A Self-Stabilizing and Local Delaunay Graph Construction*

Riko Jacob[1], Stephan Ritscher[1], Christian Scheideler[2], Stefan Schmid[2]

[1] Institut für Informatik, Technische Universität München, D-85748 Garching, Germany

jacob@in.tum.de, ritsches@in.tum.de

[2] Department of Computer Science, University of Paderborn, D-33102 Paderborn, Germany

scheideler@upb.de, schmiste@mail.upb.de

Abstract

This paper studies the construction of self-stabilizing topologies for distributed systems. While recent research has focused on chain topologies where nodes need to be linearized with respect to their identifiers, we go a step further and explore a natural 2-dimensional generalization. In particular, we present a local self-stabilizing algorithm that constructs a *Delaunay graph* from any initial connected topology and in a distributed manner. This algorithm terminates in time $O(n^3)$ in the worst-case. We believe that such self-stabilizing Delaunay networks have interesting applications and give insights into the necessary geometric reasoning that is required for higher-dimensional linearization problems.

1 Introduction

Open distributed systems such as peer-to-peer systems are often highly dynamic in the sense that nodes join and leave continuously. In addition to these natural membership changes, a system is sometimes under attack, e.g., a botnet may block entire network fractions by a denial-of-service attack. For these reasons, there is a considerable scientific interest in robust and "self-healing" topologies that can be maintained in a distributed manner even under high churn.

An important concept to build robust networks is *topological self-stabilization*: A self-stabilizing network can provably recover from *any* connected state, that is, eventually the network always returns to a desirable (to be specified) state. Despite its relevance, topological self-stabilization is a relatively new area and today, we still know only very little about the design of self-stabilizing algorithms. In particular, while much existing literature focuses on *eventual* stabilization, the required *convergence times* are still not well understood.

*Research supported by the DFG project SCHE 1592/1-1.

Recently, researchers have made progress in the field of *graph linearization* where nodes need to be arranged in a chain network which respects the node identifiers. In this paper, we go one step further and explore the *2-dimensional* case. We assume nodes are distributed in the Euclidean plane and are arbitrarily connected. A natural 2-dimensional analogon of linearization is the *Delaunay graph*, whose edge set includes all nearest neighbor connections between node pairs. Delaunay graphs are an important graph family in various CS domains, from computational geometry to wireless networking. This is due to their desirable properties such as locality, sparseness or planarity. We find that while insights from graph linearization are useful for self-stabilizing Delaunay graphs as well, the construction and analysis is more involved, requiring a deeper geometric reasoning.

1.1 Related Work

Researchers in the field of self-stabilization study algorithms that provably converge to a desirable system state from *any* initial configuration. In the seminal work by E.W. Dijkstra in 1974 [7], the problem of self-stabilization in a token ring is examined. Subsequently, many aspects of distributed systems have been explored from a self-stabilization point of view, including communication protocols, graph theory problems, termination detection, clock synchronization, and fault containment. Also general techniques for self-stabilization have been considered: In [2], Awerbuch and Varghese showed that every local algorithm can be made self-stabilizing if all nodes keep a log of the state transitions until the current state.

However, much of this work is not applicable to scenarios where faults include changes in the *topology* (e.g., see [9] for an early work on topological self-stabilization): A single fault may require the involvement of all nodes in the system and is hence expensive to repair. To reduce this overhead, researchers have started to study so-called superstabilizing protocols [8]. Topological self-stabilization is still in its infancy. Often, recovery algorithms do not work generally but only from certain degenerate network states (see, e.g., the technical report of the Chord network [18]). A notable recent exception is [14] which describes a truly self-stabilizing algorithm for skip graphs. Unfortunately, however, skip graphs do not maintain locality in the sense that nodes which are close in the metric space are also close with respect to the hop distance, and therefore cannot be used in our context.

In order to shed light onto the fundamental principles enabling provable topological self-stabilization, researchers have started to examine the most simple networks such as *line or ring graphs* (e.g., [5, 10]). Our paper goes one step further and initiates the study of self-stabilizing constructions of 2-dimensional graphs. As a case study, we consider the important family of Delaunay graphs. We assume nodes have (x, y) coordinates and are distributed in the Euclidean plane. As Delaunay graphs include all nearest neighbor edges, our algorithms also involve a kind of 2-dimensional linearization. However, it turns out that the problem is more involved, and the reasoning requires geometric techniques. Still we are able to prove

a $O(n^3)$ convergence time in the worst-case.

Delaunay graphs are known for almost a century now [6] and there exists a large body of literature. For an introduction and basic algorithms, see, e.g., the handbook by Goodman et al. [11]. The recent interest in ad-hoc networks has brought the Delaunay graph back to the limelight. Led by the energy challenges—nodes in ad-hoc networks often have a limited power supply—researchers have proposed numerous approaches for *topology control* [12, 20]. In [13], Hu presents a local topology-control algorithm for Delaunay triangulations in packet radio networks. Using neighbor negotiations, the graph is also made degree-bounded. In [4], competitive memory-less online routing algorithms on Delaunay structures are proposed. Due to the expensive distributed construction of Delaunay graphs and the sometimes long edges, alternative topologies have been proposed, some of which contain the Delaunay graph as a subgraph [15]. Note that, if the initial neighbors of a node are local (which is typically the case in wireless networks), self-stabilizing constructions are simple and can even be performed in constant time [16, 21]. In contrast to the wireless constructions, in this paper, we do not assume that nodes initially have connections to local neighbors. Rather, nodes can be connected to *any* nodes on the metric space. In this sense, our algorithms can be understood as a *topology control mechanism for wireline systems.*

1.2 Our Contributions

This paper presents the first self-stabilizing algorithm to build a Delaunay graph from *any* weakly connected network. Our algorithm is *local* in the sense that nodes are only allowed to communicate with their topological neighbors. Besides correctness, we are able to derive a $O(n^3)$ worst-case bound on the convergence time (i.e., number of communication rounds). We believe that this result has interesting implications, and that our geometric reasoning can give general insights into the design of higher-dimensional "nearest-neighbor graphs" respecting the closeness of nodes in a self-stabilizing manner. If the initial network contains the Delaunay graph, the convergence time is at most n rounds.

Compared to the trivial strategy to obtain a complete graph in $O(\log n)$ rounds in a first phase and then compute the Delaunay graph "locally" at each node in a second phase, our algorithm provides several advantages. First of all, it is not necessary to distinguish between different execution phases: Each node will perform updates according to the same set of rules at any time; only like this, the algorithm is truly self-stabilizing. Furthermore, our algorithm can deal efficiently with small topology changes: If only a small number of nodes joins or leaves, the topology is repaired *locally*; a complete re-computation is not needed. Finally the simulations show that the maximal degree and the total number of edges remain rather small in general. This keeps the resource requirements at each node small.

1.3 Paper Organization

The remainder of this paper is organized as follows. We describe our formal model and introduce some technical preliminaries in Section 2. Our algorithm is presented in Section 3, and it is subsequently analyzed in detail (Section 4). Section 5 reports on our simulation results. The paper is concluded in Section 6.

2 Model and Preliminaries

This section first introduces some notations and definitions from geometry. Subsequently, the Delaunay graph is introduced together with some important properties. In this paper, we will consider non-degenerate cases only, that is, we assume that no two nodes are at the same location, no three points are on a line, and no four points are on a circle.

2.1 Geometry

We consider the 2-dimensional Euclidean space $\mathbb{R}^2$. The *scalar product* is written as $\langle \cdot, \cdot \rangle$ and the *Euclidean norm* (the distance from the origin) is given by $\|x\| = \sqrt{\langle x, x \rangle}$. We make use of the following notation. Let $B(x, r)$ denote the *disk* (or ball) with center $x \in \mathbb{R}^2$ and radius $r \in \mathbb{R}$, i.e., $B(x, r) := \{y \in \mathbb{R}^2 : \|x - y\| \leq r\}$. Note that the border explicitly belongs to the ball in our model, and hence, a point $y \in B(x, r)$ may lie on the border. $C(x, y) := B(\frac{1}{2}(x + y), \frac{1}{2}\|x - y\|)$ is the disk *between* $x, y \in \mathbb{R}^2$. Similarly, $C(x, y, z) := B(c, r)$ with $r = \|x - c\| = \|y - c\| = \|z - c\|$ is the disk defined by non-collinear $x, y, z \in \mathbb{R}^2$. For a vector $x \neq 0$ we define $0 \neq \perp x \in R^2$ to be the perpendicular, i.e., $\langle x, \perp x \rangle = 0$. Note that $\perp x$ is unique up to constant factors.

By $\angle xzy$ we denote the area spanned by the vectors x and y attached to z, i.e., the area that can be expressed as a linear combination of the vectors x and y with non-negative factors. In particular, $\angle xzy = \angle yzx$. If a node u is contained in this area, we write $u \in \angle xzy$.

This paper makes use of the following simple geometric facts. For two general points $a, b \in \mathbb{R}^2$, due to the triangle inequality, we have that $\|a + b\| \leq \|a\| + \|b\|$. Moreover, it holds that $\|a + b\| = \|a\| + \|b\| \Leftrightarrow \exists t \geq 0 : a = t \cdot b$. Pythagoras' law says that for any $a, b \in \mathbb{R}^2$ with $\langle a, b \rangle = 0$, it holds that $\|a+b\|^2 = \|a\|^2+\|b\|^2$. If we know two points on the border of a disk, then their midpoint must be on a specific straight line. Formally, let $u, v, x \in \mathbb{R}^2$. Then $\|u - x\| = \|v - x\|$ if and only if $x = \frac{1}{2}(u + v) + t(u - v)$ for some $t \in \mathbb{R}$. For the Euclidean norm, it holds for $C = C(u, v)$ for $u, v \in \mathbb{R}^2$ that $w \in C$ and $\|w - u\| \geq \|v - u\|$ imply $w = v$.

For some proofs we want to choose a disk $\tilde{C}$ contained in a bigger disk C with at least two points on the border of $\tilde{C}$. We can make the following observations.

Fact 2.1. *Let $C = B(x, r)$ be a disk with $u, v \in C$ and $u \neq v$. Then there is a disk $\tilde{C} = B(\tilde{x}, \tilde{r}) \subseteq C$ with $\|u - \tilde{x}\| = \|v - \tilde{x}\| = \tilde{r}$.*

For the opposite direction, given a set of points, we need a disk containing all of them, with at least three on the border.

Fact 2.2. *Let $V \subset \mathbb{R}^2$ be a finite set of points, not all of them collinear. Then there are three different, not collinear points $u, v, w \in V$ with $C(u, v, w) \supset V$.*

2.2 Delaunay Graphs

We consider graphs with an *embedding* into $\mathbb{R}^2$. Let $V \subset \mathbb{R}^2$ be a finite set and $E \subset \binom{V}{2}$, then $G = (V, E)$ is called *undirected embedded graph* with *nodes* V and *edges* E. Let $n = |V|$ be the cardinality of V. We define $N_G(u) = \{v \in V : \{u, v\} \in E\}$ as the *neighbors* of u. Moreover, let $\overline{N}_G(u) = N_G(u) \cup \{u\}$ denote the *neighbors of u including u*.

Usually we speak of a *directed* graph $G = (V, E)$ with $E \subset V^2$. Then a *directed edge* from u to v is denoted by (u, v), the *undirected edge* $\{u, v\}$ represents the two directed edges (u, v) and (v, u) and $N_G(u) = \{v \in V : (u, v) \in E\}$. $\overline{N}_G(u)$ is defined analogously. Note that any undirected graph can be seen as a directed graph with this interpretation of undirected edges. This will be done implicitly throughout the paper. A directed graph is called *strongly connected*, if for every pair (u, v) of nodes $u, v \in V$ there is a directed path from u to v. A direct graph is *weakly connected*, if the graph obtained by replacing all directed edges by undirected edges is connected.

Armed with these definitions, we can now define the Delaunay graph.

Definition 2.3 (Delaunay Graph)**.** *The* Delaunay Graph

$$G_D(V) = (V, E_D(V))$$

of the vertices V is an undirected embedded graph defined by $\{u, v\} \in E_D(V) \Leftrightarrow u \neq v \wedge \exists C = B(x, r) : C \cap V = \{u, v\}$ i.e., u and v are connected, if and only if there is a disk containing only these two points of V.

Recall that we will consider non-degenerate cases, that is, we assume there is no disk $B(x, r)$ with four different points $x_1, \ldots, x_4 \in V$ on its border, i.e. $\forall B(x, r) : |V \cap \{y \in \mathbb{R}^2 : \|x - y\| = r\}| \leq 3$. It is easy to see that the Delaunay graph on a given node set always includes the convex hull edges.

2.3 Properties

We can give several equivalent formulations of Definition 2.3 that will be useful in our analysis. In a Delaunay graph, two nodes u and v are connected if and only if either they are the only two nodes in the disk $C(u, v)$, or if there exists a third node w such that u, v, and w are the only three nodes in $C(u, v, w)$. [3]

Lemma 2.4. *Let $G = (V, E_D(V))$ be a Delaunay graph. Then*

$$\begin{aligned} \{u, v\} \in E_D(V) \quad \Leftrightarrow \quad & u \neq v \wedge (C(u, v) \cap V = \{u, v\} \vee \\ & \vee \exists w \in V \setminus \{u, v\} : C(u, v, w) \cap V = \{u, v, w\}) \end{aligned}$$

The following lemma states that in a Delaunay graph, for each pair of non-adjacent nodes, there must be a "close" neighboring node.

Lemma 2.5. *Let $G = (V, E_D(V))$ be a Delaunay graph and $\{u, v\} \notin E_D(V)$. Then every disk $C = B(x, r)$ containing u and v must contain at least one neighbor $w \in N_G(u)$ with $\|w - x\| < r$.*

Proof. Consider the family of disks $C_t = B(x + t(u - x), (1 - t)\|u - x\|)$, for $t \in [0, 1]$, i.e., with center between u and x and radius at most $\|u - x\|$. Obviously $u \in C_t$ for $t \in [0, 1]$, $C_0 = C$ and $C_1 = \{u\}$. Moreover the disk C_t is always a part of the disk C: $y \in C_t$ implies $\|y - x - t(u - x)\| \leq (1 - t)\|u - x\|$, so $\|y - x\| \leq \|y - x - t(u - x)\| + t\|u - x\| \leq \|u - x\| \leq r$ and thus $C_t \subseteq C$. Let $C_{\tilde{t}}$ be a specific disk for $\tilde{t} \in [0, 1]$ and $C_{\tilde{t}} \cap V \neq \{u\}$ (not only including u), which contains the minimal number of points from V.

Recall that since we do not consider degenerate cases, no more than three points lie on a circle. Thus, and due to the minimality of $C_{\tilde{t}} \cap V$, $C_{\tilde{t}} \cap V$ will contain one or two points besides u, which are all on the border of $C_{\tilde{t}}$.

From Definition 2.3, since $\{u, v\} \notin E_D(V)$, we immediately know that there is at least one point $w \in C_{\tilde{t}} \cap V$, $w \notin \{u, v\}$. The distance between w and x must be smaller than r: By Definition 2.3 and Lemma 2.4 (for three points on a circle), $\{u, w\} \in E_D(V)$ and so $w \in N_G(u)$. Due to the triangle inequality, $\|w - x\| \leq \|w - x_t\| + \|x_t - x\| \leq (1 - t)r + tr$ with equality only for $w = u$, where $x_t = x + t(u - x)$ is the center of C_t. Therefore $\|w - x\| < r$. □

We need some properties about restrictions of Delaunay graphs to a subset of nodes $U \subset V$. It is easy to see, that the restriction of the Delaunay graph of V to U is contained in the Delaunay graph on U:

Lemma 2.6.

$$U \subset V \Rightarrow E_D(U) \supset E_D(V) \cap U \times U$$

Proof. Let $\{u, v\}$ be an edge in $E_D(V) \cap U \times U$. Then by Definition 2.3 there is a disk $C = B(x, r)$ such that $C \cap V = \{u, v\}$. Since $U \subset V$, $C \cap U = \{u, v\}$ and thus $\{u, v\} \in E_D(U)$. □

Combining this lemma with the previous one, additional insights can be gained. Let us pick U such that is contains the neighbors $\overline{N}_G(u)$ of a node u. Then u has the same neighbors in the Delaunay graph on U as in the original Delaunay graph.

Lemma 2.7. *Let $G = (V, E_D(V))$ be a Delaunay graph, $u \in V$ and $\overline{N}_G(u) \subset U \subset V$. Then $\overline{N}_{G_D(U)}(u) = \overline{N}_G(u)$.*

Proof. $\overline{N}_{G_D(U)}(u) \supset \overline{N}_G(u)$ is clear by Lemma 2.6. Now let $\{u, v\} \in U \times U \setminus E_D(V)$. So, by Lemma 2.5, in each disc $C = B(x, r)$ containing v, w there is a neighbor w of u (i.e. $w \in \overline{N}_G(u) \subset U$). Thus, by Definition 2.3, $\{u, v\} \notin E_D(V)$. □

The next, important characterization of Delaunay graphs also argues about edges that are *not* Delaunay. If and only if two nodes u and v are not connected, there must exist two neighbors x and y of u, such that the disk $C(u, v, x)$ contains only y, and x and y lie on different sides of the line connecting u and v.

Lemma 2.8. *Let $G = (V, E_D(V))$ be a Delaunay graph. Then*

$$\{u,v\} \notin E_D(V) \quad \Leftrightarrow \quad \exists x, y \in V \setminus \{u,v\} : C(u,v,x) \cap V \supset \{u,v,x,y\} \wedge \\ \wedge \langle x-u, \perp(v-u)\rangle \cdot \langle y-u, \perp(v-u)\rangle \leq 0$$

That is, x and y must be on different sides of the line connecting u and v. One can even choose $x, y \in N_G(u)$.

Proof. Observe that the lemma claims an equivalence. We will study the two directions individually.

Direction "$\Rightarrow$": First we assume $\{u,v\} \in E_D(V)$, and prove that in this case, no such x and y exist. If $\{u,v\} \in E_D(V)$, by the definition of Delaunay graphs (Definition 2.3), a disk $B(c,r)$ with $B(c,r) \cap V = \{u,v\}$ exists. By Fact 2.1, w.l.o.g. $\|u-c\| = \|v-c\| = r$. For an arbitrary node x other than u and v, and consider the disk $C(u,v,x) \neq B(c,r)$ ($B(c,r)$ cannot contain x), the borders of $C(u,v,x)$ and $B(c,r)$ intersect in exactly two points, namely u and v. Thus $C(u,v,x) \setminus B(c,r)$ lies completely on one side of the line uv. Therefore, there is no such y as required lying in $C(u,v,x)$ but on the other side of uv than x. This yields the desired contradiction to the existence of such x and y nodes.

Direction "$\Leftarrow$": Now assume $\{u,v\} \notin E_D(V)$. Then by Lemma 2.5 for every disk $B(c,r) \ni u, v$ there is a point $w \in N_G(u) \cap B(c,r)$ (i.e. $w \neq v$).

Define $C_t = C(x_t, r_t)$ to be a disk with center on the perpendicular bisector of the line through u and v, i.e., with $x_t = \frac{1}{2}(u+v) + t \cdot \perp(u-v)$ and $r_t = \|x_t - u\|$. Then $u, v \in C_t$. Since there is only a finite number of points in V, consideration with regard to Definition 2.3 of C_T and C_{-T} for big enough $T > 0$ ensures $x, y \in N_G(u)$ on different sides of uv, i.e., with $\langle x-u, \perp(v-u)\rangle \cdot \langle y-u, \perp(v-u)\rangle < 0$.

Now we choose $U = N_G(u) \cup \{v\}$ in the sense of Lemma 2.7. With increasing parameter t the circle C_t will contain less of the area on the one and more of the area on the other side of uv. Let $\tilde{t}$ the maximal t such there is an x on the opposite side of x_t with respect to the line uv, i.e., $\tilde{t} = \max\{t \in \mathbb{R} : \exists x \in C_t \cap U : \langle x-u, \perp(v-u)\rangle \cdot \langle x_t - u, \perp(v-u)\rangle < 0\}$. Let x be as in the definition of $\tilde{t}$. We know $\|x - x_{\tilde{t}}\| = r_{\tilde{t}}$ (otherwise we could increase $\tilde{t}$) and thus $C_{\tilde{t}} = C(u,v,x)$.

If $C_{\tilde{t}} \cap U = \{u,v,x\}$, then $\{u,v\} \in E_D(U)$ by Lemma 2.4. But this cannot be true according to Lemma 2.7. So take $y \in C_{\tilde{t}} \cap (U \setminus \{u,v,x\})$. As we only consider non-degenerate cases, i.e. no four points are on the border of $C(u,v,x)$ and no other point is on the line uv, y must be on the opposite side of uv with respect to x (by maximality of $\tilde{t}$, i.e. $\langle x-u, \perp(v-u)\rangle \cdot \langle y-u, \perp(v-u)\rangle < 0$. Therefore, x, y fulfill the conditions of the statement. □

We will later need the existence of special edges in Delaunay graphs. First, we observe that a Delaunay node is always connected to the *closest* node, that is, the Delaunay graph contains the nearest neighbor graph. The following lemma follows directly from the observation that, for two closest neighbors $u, v \in V$, $C(u,v) \cap V = \{u,v\}$.

Lemma 2.9. *Let $G = (V, E_D(V))$ be a Delaunay graph and $u \in V$. Then u is connected to the node $v \in V \setminus \{u\}$ with minimal Euclidean distance to u.*

Another important property of Delaunay graphs is that they are connected.

Lemma 2.10. *Every Delaunay graph $G = (V, E_D(V))$ is connected. [19]*

Moreover, it can be shown that these graphs have a planar embedding.

Lemma 2.11. *Every Delaunay graph $G = (V, E_D(V))$ is planar. [3]*

2.4 Local Algorithms and Self-Stabilization

The main objective of this paper is to devise a distributed algorithm—essentially a simple set of rules—which is run by every node all the time. Independently from the initial, weakly connected topology (nodes can be connected to any other nodes from all over the metric space), a self-stabilizing algorithm is required to eventually terminate with a correct Delaunay graph as defined in Definition 2.3. During the execution of this algorithm, each node will add or remove edges to other nodes using *local interactions* only. In order to evaluate the algorithm's performance, a synchronous model is investigated (similarly to [17]) where time is divided into *rounds*. In a round, each node is allowed to perform an update of its neighborhood, that is, remove existing edges and connect to other nodes. We study the *time complexity* of the algorithm and measure the number of rounds (in the worst-case) until a Delaunay graph is formed and the algorithm stops.

3 Self-Stabilizing Algorithm

This section presents our algorithm *ALG*. During the execution of *ALG*, all nodes continuously calculate a Delaunay graph on their neighbors, that is, each node u computes the Delaunay graph on the node set $\overline{N}(u)$—a triangulation consisting of circular edges ("convex hull") and radial edges. In the following, we will call the considered node the *active* node and the calculated Delaunay graph its so-called *local Delaunay graph*. Here *active* is *not* referring to an calculation order but emphasizes the local role of the computing node for its local Delaunay graph. Note that the local Delaunay graph of a node u, denoted by $G_L(G, u) = (\overline{N}_G(u), E_D(\overline{N}_G(u))$, also contains edges that are not incident to u, but connect neighbors of u.

The construction of the local Delaunay graph $G_L(G, u)$ is reminiscent of the *1-localized Delaunay graph* $LDEL^{(1)}(\overline{N}_G(u))$ introduced by Li et al. [15]. The major difference is that [15] assumes an underlying unit disk graph to define the neighbors of a node whereas in our construction the current approximation of the Delaunay graph is used (which can be arbitrarily bad initially).

Informally, the active node keeps edges to neighbors in the local Delaunay graph, and forms edges among them in a circular order around it. All other nodes are deferred to some Delaunay neighbor of the active node. The *Delaunay update* $\tilde{G} = (V, \tilde{E})$ of G is the union of these update edges for all nodes in G.

Definition 3.1 (Stable and Temporary Edges)**.** Stable edges *are undirected and are currently—from a local point of view—consistent with the*

Delaunay properties. Temporary edges *on the other hand are directed and will appear, be forwarded, and disappear again (i.e., become stable) during the execution of our algorithm.*

We are now ready to formally define the Delaunay update:

Definition 3.2 (Delaunay Update)**.** *Let $G = (V, E)$ be a directed graph.*

- *The* local Delaunay graph *of u is $G_L(G, u) = (\overline{N}_G(u), E_D(\overline{N}_G(u))$.*
- *Each node u selects the following edges $E_S(G, u)$ from $E_D(\overline{N}_G(u))$, which will be kept for the next round:*

$$E_S(G,u) \quad = \quad E_{stable}(G,u) \cup E_{temp}(G,u)$$

where Rule I*:*

$$\begin{aligned} E_{stable} \quad = \quad & \{\{u,v\} : v \in N_{G_L(G,u)}(u)\} \\ & \textit{(undirected edges from } u \textit{ to its neighbors in } G_L(u)\textit{)} \\ \cup \quad & \big\{\{v,w\} : v,w \in N_{G_L(G,u)}(u) \wedge \\ & \quad \nexists x \in N_{G_L(G,u)}(u) : x \in \angle vuw\big\} \\ & \textit{(undirected circular edges between } u\textit{'s neighbors)} \end{aligned}$$

and Rule II*:*

$$\begin{aligned} E_{temp}(G,u) \quad = \quad & \big\{(v,w) : v \in N_{G_L(G,u)}(u), w \in N_G(u) \setminus \overline{N}_{G_L(G,u)}(u) \wedge \\ & \quad \forall x \in N_{G_L(G,u)}(u) : \|x - w\| \geq \|v - w\|\big\} \\ & \textit{(directed edges from } u\textit{'s non-neighbors to neighbors)} \end{aligned}$$

Rule II keeps directed edges between a node's neighbor and a non-neighbor if there is no closer neighbor to the non-neighbor (a nearest connection strategy).

- *Then the* Delaunay update *is $\tilde{G} = (V, \tilde{E})$ with*

$$\tilde{E} = \bigcup_{u \in V} E_S(G,u),$$

the graph that arises when all nodes have chosen their new neighbors for the next round.

Observe that *ALG* follows a nearest neighbor strategy in the sense that temporary circular edges are only allowed from closest neighbors to non-neighbors of the active node. Moreover, an important property of our algorithm is that temporary edges are forwarded to closer nodes. We will say the edge (u, v) is *passed* to node w, if (u, v) is replaced by (w, v); the node pointed to remains the same.

In summary, in algorithm *ALG*, each node $u \in V$ runs the following simple code:

```
while (true) {
    compute E_S(G,u) (cf Definition 3.2);
    propagate edge updates;
}
```

Figure 1 gives a simple example to acquaint ourselves with *ALG*.

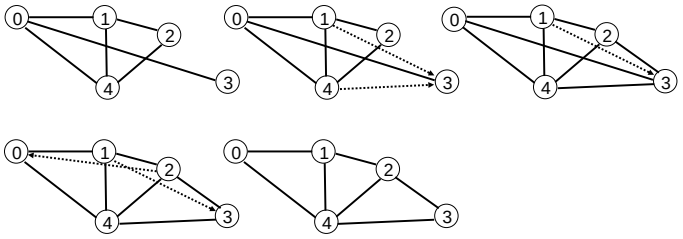

Figure 1: *ALG* sample execution (Delaunay updates from top left to bottom right). In the first Delaunay update, Nodes 1 and 4 connect temporarily to Node 3. Subsequently, Nodes 2 and 4 can connect to Node 3, rendering the stable edge between Node 0 and Node 3 superfluous. After four Delaunay updates, the nodes are triangulated.

4 Analysis

In this section, the following theorem is derived.

Theorem 4.1. *Let $G = (V, E)$ be a directed embedded, weakly connected graph. Then ALG requires at most $O(n^3)$ rounds (i.e. Delaunay updates) until the topology converges to the Delaunay graph $G_D(V)$.*

Our analysis is organized as follows. First we study basic properties of the Delaunay updates and show that Delaunay edges will not be removed in updates and become stable. Subsequently, we prove that starting from a supergraph, superfluous non-Delaunay edges will be removed in time $O(n)$. Finally, we put things together, and show that our algorithm has a unique fixpoint, where the "local Delaunay graph" equals the real (unique) Delaunay graph; from a potential function argument, the $O(n^3)$ convergence time follows.

4.1 Delaunay Updates

We start with two fundamental properties of the Delaunay updates.

Lemma 4.2. *Let $G = (V, E)$ be a directed embedded graph and $\tilde{G} = (V, \tilde{E})$ its Delaunay update. Then Delaunay edges of G will also be in $\tilde{G}$, that is,*

$$(u, v) \in E \cap E_D(V) \quad \Rightarrow \quad \{u, v\} \in \tilde{E}.$$

Proof. Since $(u, v) \in E$, $u, v \in \overline{N}_G(u)$. By Lemma 2.6, $\{u, v\} \in E_D(\overline{N}_G(u))$ and by Definition 3.2, Rule I, $\{u, v\} \in \tilde{E}$. □

Moreover, the following lemma claims that Delaunay updates maintain connectivity.

Lemma 4.3. *Let $G = (V, E)$ be a directed embedded graph and $\tilde{G} = (V, \tilde{E})$ its Delaunay update. If G is (weakly or strongly) connected, then so is $\tilde{G}$.*

Proof. It is enough to show, that for every neighbor w of u in G there is a directed path from u to w in $\tilde{G}$. By Definition 3.2, we have to consider two cases. If $w \in N_{G_L(G,u)}(u)$, then $(u,w) \in \tilde{E}$ is a path from u to w. Otherwise $(v,w) \in E$ for some $v \in N_{G_L(G,u)}(u)$, since directed edges are forwarded between nodes, while the pointed-to node remains the same. Thus (u,v) and (v,w) form a path from u to w. □

Note that Lemma 4.3 proves that all paths are maintained during updates.

4.2 Superfluous Edges

Lemma 4.2 implies that if every Delaunay edge will be created in some round, we end up with a supergraph of $G_D(V)$. Assuming that this happened, this section will show that all non-Delaunay edges will disappear after a few rounds, so that we are left with just the Delaunay graph.

First we need that the circular connections of a node's Delaunay neighbors are Delaunay edges.

Lemma 4.4. *Let $G = (V,E)$ be a directed embedded graph with $N_G(u) \supseteq N_{G_D(V)}(u)$. Then*

$$A := \left\{\{v,w\} : v,w \in N_{G_L(G,u)}(u) \wedge \nexists x \in N_{G_L(G,u)}(u) : x \in \angle vuw\right\} \subseteq E_D(V).$$

Proof. We prove by contradiction. Assume $\{v,w\} \in A \setminus E_D(V)$. Since u, v, w are not collinear, $C(u,v,w)$ exists. Then, by Lemma 2.5, $C(u,v,w)$ must contain a "close" neighbor $y \in N_{G_L(G,u)}(u)$. If $\langle y-u, \perp(v-u)\rangle \cdot \langle w-u, \perp(v-u)\rangle \leq 0$, we can use Lemma 2.8 to conclude that $\{u,v\} \notin E_D(V)$ (nodes w and y on different sides of uv). The local Delaunay graph $G_L(G,u)$ is a Delaunay graph on the nodes $U = N_G(u) \subset N_{G_D(V)}(u)$ by assumption. Lemma 2.7 and $v \in N_{G_L(G,u)}(u)$ yields the desired contradiction. Thus $\langle y-u, \perp(v-u)\rangle \cdot \langle w-u, \perp(v-u)\rangle > 0$. Analogously we can get $\langle y-u, \perp(w-u)\rangle \cdot \langle v-u, \perp(w-u)\rangle > 0$. So y, w lie on the same side of uv and y, v lie on the same side of uw. But this means $y \in \angle vuw$ and therefore $y \notin N_{G_L(G,u)}(u)$ which contradicts Lemma 2.5 and concludes the proof. □

The following helper lemma is crucial for our convergence analysis, as it shows that non-Delaunay edges become shorter over time. The lemma takes into account that *ALG* follows a nearest neighbor strategy.

Lemma 4.5. *Let $G = (V,E)$ be a directed embedded graph with $E \supseteq E_D(V)$ and $\tilde{G} = (V,\tilde{E})$ its Delaunay update. Then for every non-Delaunay edge in $\tilde{G}$ there is a strictly longer non-Delaunay edge in G, formally, $(v,w) \in \tilde{E} \setminus E_D(V) \Rightarrow \exists (u,w) \in E \setminus E_D(V) : \|u-w\| > \|v-w\|$.*

Proof. Let $(v,w) \in \tilde{E}$ be an edge in the updated graph. Then according to *ALG*, there are three possibilities that lead to this edge in $\tilde{E}$. Either v is a local neighbor of w (i.e. $v \in N_{G_L(G,w)}(w)$), or v and w are local neighbors of u with (v,w) in the local hull (i.e. $v,w \in N_{G_L(G,u)}(u) \wedge \nexists x \in N_{G_L(G,u)}(u) : x \in \angle vuw$), or w is no local neighbor of u and v is the local neighbor with smallest distance to w (i.e. $v \in N_{G_L(G,u)}(u), w \in$

$N_G(u) \setminus \overline{N}_{G_L(G,u)}(u) \wedge \forall y \in N_{G_L(G,u)}(u) : \|y - w\| \geq \|v - w\|$). We will consider the three cases in turn.

If $v \in N_{G_L(G,w)}(w)$: By Lemma 2.7 and since $E \supseteq E_D(V)$, w's local Delaunay neighbors are exactly its Delaunay neighbors ($N_{G_L(G,w)}(w) = N_{G_D(V)}(w)$), and thus $\{v, w\} \in E_D(V)$. This is a contradiction to our assumption that $(v, w) \in \tilde{E} \setminus E_D(V)$, and hence the claim holds trivially.

If $v, w \in N_{G_L(G,u)}(u) \wedge \nexists x \in N_{G_L(G,u)}(u) : x \in \angle vuw$: In this case the contradiction follows from Lemma 4.4, which tells us that $\{v, w\} \in E_D(V)$.

If $v \in N_{G_L(G,u)}(u), w \in N_G(u) \setminus \overline{N}_{G_L(G,u)}(u) \wedge \forall y \in N_{G_L(G,u)}(u) : \|y - w\| \geq \|v - w\|$: Given $w \in N_G(u) \setminus \overline{N}_{G_L(G,u)}(u)$ it remains to prove the existence of a point $v \in N_{G_L(G,u)}(u)$ with $\|v - w\| < \|u - w\|$. By Lemma 2.5, $C(u, w)$ contains a "close neighbor" $v \in N_{G_L(G,u)}(u)$ with $\|v - \frac{1}{2}(u+w)\| < \frac{1}{2}\|u - w\|$. Then $\|v - w\| \leq \|v - \frac{1}{2}(u+w)\| + \|\frac{1}{2}(u+w) - w\| < \|u - w\|$. □

We are now ready to prove that superfluous edges disappear quickly in at most n rounds.

Lemma 4.6. *Let $G = (V, E)$ be a directed embedded graph with $E \supseteq E_D(V)$, i.e., G is a supergraph of the Delaunay graph $G_D(V)$. Then ALG converges to $G_D(V)$ in at most n rounds.*

Proof. Consider the sequence of graphs $G_0 = G, G_1, \ldots$, where G_{i+1} is the Delaunay update of $G_i = (V, E_i)$. By Lemma 2.7, Delaunay neighbors are not removed in updates, i.e., $E_i \supseteq E_D(V)$ for all i. Moreover, according to Lemma 4.5, no new edges are added during an update operation, implying that $E_D(V)$ is stable.

Define l_i to be the maximal non-Delaunay edge distance in G_i, i.e., $l_i = \max\{\|u - v\| : (u, v) \in E_i \setminus E_D(V)\}$. Obviously if there are no non-Delaunay edges left, the graph is Delaunay, i.e., $l_i = -\infty \Leftrightarrow G_i = G_D(V)$. By Lemma 4.5, for each edge (v, w) in $E_i \setminus E_D(V)$ there is a strictly longer edge (u, w) in $E_{i-1} \setminus E_D(V)$. According to our algorithm ALG, for each directed non-Delaunay edge, the second pointed to node remains fixed and the other node gets closer (w.r.t. Euclidean distances) in each step. Since there are only $n-1$ nodes different from w and the nearest is always a Delaunay neighbor (cf. Lemma 2.9), $G_{|V|-1} = G_D(V)$. □

4.3 Fixpoint and Convergence

We will first show that there is no "dead end", i.e., as long as we do not reach the Delaunay graph, local updates will change the graph.

Lemma 4.7. *Let $V \subset \mathbb{R}^2$ be a finite set of nodes in general positions. Then the Delaunay graph $G = G_D(V) = (V, E_D(V))$ is the only weakly connected stable graph on the nodes V, i.e., the only graph that equals its Delaunay update $\tilde{G} = (V, \tilde{E})$.*

Proof. First recall the fact that the global Delaunay triangulation and hence $G_D(V)$ is unique. Moreover, recall from Lemma 4.6 (and its proof) that the edge set $E_D(V)$ is stable with respect to update operations.

We now need to show uniqueness of ALG's fixpoint. Remember that each node of the graph is associated with a point in the plane. We consider

the embedding of the graph in which all edges are replaced by undirected edges and represented by straight lines. We call a graph G *locally triangulated* if for each node $u \in V$ the induced subgraph on the node set $\overline{N}_G(u)$ is a triangulation. According to Definition 3.2, a stable graph with respect to ALG is locally triangulated.

We pursue the following strategy: We prove by induction that a locally triangulated graph is a *planar graph* with respect to the above embedding (i.e. no two edges cross). Thus fixpoints must be planar graphs. From this and connectedness, however, uniqueness follows due to the classic result (cf, e.g., Chapter 9.3 in the book [3] by Berg *et al.*) that from any triangulation, a sequence of edge flips leads to the Delaunay graph.

Therefore, it only remains to prove planarity. Assume for contradiction that a graph G with n nodes is locally triangulated but not planar.

For $n = 1, 2, 3$, this is obviously impossible. Thus consider $n = 4$ and call the nodes w_1, w_2, w_3, w_4. W.l.o.g., assume that the edges $\{w_1, w_3\}$ and $\{w_2, w_4\}$ cross (here we don't worry about the direction of the edges since they are replaced by lines). Since we assumed the graph G to be connected, there must be another edge. W.l.o.g., let this edge be $\{w_1, w_2\}$. Since G is locally triangulated, looking at node w_1 implies that the edge $\{w_2, w_3\}$ must belong to G. Now looking at w_2 gives a contradiction since the two crossing edges both belong to the induced subgraph on $\overline{N}_G(w_2) = \{w_1, w_2, w_3, w_4\}$. This induced subgraph is not triangulated and thus G is not locally triangulated.

For $n > 4$, we show that the existence of a connected locally triangulated graph on n nodes with crossing edges implies a connected locally triangulated graph on $n - 1$ nodes with crossing edges. Thus no such graph can exist.

Consider an arbitrary pair of crossing edges. Since $n > 4$ we can choose a node v not incident to any of these edges. We consider the graph G' obtained by removing v and all edges incident to v. Since G is locally triangulated and no three points are collinear, G' is still connected. Since we only worry about local triangulation, this property may be distorted only for neighbors of v. Those form due to the edge removal a polygon in G' (i.e., each neighbor of v has edges to exactly two other neighbors of v). Since any polygon can be triangulated (cf e.g., Chapter 3.1 in the book [3] by Berg *et al.*), we can add edges to G' such that the graph is locally triangulated. Since the pair of crossing edges remains untouched, this graph fulfills the induction hypothesis, which leads to the desired contradiction. □

For the convergence proof we need a potential function.

Definition 4.8 (Potential ϕ)**.** *Let $G = (V, E)$ be a directed embedded graph. Then the* potential $\phi_G(v)$ *of a node v is defined as the number of nodes $w \in V$ that are better approximations of the Delaunay neighbors than its current neighbors. This means they would be neighbors of v in the local Delaunay graph containing v, its neighbors and w. Formally*

$$\phi_G(v) = |\{w \in V \setminus \overline{N}_G(v) : \{v, w\} \in E_D(\overline{N}_G(v) \cup \{w\})\}|.$$

The potential of the whole graph is $\phi(G) = \sum_{v \in V} \phi_G(v)$.

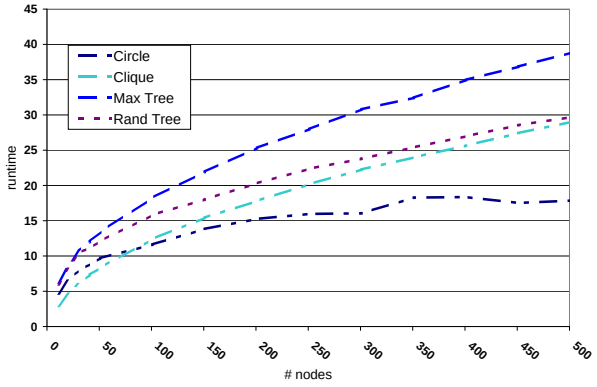

Figure 2: Convergence times (average over 100 runs) for different initial networks Circle, Clique, Max Tree, and Rand Tree.

We now observe that the potential $\phi(G)$ is *monotone*.

Lemma 4.9. *Let $G = (V, E)$ be a directed embedded graph and $\tilde{G} = (V, \tilde{E})$ its Delaunay update. Then $\phi(G) \geq \phi(\tilde{G})$.*

Proof. Consider a node $v \in V$ and define X_v and $\tilde{X}_v$ as

$$X_v := \left\{w \in V \setminus \overline{N}_G(v) : \{v, w\} \in E_D(\overline{N}_G(v) \cup \{w\})\right\}$$

$$\tilde{X}_v := \left\{w \in V \setminus \overline{N}_{\tilde{G}}(v) : \{v, w\} \in E_D(\overline{N}_{\tilde{G}}(v) \cup \{w\})\right\}$$

It suffices to show $X_v \supset \tilde{X}_v$ for all v. Therefore consider an arbitrary $w \in \tilde{X}_v$, i.e., $w \in V \setminus \overline{N}_{\tilde{G}}(v)$ and $\{v, w\} \in E_D(\overline{N}_{\tilde{G}}(v) \cup \{w\})$.

First we have to show $w \notin \overline{N}_G(v)$. For the sake of contradiction, assume $w \in \overline{N}_G(v)$. Since $\overline{N}_{\tilde{G}}(v)$ contains the local Delaunay neighbors $N_{G_L(G,v)}(v)$ by Rule I, the node set of the local Delaunay graph $\overline{N}_G(v)$ must contain witnesses in the sense of Lemma 2.8. These nodes are local Delaunay neighbors of v and therefore their connection to v persists in $\tilde{G}$. So there are $x, y \in \overline{N}_{G_L(G,v)}(v) \setminus \{v, w\} \subset \overline{N}_G(v) \cap \overline{N}_{\tilde{G}}(v)$ such that $C(v, w, x) \cap \overline{N}_G(v) \supset \{v, w, x, y\}$ and x, y are on opposite sides of the line vw. But, again by Lemma 2.8, $\{v, w\} \notin E_D(\overline{N}_{\tilde{G}}(v) \cup \{w\})$ which contradicts the assumption $w \in \tilde{X}_v$.

It remains to show $\{v, w\} \in E_D(\overline{N}_G(v) \cup \{w\})$. We prove by contradiction, again. Assume $\{v, w\} \notin E_D(\overline{N}_G(v) \cup \{w\})$. So, by Lemma 2.8, there are $x, y \in \overline{N}_{G_L(G,v)}(v) \setminus \{v, w\} \subset \overline{N}_G(v) \cap \overline{N}_{\tilde{G}}(v)$ such that $C(v, w, x) \cap \overline{N}_G(v) \supset \{v, w, x, y\}$ and x, y are on opposite sides of the line vw. Again Lemma 2.8, together with Lemma 2.6, implies that $\{v, w\} \notin E_D(\overline{N}_{\tilde{G}}(v) \cup \{w\})$ and leads to the desired contradiction. □

Combining all our insights, we can now prove our main result.

Theorem 4.10. *Let $G = (V, E)$ be a directed embedded, weakly connected graph. Then ALG requires at most $O(n^3)$ rounds (i.e. Delaunay updates) until the topology converges to the Delaunay graph $G_D(V)$.*

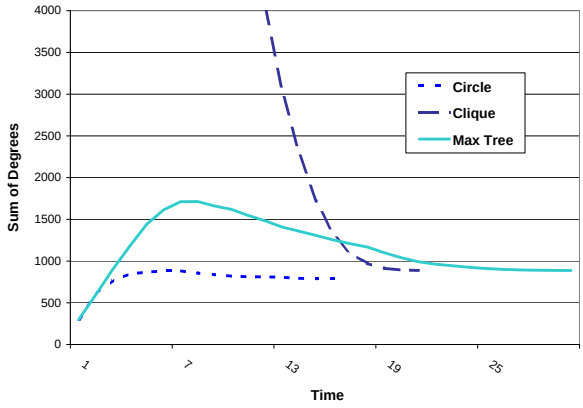

Figure 3: Evolution of the sum of the node degrees (network with 300 nodes).

Proof. Consider the sequence of graphs $G_0 = G$, G_1, ..., where G_{i+1} is the Delaunay update of $G_i = (V, E_i)$. Due to Lemma 4.3 each graph in this sequence is weakly connected. As soon as $E_D(V) \subseteq E_i$, we know $G_{i+n} = G_D(V)$ from Lemma 4.6. So we just have to consider the case $E_D(V) \not\subseteq E_i$.

From Lemma 4.9 we know that the potential cannot increase. In particular, it holds that once a node leaves the potential set

$$\{w \in V \setminus \overline{N}_G(v) : \{v, w\} \in E_D(\overline{N}_G(v) \cup \{w\})\},$$

it will never be member of the set again. Therefore, it remains to show that after every at most n steps, the cardinality of the set decreases (by a positive integer value): Since the potential is bounded by $n \cdot (n-1)$ and the only graph with potential 0 is the Delaunay graph, this gives the desired bound on the convergence time.

Now assume for the case of contradiction that the potential set has the same cardinality for more than n rounds. This implies that no new Delaunay edge appeared during this time period. Since each temporary edge is forwarded no more than $n-1$ times, the topology must describe a Delaunay fixpoint in the sense of Lemma 4.7. Since the graph is connected, it must be the Delaunay graph. This contradiction proves the claim. □

5 Simulations

In order to complement our formal analysis, we briefly report on some of the results obtained during our *in silico* experiments. Let us emphasize that the lessons in this section are preliminary, and a more extensive simulation study is left to the future.

We examined different initial topologies. In the Circle topology, nodes are arranged and connected in a circle-like fashion in the Euclidean plane; "close" nodes are therefore already linked. In the Clique topology,

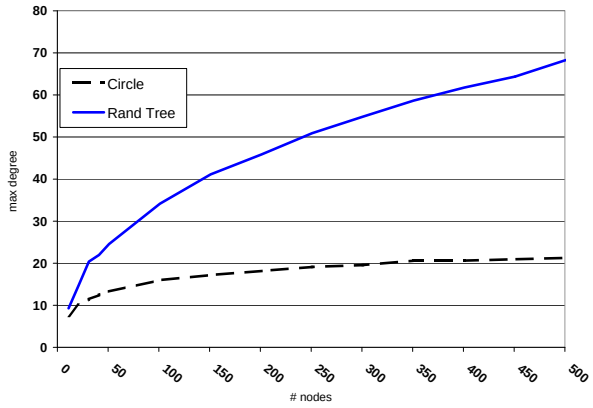

Figure 4: Maximal node degree for CIRCLE and RAND TREE networks.

nodes are distributed uniformly over the plane, and are completely connected to each other; in particular, CLIQUE contains the Delaunay graph as a subgraph. A particularly hard, non-local case is modeled with the topology MAX TREE: nodes are distributed uniformly at random in the plane, and are connected in a *maximum* spanning tree fashion. In other words, nodes are typically connected to far away nodes only. In contrast, in the RAND TREE topology, the randomly distributed nodes form a random tree. Although our algorithm also works for directed graphs, we only present simulation results for undirected graphs.

Figure 2 shows the resulting runtimes (in number of rounds). We first observe that for all topologies CIRCLE, CLIQUE, MAX TREE, and RAND TREE, the actual number of rounds is quite small. Indeed, we believe that our asymptotic analysis may be too pessimistic (or at least hides very small constants only), for any topology. Note also that the runtime of CIRCLE and CLIQUE is smaller than the runtimes of the trees. In case of CIRCLE, this can be explained by the high initial locality; the graph also already contains the convex hull. A good convergence time for CLIQUE corresponds to our formal analysis, where we proved a better performance if the initial topology is a super-graph of the Delaunay graph. Finally, it does not come as a surprise that the maximum spanning tree yields the worst results.

An interesting performance measure for any topological, self-stabilizing scheme is the node degree. Figure 3 depicts how the sum of the node degrees (incoming plus outgoing) evolves over time in a system with 300 nodes. As expected, in the CLIQUE, the degree declines sharply. Here, in order to improve presentation, we omitted the high initial degrees on the left; also recall that the execution on complete networks is faster, which explains the missing data points to the right. In the CIRCLE graph, the degree increases slightly in the beginning, but drops again soon and comes to a stable value. The maximal edge count observed during all 100 runs with 300 nodes was 927. MAX TREE yields a similar picture; how-

ever, the degree can become higher (maximum over all runs was 2024) and it takes more time to reach the equilibrium point. Apparently, here the non-locality entails a certain additional degree cost. Finally, let us remark that in none of our experiments, the degree reached values larger than twice the final number of Delaunay edges, unless the initial topology was already very dense—in which case the number of edges declined sharply. Figure 4 plots the *maximal* node degree (rather than the sum) for different networks. Generally, also here it can be observed that if the initial topology has a low degree and is sufficiently "local" already, there is typically no node with high degree.

We averaged each experiment over 100 runs, and found that while the runtimes for the trees are very stable, the CIRCLE topology exhibits quite a high variance ($\sigma^2 \approx 20$ for 300 nodes). We have experimented with an alternative Rule II for our Delaunay updates, which is not a nearest neighbor but a *circular connection strategy*. The selected temporary edges are

$$E_{temp}(G,u) = \{(v,w) : v \in N_{G_L(G,u)}(u), w \in V \setminus \overline{N}_{G_L(G,u)}(u) \wedge \forall x \in N_{G_L(G,u)}(u) : x \notin \angle vuw\}.$$

We conjecture that this strategy also converges to the Delaunay graph. While the runtime (and also the average degree) is typically slightly worse in our simulations, for certain star-shaped topologies, the variance can be smaller. We will not go into these details here, but would like to point out that—depending on the application—considering the circular variation of *ALG* may help.

6 Conclusion

The relatively young field of topological self-stabilization promises the advent of very robust network structures that recover from arbitrary changes or attacks. While already several solutions for graph linearization have been proposed, our work initiates the study of more complex, 2-dimensional stabilization mechanisms. Especially, we show how to construct Delaunay graphs, and also provide a convergence time guarantee. We believe that our construction can be useful in several settings, e.g., in social networks where participants want to organize in such a manner that participants with similar interests are connected. From this perspective, our algorithms can be regarded as *a topology control mechanism for wireline networks*.

In our future research, we plan to investigate whether our algorithm (and thus the convergence time and the degree during the execution) can be improved. Moreover, we will study whether, and if yes *how*, the insights gained for the 2-dimensional case can also be adopted for three or more dimensional distributions or graphs. Finally, we will analyze the effect of different *scalable scheduling regimes* (see [10]) where in each round, an independent set of operations is executed.

References

[1] J. Aspnes and Y. Wu. $O(\log n)$-time overlay network construction from graphs with out-degree 1. In *Proc. OPODIS*, 2007.

[2] B. Awerbuch and G. Varghese. Distributed program checking: A paradigm for building self-stabilizing distributed protocols. In *Proc. FOCS*, pages 258–267, 1991.

[3] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, 2008.

[4] P. Bose and P. Morin. Online routing in triangulations. In *Proc. ISAAC*, pages 113–122, 1999.

[5] T. Clouser, M. Nesterenko, and C. Scheideler. Tiara: A self-stabilizing deterministic skip list. In *Proc. SSS*, 2008.

[6] B. N. Delaunay. Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, 7(6):793–800, 1934.

[7] E. Dijkstra. Self-stabilization in spite of distributed control. *Communications of the ACM*, 17:643–644, 1974.

[8] S. Dolev and T. Herman. Superstabilizing protocols for dynamic distributed systems. *Chicago Journal of Theoretical Computer Science*, 4:1–40, 1997.

[9] E. M. Gafni and D. P. Bertsekas. Asymptotic optimality of shortest path routing algorithms. *IEEE Trans. Inf. Theor.*, 33(1):83–90, 1987.

[10] D. Gall, R. Jacob, A. Richa, C. Scheideler, S. Schmid, and H. Täubig. Brief announcement: On the time complexity of distributed topological self-stabilization. In *Proc. SSS*, 2009.

[11] J. E. Goodman and J. O'Rourke, editors. *Handbook of Discrete and Computational Geometry*. CRC Press, Inc., 1997.

[12] T. Hou and V. Li. Transmission range control in multihop packet radio networks. *IEEE Trans. Comm.*, 1986.

[13] L. Hu. Topology control for multihop packet radio networks. *IEEE Trans. Comm.*, 1993.

[14] R. Jacob, A. Richa, C. Scheideler, S. Schmid, and H. Täubig. A distributed polylogarithmic time algorithm for self-stabilizing skip graphs. In *Proc. PODC*, 2009.

[15] X.-Y. Li, G. Calinescu, and P.-J. Wan. Distributed construction of planar spanner and routing for ad hoc wireless networks. In *Proc. INFOCOM*, 2002.

[16] X.-Y. Li, G. Calinescu, P.-J. Wan, and Y. Wang. Localized delaunay triangulation with application in ad hoc wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(10):1035–1047, 2003.

[17] M. Onus, A. Richa, and C. Scheideler. Linearization: Locally self-stabilizing sorting in graphs. In *Proc. ALENEX*, 2007.

[18] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. Technical Report MIT-LCS-TR-819, MIT, 2001.

[19] I. Stojmenovic. *Handbook of Wireless Networks and Mobile Computing*. Wiley, 2002.

[20] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Trans. Comm.*, 1984.

[21] Y. Wang and X.-Y. Li. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. In *Proc. DIALM-POMC*, 2003.