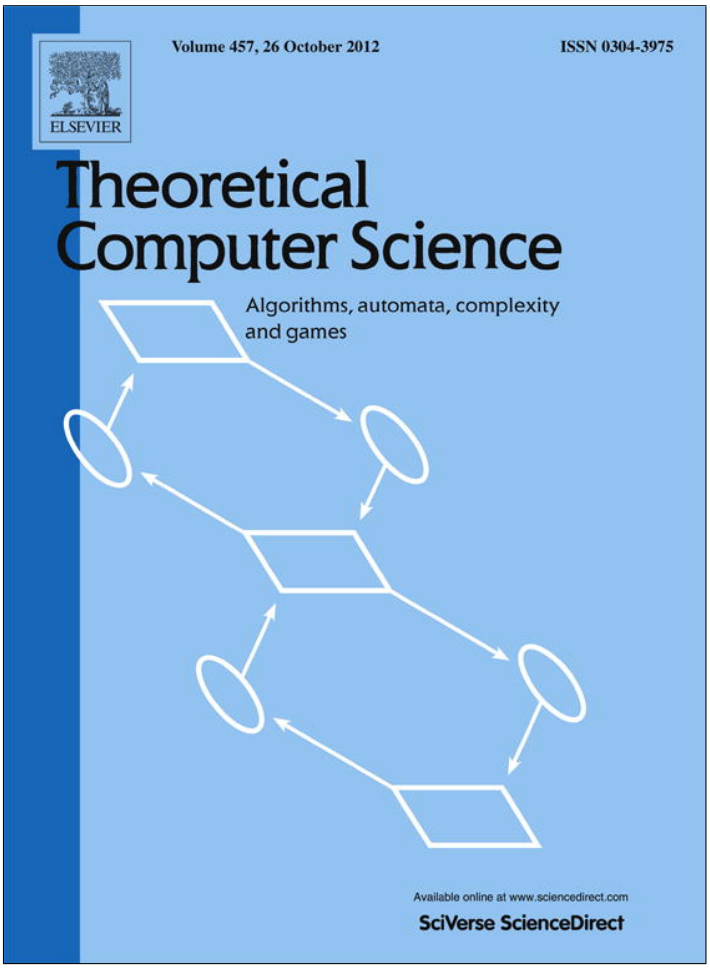

Theoretical Computer Science 457 (2012) 137–148

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Towards higher-dimensional topological self-stabilization: A distributed algorithm for Delaunay graphs

Riko Jacob [a], Stephan Ritscher [b], Christian Scheideler [c], Stefan Schmid [d,*]

[a] *ETH Zurich, Zurich, Switzerland*
[b] *Institut für Informatik, TU München, Garching, Germany*
[c] *Department of Computer Science, University of Paderborn, Paderborn, Germany*
[d] *Deutsche Telekom Laboratories (T-Labs) & TU Berlin, Berlin, Germany*

ARTICLE INFO

Article history:
Received 16 September 2011
Received in revised form 19 April 2012
Accepted 12 July 2012
Communicated by D. Peleg

Keywords:
Distributed algorithms
Topology control
Social networks

ABSTRACT

This article 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 explore a natural and relevant 2-dimensional generalization. In particular, we present a local self-stabilizing algorithm DSTAB which is based on the concept of "local Delaunay graphs" and which forwards temporary edges in greedy fashion reminiscent of compass routing. DSTAB constructs a *Delaunay graph* from any initial connected topology and in a distributed manner in time $O(n^3)$ in the worst-case; if the initial network contains the Delaunay graph, the convergence time is only $O(n)$ rounds. DSTAB also ensures that individual node joins and leaves affect a small part of the network only. Such self-stabilizing Delaunay networks have interesting applications and our construction gives 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 can be 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 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, only little is known about the design of self-stabilizing algorithms. In particular, while existing literature often focuses on *eventual* stabilization, the required *convergence times* are less explored.

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 article, we go an important step further and explore the *2-dimensional* case. We assume that the nodes are distributed in the Euclidean plane and are connected arbitrarily. A natural 2-dimensional analog of the chain network is the *Delaunay graph*, whose edge set includes all nearest neighbor connections between node pairs.

* Corresponding author. Tel.: +49 175 9309875.
E-mail addresses: rjacob@inf.ethz.ch (R. Jacob), ritsches@in.tum.de (S. Ritscher), scheideler@upb.de (C. Scheideler), schmiste@tik.ee.ethz.ch, stefan@net.t-labs.tu-berlin.de (S. Schmid).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.07.029

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 and 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 Dijkstra in 1974 [9], 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: for example, 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 [11] 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 [10].

Unfortunately, many topology maintenance algorithms today work only from certain degenerate network states, and are hence not fully self-stabilizing (see, e.g., the technical report of the Chord network [25]). Notable recent exceptions are SKIP+ [17] and *Re-Chord* [19] (published after [18]) which describe local self-stabilizing algorithms for hypercubic graphs (see also [4]). Unfortunately, however, these 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., [7,12,22]) that can serve as a basis constructing for more complex topologies such as de Bruijn or skip graphs [24]. Our article 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 have been studied for almost a century now [8] and there exists a large body of literature. For an introduction and basic algorithms, see, e.g., the handbook by Goodman and O'Rourke [14]. The recent interest in wireless and ad-hoc networks has brought the Delaunay graph (and locally computable graphs in general) back to the limelight (see, e.g., [1,5,13]). Led by the energy challenges – nodes in ad-hoc networks often have a limited power supply – researchers have proposed numerous approaches for *topology control* [15,27]. In [16], 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 [6], 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, e.g., graphs describing the intersection of the Delaunay triangulation and the Unit Disk Graph [20]. 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 [21,28]. In contrast to the wireless constructions, in this article, 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 seen as a *topology control mechanism for wireline systems*.

1.2. Our contributions

This article presents the first self-stabilizing algorithm (called DStab) 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. It is based on the notion of *local Delaunay graphs*: each node computes the Delaunay graph of its current neighbors, and forwards non-Delaunay edges according to a greedy routing algorithm (with respect to distance). Besides correctness, we are able to derive a $O(n^3)$ worst-case bound on the convergence time (i.e., number of communication rounds); if the initial network contains the Delaunay graph, the convergence time is only $O(n)$ rounds. Moreover, individual joins and leaves affect only a small part of the network. 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.

Compared to the trivial (not self-stabilizing) 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 (note that such a distinction contradicts the self-stabilization property): 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 join or leave, 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. Organization

The remainder of this article 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 article 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 article, 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 article 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 $x = \frac{1}{2}(u + v) + t\perp(u - v)$ 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 $E \subseteq \binom{V}{2}$, then $G = (V, E)$ is called an *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 $E \subset V \times V$. 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 article. 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.

In this rest of this paper, we will often speak about *faces* and *triangulations*.

Definition 2.4 (*Face*)**.** A *face* is a connected subset of the plane in the embedding which is constrained by edges (arcs).

Definition 2.5 (*Triangulation*)**.** We say that a planar embedded graph is *triangulated* if each finite *face* forms a triangle.

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

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

The following important lemma states that in a Delaunay graph, for each pair of non-adjacent nodes, there must be a "close" neighboring node. Note that in the lemma, u and v are symmetric, implying that *both* nodes must have such a neighbor. In other words, for both nodes a close neighbor must exist; this will be useful to prove convergence of our algorithm.

Lemma 2.7. *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 2.6 (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 $U \subseteq V$. 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 2.8.

$$U \subseteq V \Rightarrow E_D(U) \supset E_D(V) \cap (U \times U).$$

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 $U \subseteq V$, $C \cap U = \{u, v\}$ 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 it 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 2.9. *Let $G = (V, E_D(V))$ be a Delaunay graph, $u \in V$ and $\overline{N}_G(u) \subseteq U \subseteq V$. 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 2.8. Now let $\{u, v\} \in (U \times U) \setminus E_D(V)$. So, by Lemma 2.7, in each disk $C = B(x, r)$ containing u and v, there is a neighbor w of u (i.e. $w \in \overline{N}_G(u) \subseteq U$). 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 y, and x and y lie on different sides of the line connecting u and v.

Lemma 2.10. *Let $G = (V, E_D(V))$ be a Delaunay graph. Then*

$$\begin{aligned}\{u, v\} \notin E_D(V) \Leftrightarrow\ & \exists x, y \in V \setminus \{u, v\} : \\ & C(u, v, x) \cap V \supseteq \{u, v, x, y\} \wedge \\ & \langle x - u, \perp(v - u)\rangle \cdot \langle y - u, \perp(v - u)\rangle < 0.\end{aligned}$$

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 "⇐": 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, 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 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 "⇒": Now assume $\{u, v\} \notin E_D(V)$. Then by Lemma 2.7 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 sufficiently large $T > 0$ ensures $x, y \in N_G(u)$ on different sides of uv, i.e., it holds $\langle x - u, \perp(v - u)\rangle \cdot \langle y - u, \perp(v - u)\rangle < 0$.

Now we choose $U = N_G(u) \cup \{v\}$ in the sense of Lemma 2.9. 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 $\tilde{t}$ be 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 2.6. But this cannot be true according to Lemma 2.9. So there exists a $y \in C_{\tilde{t}} \cap (U \setminus \{u, v, x\})$. 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 and 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 2.11. *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 2.12. *Every Delaunay graph $G = (V, E_D(V))$ is connected [26].*

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

Lemma 2.13. *Every Delaunay graph $G = (V, E_D(V))$ is planar [3].*

2.4. Local algorithms and self-stabilization

The main objective of this article 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 [23]) 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 DStab. During the execution of DStab, 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 a 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.[1]

Our construction uses directed edges only, but we will sometimes write $\{u, v\}$ to refer to a situation where u has an edge to v ($(u, v) \in E$) and vice versa ($(v, u) \in E$), i.e., $\{u, v\}$ is a bidirectional or anti-parallel edge. Semantically, a directed edge (u, v) means "u sees v", i.e., u can send messages to v. A node u is responsible for all its outgoing edges, but u can also request that, e.g., its neighbors establish edges between themselves.

[1] 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. [20]. The major difference is that [20] 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 computes the local Delaunay graph on its (outgoing) neighbors (Rule I): it requests/keeps (bidirected) edges to neighbors in the local Delaunay graph, and forms (bidirected) edges among them in a circular order around it. We will refer to these edges as *stable* for this node; each edge which is stable for at least one node is called stable. In addition (Rule II), a node forwards directed edges which are not part of the local Delaunay graph to the neighbor which is closest to the edge's destination (so-called *temporary edges*); thus, the forwarding of temporary edges is essentially a greedy "distance compass routing" process [6]. Multiple parallel edges are merged.

The *Delaunay update* $\tilde{G} = (V, \tilde{E})$ of G is the union of these update edges for all active 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 currently – from a local point of view of at least one node – consistent with the Delaunay properties. *Temporary edges* on the other hand will appear, be forwarded, and disappear again (i.e., become stable, or are duplicates and are merged) during the execution of our algorithm. Note that an edge can be both stable and temporary, depending on the view point.

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:

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

where *Rule I*:

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

and *Rule II*:

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

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.

An important property of our algorithm is that temporary edges are forwarded to closer nodes ("nearest neighbor strategy"). 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. To show convergence of our algorithm, we will prove that non-Delaunay edges must originate from Rule II, and due to the Delaunay updates, they will become shorter over time. In the following, we will refer to a temporary edge that emerged from a stable edge as a *new* temporary edge; a "new" edge that is passed on is not considered a new temporary edge.

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

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

Fig. 1 gives a simple example to acquaint ourselves with DStab.

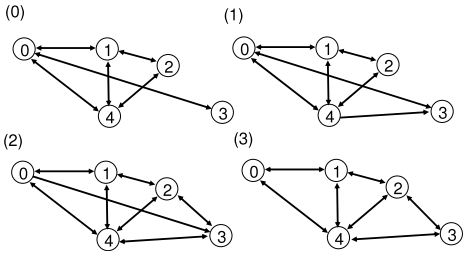

Fig. 1. Execution of DSTAB (Delaunay updates from top left to bottom right): in the first Delaunay update, Node 0 passes (0, 3) to Node 4 (the edge becomes (4, 3)), however (0, 3) is reinserted by Node 3 as Node 1 is the only neighbor in the local Delaunay graph. In the second update, Node 0 passes on the same edge again, and Node 4 inserts its local Delaunay edges concerning the newly discovered neighbor (Node 3). After Node 0 forwards the edge again, the graph is 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* DSTAB *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 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.

At the end of the section, we discuss dynamic scenarios and show that DSTAB can be extended to support efficient node joins and leaves.

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,*

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

Proof. Since $(u, v) \in E$, $u, v \in \overline{N}_G(u)$. By Lemma 2.8, $\{u, v\} \in E_D(\overline{N}_G(u))$ 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 show that the Rule I connections of a node whose neighbor set is a superset of its Delaunay neighbors, are Delaunay edges: the set of triangles (radial and circular edges) in the local Delaunay graph (of the supergraph) involving a node u is equivalent to the set of triangles (radial and circular edges) involving node u in the global Delaunay graph G_D.

Lemma 4.4. *Let $G = (V, E)$ be a directed embedded graph containing a superset of a node u's neighbors in the Delaunay graph, i.e., with $N_G(u) \supseteq N_{G_D(V)}(u)$. Moreover, let $E_L(G, u)$ be the set of edges in the local Delaunay graph $G_L(G, u)$ of G (i.e., the edges of the triangles involving node u), and let $E_D(G, u)$ be the set of edges of the triangles involving u in the global Delaunay graph of G. Then*

$$E_L(G, u) = E_D(G, u).$$

Proof. Observe that from the definition of the Delaunay graph on G, the presence of an edge $\{v, w\}$ in G_D is due to the absence of nodes in the corresponding disks for this edge, and clearly, a disk remains empty if a subset of the nodes is considered only; thus the Delaunay edges of G are maintained in subgraphs: according to Lemma 2.9, the Delaunay graph over a subset U of the nodes which includes the Delaunay neighbors of u does not change the neighborhood of a node u, that is, $N_G(u) \supseteq N_{G_D(V)}(u)$ implies that $N_{G_L(G,u)}(u) = N_{G_D(V)}(u)$. From the fact that both the Delaunay graph of a supergraph and the Delaunay graph of a subgraph are planar and triangulated, it follows that the corresponding global Delaunay graphs are equivalent. □

The following helper lemma is crucial for our convergence analysis, as it shows that non-Delaunay edges (which must originate from Rule II) become shorter over time due to the existence of the close neighbors (Lemma 2.9). That is, the lemma takes into account that DStab 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 DStab, 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 2.9 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 2.7, $C(u, w)$ 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* DStab *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 2.9, 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 length 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 DStab, for each directed non-Delaunay edge, the node that is pointed to 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 2.11), $G_{|V|-1} = G_D(V)$. □

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 DStab'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 DStab 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., Theorem 9.8 in the introductory book [3] by Berg et al.) that from any triangulation, the planar edge flips of DStab lead 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 do not 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 triangulations, 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 [3], 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 $\phi(G) = \sum_{v \in V} \phi_G(v)$.

A graph with an edge set that forms a superset of the Delaunay graph has a potential 0. Moreover, we 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 $X_v \supseteq \tilde{X}_v$ 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 2.10. These nodes are local Delaunay neighbors of v and therefore their connection to v persists in $\tilde{G}$. So there are $x, y \in \overline{N}_{G_L(G,v)}(v) \setminus \{v, w\} \subseteq \overline{N}_G(v) \cap \overline{N}_{\tilde{G}}(v)$ such that $C(v, w, x) \cap \overline{N}_G(v) \supseteq \{v, w, x, y\}$ and x, y are on opposite sides of the line vw. But, again by Lemma 2.10, $\{v, w\} \notin E_D(\overline{N}_{\tilde{G}}(v) \cup \{w\})$ 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 2.10, there are $x, y \in \overline{N}_{G_L(G,v)}(v) \setminus \{v, w\} \subseteq \overline{N}_G(v) \cap \overline{N}_{\tilde{G}}(v)$ such that $C(v, w, x) \cap \overline{N}_G(v) \supseteq \{v, w, x, y\}$ and x, y are on opposite sides of the line vw. Again Lemma 2.10, together with Lemma 2.8, 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* DStab *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 $G_0 = G, G_1, \ldots$, 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 $E_D(V) \nsubseteq E_i$.

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 a member of the set again. Therefore, it remains to show that after every at most $O(n)$ 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 graphs with

potential 0 are graphs with an edge set that forms a superset of 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 a period of $n+1$ rounds except for the first round. This implies that no new temporary edge appeared and there are no temporary edges left at the end of this time period: by definition, any new temporary edge must emerge from a stable edge (Rule II); however this means that the potential must have been reduced in the round before the temporary edge was created. (Note that a temporary edge can vanish without changing the potential by merging with an existing stable edge.) Moreover, a temporary edge can be passed on at most $n-1$ times. On the other hand, stable edges can only be created in the first round of the period, as a constant potential implies that no new neighbors are introduced to the nodes. Thus, 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. □

4.4. Joins and leaves

Besides efficient stabilization of the overall topology, it is desirable that individual nodes can join and leave "locally", i.e., with relatively low overhead in terms of time and message complexity. In order to extend our algorithm to a dynamic setting where nodes can join, we pursue the following strategy: no change is required for leaves, and a node u can leave the network without notice which invalidates all incident edges of u. When a node u joins, it contacts an arbitrary existing node v in the network (directed edge (u, v)), and requests the antiparallel edge (v, u). The corresponding edges are treated (i.e., forwarded) as in our self-stabilizing algorithm, and u repeatedly re-requests such edges until some node includes u in its local Delaunay graph.

Let δ be the size of the Delaunay cell of a joining node u (i.e., the number of nodes in the polygon if u is removed) and let λ be the time complexity of nearest neighbor routing (i.e., of the temporary edge forwarding) [6]. Then, a node join requires $O(\delta+\lambda)$ time: any edge can be established in time $O(\delta)$ while temporary edges are forwarded at most $O(\lambda)$ many hops. Similarly, for the case where a node u leaves, let δ denote the number of nodes in the (not necessarily convex) polygon without u. The new stable Delaunay edges for the cell are found in time $O(\delta)$, as in the worst case, edges enter the polygon one by one; however, although the Delaunay edges appear quickly, temporary edges may be forwarded through the graph and take $O(n)$ hops until they disappear. Thus, we have the following corollary. As it takes at most $O(\delta)$ rounds to span the Delaunay edges, and each edge can trigger at most $O(\delta)$ connections, a leave entails changes to at most $O(\delta^3)$ edges.

Corollary 4.11. *A node join requires $O(\delta+\lambda)$ time, where δ is the number of nodes of the Delaunay polygon without the joining node, and λ is the routing diameter. A node leave requires $O(\delta)$ time until the new Delaunay edges are computed and $O(n)$ time until the temporary edges disappear where n is the network size; at most $O(\delta^3)$ edges are affected by the leave.*

5. Simulations

In order to complement our formal analysis, we report on some simulation results. 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, 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.

Fig. 2 (*top*) 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; observe however that our results indicate that at larger scale Rand Tree may be relatively faster than Clique. In the 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. Fig. 2 (*middle*) 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; however, 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

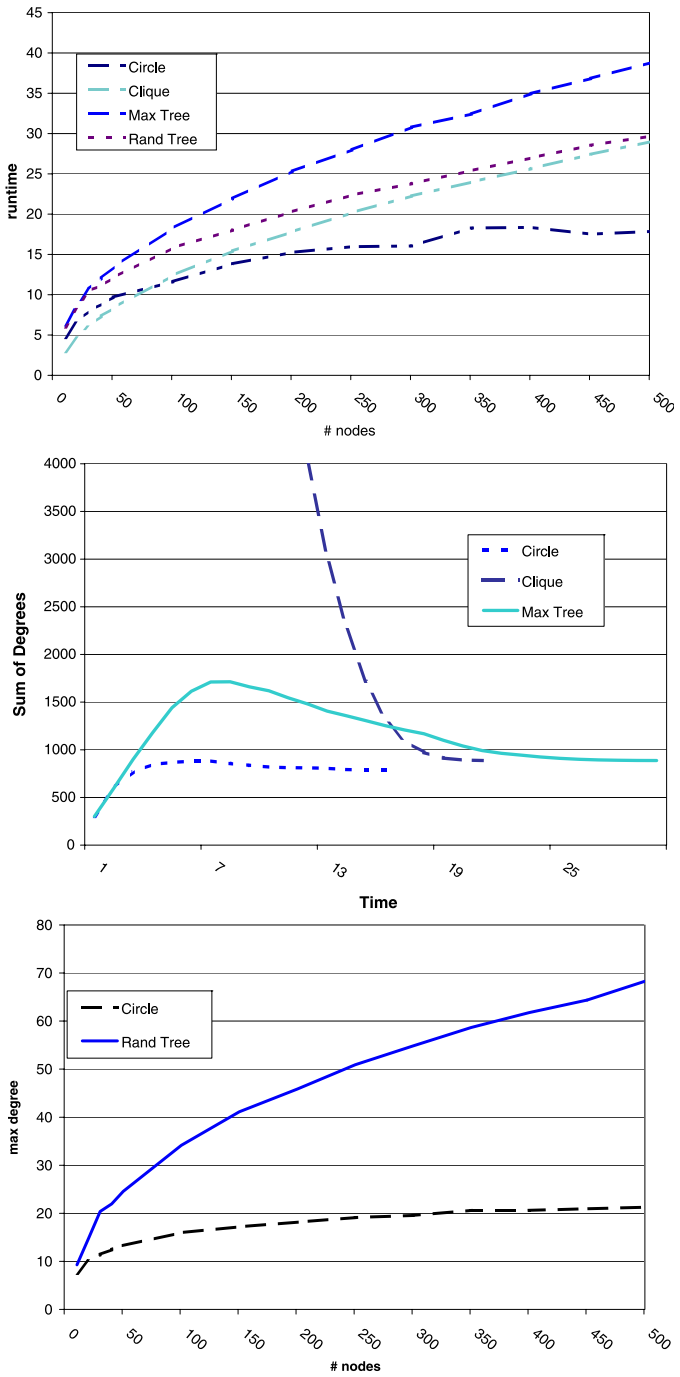

Fig. 2. *Top*: Convergence times (average over 100 runs) for different initial networks CIRCLE, CLIQUE, MAX TREE, and RAND TREE. *Middle:* Evolution of the sum of the node degrees (network with 300 nodes). *Bottom:* Maximal node degree for CIRCLE and RAND TREE networks.

which case the number of edges declined sharply. Fig. 2 (*bottom*) 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

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

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 DStab 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*.

Further research is needed to understand the achievable time complexity for self-stabilizing Delaunay graphs: currently, the best lower bound we can provide is $\Omega(n)$ and follows from linearization (see [23]). Also, while we believe that the step from 1-dimensional to 2-dimensional "linearization" is a critical one as it introduces the geometrical dimension to the problem, and that many of our techniques can be used also for higher dimensions, the generality of our approach remains an open question. Finally, it would also be interesting to analyze the effect of different *scalable scheduling regimes* (see [12]) where in each round, only an independent set of operations can be executed.

Acknowledgments

A preliminary version of this article without proofs has been published at the 20th International Symposium on Algorithms and Computation (ISAAC), 2009 [18]. This research is supported by the DFG project SCHE 1592/1-1.

References

[1] F. Araújo, L. Rodrigues, Fast localized delaunay triangulation, in: Proc. 8th international conference on Principles of Distributed Systems, OPODIS, 2005, pp. 81–93.
[2] B. Awerbuch, G. Varghese, Distributed program checking: a paradigm for building self-stabilizing distributed protocols, in: Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science, FOCS, 1991, pp. 258–267.
[3] M.d. Berg, O. Cheong, M.v. Kreveld, M. Overmars, Computational Geometry: Algorithms and Applications, 3rd edition, Springer-Verlag TELOS, 2008.
[4] A. Berns, S. Ghosh, S. Pemmaraju, Building self-stabilizing overlay networks with the transitive closure framework, in: Proc. 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS, 2011, pp. 62–76.
[5] P. Bose, P. Carmi, M. Smid, D. Xu, Communication-efficient construction of the plane localized delaunay graph, in: Procl 9th Latin American Conference on Theoretical Informatics, LATIN, 2010, pp. 282–293.
[6] P. Bose, P. Morin, Online routing in triangulations, SIAM Journal on Computing 33 (2004) 937–951.
[7] T. Clouser, M. Nesterenko, C. Scheideler, Tiara: a self-stabilizing deterministic skip list, in: Proc. 10th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems, SSS, 2008, pp. 124–140.
[8] B.N. Delaunay, Sur la sphère vide, Bulletin of Academy of Sciences of the USSR 7 (1934) 793–800.
[9] E. Dijkstra, Self-stabilization in spite of distributed control, Communications of the ACM 17 (1974) 643–644.
[10] S. Dolev, T. Herman, Superstabilizing protocols for dynamic distributed systems, Chicago Journal of Theoretical Computer Science 4 (1997) 1–40.
[11] E. Gafni, D.P. Bertsekas, Distributed algorithms for generating loop-free routes in networks with frequently changing topology, IEEE Transactions on Communications 29 (1981).
[12] D. Gall, R. Jacob, A. Richa, C. Scheideler, S. Schmid, H. Täubig, Time complexity of distributed topological self-stabilization: the case of graph linearization, in: Proc. 9th Latin American Theoretical Informatics Symposium, LATIN, 2010.
[13] J. Gao, L.J. Guibas, J. Hershberger, L. Zhang, A. Zhu, Geometric spanner for routing in mobile networks, in: Proc. 2nd ACM international symposium on Mobile Ad hoc Networking and Computing, MobiHoc, 2001, pp. 45–55.
[14] J.E. Goodman, J. O'Rourke (Eds.), Handbook of Discrete and Computational Geometry, CRC Press, Inc., 1997.
[15] T. Hou, V. Li, Transmission range control in multihop packet radio networks, IEEE Transactions on Communications (1986) 38–44.
[16] L. Hu, Topology control for multihop packet radio networks, IEEE Transactions on Communications (1993) 1474–1481.
[17] R. Jacob, A. Richa, C. Scheideler, S. Schmid, H. Täubig, A distributed polylogarithmic time algorithm for self-stabilizing skip graphs, in: Proc. ACM Symp. on Principles of Distributed Computing, PODC, 2009, pp. 131–140.
[18] R. Jacob, S. Ritscher, C. Scheideler, S. Schmid, A self-stabilizing and local Delaunay graph construction, in: Proc. 20th International Symposium on Algorithms and Computation, ISAAC, 2009, pp. 771–780.
[19] S. Kniesburges, A. Koutsopoulos, C. Scheideler, Re-chord: a self-stabilizing chord overlay network, in: Proc. 23rd ACM Symposium on Parallelism in Algorithms and Architectures, SPAA, 2011, pp. 235–244.
[20] X.-Y. Li, G. Calinescu, P.-J. Wan, Distributed construction of planar spanner and routing for ad hoc wireless networks, in: Proc. IEEE INFOCOM, 2002, pp. 1268–1277.
[21] X.-Y. Li, G. Calinescu, P.-J. Wan, Y. Wang, Localized Delaunay triangulation with application in ad hoc wireless networks, IEEE Transactions on Parallel and Distributed Systems 14 (2003) 1035–1047.
[22] R. Nor, M. Nesterenko, C. Scheideler, Corona: a stabilizing deterministic message-passing skip list, in: Proc. 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS, 2011, pp. 356–370.
[23] M. Onus, A. Richa, C. Scheideler, Linearization: locally self-stabilizing sorting in graphs, in: Proc. 9th Workshop on Algorithm Engineering and Experiments, ALENEX, 2007, pp. 229–237.
[24] P. Stevens, A. Richa, C. Scheideler, Self-stabilizing de Bruijn networks, in: Proc. 13th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS, 2011, pp. 416–430.
[25] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup service for internet applications, Technical Report MIT-LCS-TR-819 MIT, 2001.
[26] I. Stojmenovic, Handbook of Wireless Networks and Mobile Computing, Wiley, 2002.
[27] H. Takagi, L. Kleinrock, Optimal transmission ranges for randomly distributed packet radio terminals, IEEE Transactions on Communications (1984) 246–257.
[28] Y. Wang, X.-Y. Li, Localized construction of bounded degree and planar spanner for wireless ad hoc networks, in: Proc. 2003 Joint Workshop on Foundations of Mobile Computing, DIALM-POMC, 2003, pp. 59–68.