A Self-stabilizing and Local Delaunay Graph Construction*

Riko Jacob[1], Stephan Ritscher[1], Christian Scheideler[2], and 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.

Recently, progress was made in the area 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.

* Research supported by the DFG project SCHE 1592/1-1. Due to space constraints, many proofs and simulation results are only presented in the technical report [9].

Y. Dong, D.-Z. Du, and O. Ibarra (Eds.): ISAAC 2009, LNCS 5878, pp. 771–780, 2009.
© Springer-Verlag Berlin Heidelberg 2009

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 [4], 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 [1], 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 [6] 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 [5]. 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).* A notable recent exception is [8] 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., [3, 7]). 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.

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.

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 $0 \neq \perp x \in \mathbb{R}^2$ 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, $\|a+b\| = \|a\| + \|b\| \Leftrightarrow \exists t \geq 0 : a = t \cdot b$. 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 $C(u, v, w)$. [2]

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

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

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 $\|w - x\| < r$.*

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, $\{u, v\} \notin E_D(U)$. □

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*

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

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

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. [12]*

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. [2]*

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 [11]) 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. [10]. The major difference is that [10] 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. Due to the division into rounds, the updates are well-defined and the actions of different nodes in the same round do not interfere.

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:*

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

where Rule I*:*

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

and Rule II*:*

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

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.

4 Analysis

We start with two fundamental properties of the Delaunay updates.

Lemma 4.1. *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 4.2. *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 4.2 proves that all paths are maintained during updates.

4.1 Superfluous Edges

Lemma 4.1 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 4.3. *Let $G = (V, E)$ be a directed embedded graph with $N_G(u) \supseteq N_{G_D(V)}(u)$. Then*

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

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 4.4. *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, $(v, w) \in \tilde{E} \setminus E_D(V) \Rightarrow \exists (u, w) \in E \setminus E_D(V) : \|u - w\| > \|v - w\|$.*

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

Lemma 4.5. *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.*

4.2 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 4.6. *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 $\tilde{G} = (V, \tilde{E})$.*

For the convergence proof we need a potential function.

Definition 4.7 (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 $\phi(G) = \sum_{v \in V} \phi_G(v)$.

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

Lemma 4.8. *Let $G = (V, E)$ be a directed embedded graph and $\tilde{G} = (V, \tilde{E})$ its Delaunay update. Then $\phi(G) \geq \phi(\tilde{G})$.*

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

Theorem 4.9. *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 $G_D(V)$.*

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 4.2 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 4.5. So we just have to consider the case $E_D(V) \not\subseteq E_i$.

From Lemma 4.8 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 4.6. Since the graph is connected, it must be the Delaunay graph. This contradiction proves the claim. □

References

[1] Awerbuch, B., Varghese, G.: Distributed program checking: A paradigm for building self-stabilizing distributed protocols. In: Proc. FOCS, pp. 258–267 (1991)
[2] Berg, M.d., Cheong, O., Kreveld, M.v., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer-Verlag TELOS, Heidelberg (2008)
[3] Clouser, T., Nesterenko, M., Scheideler, C.: Tiara: A self-stabilizing deterministic skip list. In: Proc. SSS (2008)
[4] Dijkstra, E.: Self-stabilization in spite of distributed control. Communications of the ACM 17, 643–644 (1974)
[5] Dolev, S., Herman, T.: Superstabilizing protocols for dynamic distributed systems. Chicago Journal of Theoretical Computer Science 4, 1–40 (1997)
[6] Gafni, E.M., Bertsekas, D.P.: Asymptotic optimality of shortest path routing algorithms. IEEE Trans. Inf. Theor. 33(1), 83–90 (1987)
[7] Gall, D., Jacob, R., Richa, A., Scheideler, C., Schmid, S., Täubig, H.: Brief announcement: On the time complexity of distributed topological self-stabilization. In: Proc. SSS (2009)
[8] Jacob, R., Richa, A., Scheideler, C., Schmid, S., Täubig, H.: A distributed polylogarithmic time algorithm for self-stabilizing skip graphs. In: Proc. PODC (2009)
[9] Jacob, R., Ritscher, S., Scheideler, C., Schmid, S.: A self-stabilizing and local delaunay graph construction. Tech. Report TR-TI-09-307, University of Paderborn (2009)
[10] Li, X.-Y., Calinescu, G., Wan, P.-J.: Distributed construction of planar spanner and routing for ad hoc wireless networks. In: Proc. INFOCOM (2002)
[11] Onus, M., Richa, A., Scheideler, C.: Linearization: Locally self-stabilizing sorting in graphs. In: Proc. ALENEX (2007)
[12] Stojmenovic, I.: Handbook of Wireless Networks and Mobile Computing. Wiley, Chichester (2002)