate these extra points as well.

Remark In fact, it is easy to construct another class $\mathcal{C}'$ of digraphs which can be used to prove Theorem 1 and which consists of plane digraphs arising from wireless networks with just two different broadcast radii. We discuss this further in the extended version of this paper.

3 Conclusions

Since the introduction of Face Routing, considerable efforts have been made trying to extend it to wider families of networks. The robustness and simplicity of Face Routing make it a tantalizing model to strive for, as the computational requirements in network communications are thus greatly reduced, e.g. drastic reductions on the amount of traffic, and the elimination of costly mechanisms such as routing tables that have to be updated periodically [11]. It is worth mentioning here that the locality condition, as stated here, implies a robustness beyond existing modes; since a message is not aware of the existence of most nodes of a network, it is unaffected by their potential failures (as long as the network remains connected).

Especially, it would be desirable to achieve these properties of Face Routing for cellular networks in $\mathbb{R}^3$ or directed networks as these settings are a reality in telephony and sensor networks nowadays. Unfortunately, extending Face Routing beyond undirected networks in the plane has proved to be a formidable task, and all the evidence suggests it may not be possible. As far as we know, this is the first result that proves some *intrinsic* limitations of the model. The results presented here might give a hint as to why previous attempts have been unsuccessful, and might in fact dictate restrictions that should be imposed on wireless and, in general, communication networks to take full advantage of the benefits of Face Routing.

References

[1] P. Bose, P. Morin, I. Stojmenic, and J. Urrutia, *Routing with guaranteed delivery in ad hoc wireless networks.* In 3rd Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication (DIALM '99), pp. 48-55 (1999).

[2] E. Chavez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, J. Urrutia, *Route Discovery with Constant Memory in Oriented Planar Geometric Networks.* Networks, Volume 48, Issue 1, pages 7-15.

[3] E. Chavez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, J. Urrutia *Traversal of a quasi-planar subdivision without using mark bits.* Journal of Interconnection Networks Vol. 5, Issue 4, pp. 395 - 407, 2004.

[4] M. de Berg, M. van Kreveld, R. van Oostrum, and M. Overmars. *Simple traversal of a subdivision without extra storage.* International Journal of Geographic Information Systems, 11:359373, 1997.

[5] S. Durocher, D. Kirkpatrick, L. Narayanan, *On Routing with Guaranteed Delivery in Three-Dimensional Ad Hoc Wireless Networks.* In proceedings of ICDCN 2008.

[6] R. Flury and R. Wattenhofer, *Randomized 3D Geographic Routing.* In 27th Annual IEEE Conference on Computer Communications (INFOCOM), Phoenix, USA, April 2008.

[7] P. Fraigniaud and D. Ilcinkas, *Digraphs Exploration with Little Memory.* In 21st Symposium on Theoretical Aspects of Computer Science (STACS'04), LNCS 2296, pages 246-257, 2004.

[8] M. Fraser, *Local Routing on Tori.* In Proceedings of ADHOCNOW, held in Morelia Mexico, Sep 24-26, 2007, pp. 1- 14, Springer LNCS 4686, 2007.

[9] E. Kranakis, H. Singh, J. Urrutia, *Compass Routing in Geometric Graphs*, in proceedings of 11th Canadian Conference on Computational Geometry, CCCG-99, pages 51-54, Vancouver Aug. 15-18, 1999.

[10] O. Reingold, *Undirected ST-Connectivity in Log-Space*, Proceedings of the 37th annual ACM symposium on Theory of Computing, pp. 376-385, ACM, 2005.

[11] J. Urrutia, *Local solutions for global problems in wireless networks.* Journal of Discrete Algorithms, 5, pp. 395-407, 2007.