Near-optimal knowledge-free resilient leader election

(Article begins on next page)

01 February 2025

Near-Optimal Knowledge-Free Resilient Leader Election [*]

Yuanqiu Mo [a], Giorgio Audrito [b], Soura Dasgupta [c,**], Jacob Beal [d]

[a] *Southeast University, Nanjing 211189 China*

[b] *University of Turin, Turin (TO) 10149 Italy*

[c] *University of Iowa, Iowa City (IA) 52242 USA*

[d] *Raytheon BBN Technologies Cambridge (MA) 02138 USA*

Abstract

Leader election, is a fundamental coordination problem in distributed systems. It has been addressed in many ways for different systems. Among these approaches, resilient leader election algorithms are of particular interest due to the ongoing emergence of open, complex distributed systems such as smart cities and the Internet of Things. However, previous algorithms attaining the optimal scaling of O(diameter) stabilization time complexity either assume some prior knowledge of the network or else that very large messages can be sent. In this paper, we present a resilient leader election algorithm with O(diameter) stabilization time, small messages, and no prior knowledge of the network. This algorithm is based on aggregate computing, which provides a layered approach to algorithm development based on composition of resilient algorithmic "building blocks." With our algorithm, a key design function $g(\cdot)$ defines important performance attributes: a fast-growing $g(\cdot)$ will delay discarding of obsolete data, while a slow-growing $g(\cdot)$ will slow down convergence to a single leader. We prove that the best asymptotic behavior for $g(x)$ is $(1 + \sqrt{2})x + o(x)$, guaranteeing a near-optimal time complexity of $(2 + 2\sqrt{2})$ diameter + o(diameter) rounds for stabilization.

Key words: leader election; multiagent system; resilience; aggregate computing; nonlinear feedback control; global stability.

1 Introduction

There has been much interest in the control and stability of multiagent systems involving consensus, (Olfati-Saber *et al.* 2007), gossip, (Yu *et al.* 2017), distributed agreement, (Cao *et al.* 2008) and formation control, (Baillieul & Suri 2003), (Dasgupta *et al.* 2011), (Summers *et al.* 2009), (Fidan *et al.* 2013) and (Summers *et al.* 2011). A common feature is that all are distributed graph algorithms with local control action using only information flow between neighbors. Most are also nonlinear systems. This paper considers another such, presenting and an-

alyzing a leader election algorithm involving feedback composition of two nonlinear systems, each itself a distributed graph algorithm.

Leader election is a fundamental problem in distributed systems, in which a network of nodes collectively select a single leader in a distributed and (in our case) resilient fashion. Resilience ensures election of a *single* leader and recovery from transient perturbations such as the disappearance of leaders, temporary emergence of false leaders, and link failures that do not disconnect the graph.

Leader election studies have considered time, space, and message complexity of *deterministic* leader election in general networks with identifiers, in *probabilistic* election in anonymous networks, and in networks like ring, complete, and asynchronous graphs. We consider the problem in the context of open, complex distributed systems such as smart cities and the Internet of Things, which can have large numbers of devices and intermittent perturbation of both network membership and topology. We require the knowledge of neither N, the number of nodes, nor the diameter D of the network. All we assume is that each node carries a unique identifier, a require-

[**]Corresponding author: Soura Dasgupta.

Email addresses: moyuanqiu@gmail.com (Yuanqiu Mo), giorgio.audrito@unito.it (Giorgio Audrito), soura-dasgupta@uiowa.edu (Soura Dasgupta), jakebeal@ieee.org (Jacob Beal).

ment common to many networking algorithms[1], and exchanges messages with its neighbors in synchronous rounds. Simulations show that performance remains the same even under fair asynchronous execution.

In general synchronous networks, the global lower bound on the time required for a leader election algorithm is $\Omega(D)$ rounds[2] (Kutten *et al.* 2015). An algorithm meeting this bound is (Peleg 1990), with time and message complexities of $O(D)$ and $O(D\mathcal{E})$ respectively, $\mathcal{E}$ being the number of edges. Closely competitive is (Casteigts *et al.* 2016), which proposes a solution that elects a leader in $O(D+\log N)$. Both terminate on completion, as do most leader election algorithms, thus do not permit perturbations and are unsuitable for use in large open distributed systems, since nodes (including leaders) are continually joining and departing such networks.

Some solutions in optimal $O(D)$ time consider perturbations but assume prior knowledge of N or D. One in (Awerbuch *et al.* 1993) uses IDs and a *known* upper bound on D as does (Burman & Kutten 2007). Such knowledge is inconsistent with node loss or network growth common in open networks. Algorithms accommodating perturbations without network knowledge include (Aggarwal & Kutten 1993), where false leaders are eliminated with messages that may increase to unbounded length; and (Kravchik & Kutten 2013), which utilizes a synchronizer requiring knowledge of an upper bound on N; (Datta *et al.* 2011a, Datta *et al.* 2011b) and (Altisen *et al.* 2017) assume a distributed unfair daemon (Dubois & Guerraoui 2013) but stabilize in a far from optimal $O(N)$ as opposed to $O(D)$ rounds.

We adopt an *aggregate computing* (Beal *et al.* 2015) approach that offers a layered abstraction to simplifying the design, creation and maintenance of complex open distributed systems. Key for this paper is the middle layer, comprising three general "building-block" operators whose compositions have been shown in (Beal & Viroli 2014) and (Viroli *et al.* 2018) to realize a broad class of dispersed services. The two relevant here are: G-blocks that spread information through a network of devices and C-blocks that summarize salient information about the network to be used by interacting units.

We enunciate a globally uniformly asymptotically stable (GUAS) algorithm that uses a feedback interconnection of G and C blocks. In the remainder of this paper, we call it GCF abbreviating *"G and C blocks with Feedback"*. This algorithm is time optimal with a stabilization time of $O(D)$ with a message complexity of $O(D\mathcal{E})$. It uses

the Adaptive Bellman-Ford (ABF) algorithm (Mo *et al.* 2019) to estimate each node's distance estimate from its current leader and uses these in turn to estimate what we call a pseudo-diameter (which is like a diameter). This pseudo-diameter estimate is used to circumvent the *need for any prior network knowledge like the diameter in GCF's execution.*

The algorithm is resilient in the sense that it recovers from intermittent loss of nodes and leaders without knowledge of the network size. To achieve such recovery using network flooding would require strategies like time-to-live which in turn require diameter estimates. Too large a diameter estimate, will severely slow convergence. As experimentally demonstrated in (Pianini *et al.* 2016), too small an estimate will cause the behavior to collapse. On the other hand, the use of packet tracking, causes the required storage to grow without bound over time.

In contrast, the mechanism of resilience in this paper is as follows. Using the pseudo-diameter estimate $D_i(t)$ it carries, a leader i, computes a radius of influence, $R_i(t) = g(D_i(t))$, where $g(\cdot)$, is a free design function. The leader i then broadcasts this $R_i(t)$ to its followers. Each node j scouts its neighbors to see if any has a leader with priority higher than j's priority and carries a radius of influence greater than this neighbor's distance estimate. If none such can be found, then j appoints itself a leader. If such neighbors are found, then j looks at their leaders and chooses the leader with the highest priority as its own. As explained in Section 3, it is this device that permits the network to recover from the loss of a leader.

Like most feedback systems, the function $g(\cdot)$ determines important performance attributes. A high $g(\cdot)$ speeds convergence but impairs resilience by delaying recovery from the loss of a leader, while a low $g(\cdot)$ improves resilience but slows convergence. This trade-off is optimized for the best asymptotic behavior with $g(x) = (1 + \sqrt{2})x$ and a recovery time of $2(1 + \sqrt{2})D$.

Section 2 gives preliminaries, Section 3 the algorithm, and Section 4 the analytical framework. Section 5 proves resilience by upper bounding the time to recovery from transient perturbations, and proves GUAS and K-competitive performance with the optimal (Definition 6). Section 6 gives simulations with asynchronous rounds, singular and persistent perturbations; and shows that GCF compares favorably to the most directly comparable prior algorithm, from (Datta *et al.* 2011b), improving the transient behavior and being able to better withstand persistent perturbations. Section 7 summarizes and concludes. A preliminary version without proofs and the optimization of $g(\cdot)$ appears in (Mo *et al.* 2020).

---

[1] Note that while this implies identifiers must contain at least $\log_2 N$ bits, this requirement is trivially satisfied by the lower levels of the network stack in any modern real-world network.

[2] X is $\Omega(D)$ if there exist $k_1 > 0$ and $D_0 > 0$ such that $X \geq k_1 D$ for $D \geq D_0$.

2 Preliminaries

Consider an undirected graph $\mathcal{G} = (V, E)$, with $V = \{1, 2, ..., N\}$ the set of nodes (devices) and E the edge set. We assume that the index of a node represents its ID and also reflects its priority. A node with a lower ID has a higher priority, i.e. i has a higher priority than $i + 1$. The highest priority node, $i = 1$ (unless it is lost) must be elected as the sole leader. An edge represents a communication link between nodes, and i is a neighbor of j if there is an edge between them; $\mathcal{N}(i)$ is the set of neighbors of node i. Edge lengths are 1, i.e. distances are hop counts. The shortest distance d_i of i from 1, the desired leader, obeys the following recursion from (Mo *et al.* 2019):

$$d_i = \begin{cases} 0, & i = 1 \\ \min_{j \in \mathcal{N}(i)} \{d_j\} + 1, & i \neq 1 \end{cases} \quad (1)$$

These can be estimated using ABF, (Mo *et al.* 2019) which updates $\hat{d}_i(t)$, estimate of d_i by replacing each d_k in (1) by $\hat{d}_k(t)$:

$$\hat{d}_i(t+1) = \begin{cases} 0, & i = 1 \\ \min_{j \in \mathcal{N}(i)} \{\hat{d}_j(t)\} + 1, & i \neq 1 \end{cases} \quad (2)$$

Based on (1), we introduce two related definitions:
Definition 1. *A j minimizing the right hand side of (1) is a* true constraining node *of i, 1 being its own.*
Definition 2. *The pseudo-diameter $\mathcal{D}$ is*
$$\mathcal{D} = \max\{d_i \mid i \in V\}.$$

$\mathcal{D}$ may be smaller than the diameter of $\mathcal{G}$. Each i has a true constraining node: If $i = 1$, it is it's own; else there is a j that minimizes the second bullet in (1)

3 Algorithm

The block diagram of the leader election algorithm is in Figure 1. It uses a feedback interconnection comprising two aggregate computing building blocks, G and C, (Viroli *et al.* 2018), each of which is itself a distributed algorithm.Being a feedback composition of G and C it is called GCF. We define a function $g(\cdot)$ that is a design parameter.
Definition 3. *The function $g : \mathbb{N} \to \mathbb{N}$ with $\mathbb{N}$ the set of natural numbers is progressive, i.e., $g(x) > x$, and monotonic, i.e., $x \leq y \Rightarrow g(x) \leq g(y)$, design function.*

As part of our convergence analysis we will determine how one should select $g(\cdot)$. Observe $x + 1$ is a valid example of $g(x)$. As $g(\cdot) : \mathbb{N} \to \mathbb{N}$, every $g(x)$ obeys

$$g(x) \geq x + 1, \ \forall \ x \in \mathbb{N}. \quad (3)$$

Figure 1 shows the structure of the algorithm, with z^{-1} a unit delay. We assume each node in $V = \{1, \cdots, N\}$ has a unique ID. *The goal is to elect the highest priority node 1 as the single leader.*
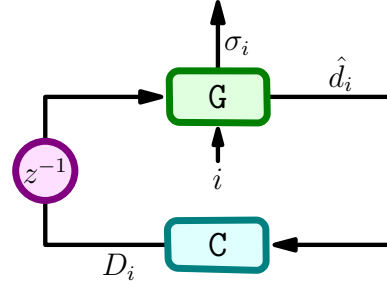

Fig. 1. Block diagram of GCF. G elects leaders, broadcasts radii of influence $R_i(t)$ and computes distance estimate $\hat{d}_i(t)$ of i from its leader and feeds these to C. C produces pseudo-diameter estimates $D_i(t)$, fed back to G in the following round to compute a leader's radius of influence. Nodes within $R_i(t)$ of a leader become its followers.

3.1 Algorithm details

A pseudo-code is in Algorithm 1. *Note every state is updated at every time step with no stopping criteria.*

The inputs: A connected undirected graph $\mathcal{G}$ and a unique ID i for node i. *Diameter is not needed.*
The output: The leader of each node: $\sigma_i(t)$ is i's leader i.e., *i follows $\sigma_i(t)$.* If $\sigma_i(t) = i$, then i is a leader.
State variables: $\sigma_i(t)$ and the following:

- $\hat{d}_i(t)$: The distance (hop count) estimate of i from its leader. It is computed similarly to (2), although the j used to update has more constraints.
- $R_i(t)$: The radius of influence. A leader computes this using its pseudo-diameter estimate, $D_i(t)$ as $R_i(t) = g(D_i(t))$, broadcasts this to its followers.
- $D_i(t)$: A *pseudo-diameter* estimate collected by the C-block and transmitted back to leaders:

$$D_i(t+1) = \max \big\{\hat{d}_i(t+1), \{D_j(t) \mid j \in \mathcal{N}(i) \wedge \\ \hat{d}_j(t) > \hat{d}_i(t+1)\}\big\} \quad (4)$$

The G-block updates the remaining variables using,

$$(\sigma_i(t+1), \hat{d}_i(t+1), -R_i(t+1)) = \min \big\{(i, 0, -g(D_i(t))), \\ \{(\sigma_j(t), \hat{d}_j(t) + 1, -R_j(t)) : \ j \in \mathcal{N}(i) \wedge \hat{d}_j(t) < R_j(t)\}\big\} \quad (5)$$

where ordering between triples is lexicographic. Thus in $\min\{(a_1, b_1, h_1), (a_2, b_2, h_2)\} = (a_i, b_i, h_i)$ where i is chosen to minimize a_i. In case of a tie the one among these with the smallest b_i is chosen. With a further tie, the one with smallest h_i is picked. The presence of $-R_i(t)$ rather than $R_i(t)$ signifies the fact that we are maximizing $R_i(t)$, i.e. minimizing $-R_i(t)$. (See (C) below). *The compact algorithm is thus (4) and (5).* The sequel explains its significance.

When does i elect itself a leader i.e. $\sigma_i(t + 1) = i$?

(i) It scouts its neighbor's states. *(One hop information)*

(ii) Checks if there is a neighbor j that is following a leader having a higher priority than i, i.e. $\sigma_j(t) < i$.

(iii) Checks if such a neighbor's distance estimate is less than its radius of influence, i.e. $\hat{d}_j(t) < R_j(t)$.

(iv) If (ii) is false then by virtue of the first variable in the tuples in (5), the lexicographic minimization ensures that the first tuple wins out. If (iii) is false then the set over which the second tuple in (5) is chosen is empty and the first wins by default. Thus i becomes a leader, as it sets its state as:

$$\sigma_i(t+1) = i, \quad \hat{d}_i(t+1) = 0, \quad R_i(t+1) = g(D_i(t)) \quad (6)$$

A neighbor of i that satisfies both (ii) and (iii) is called an *eligible neighbor*. The set of eligible neighbors is $\mathcal{E}_i(t+1)$ (see line 5 of the pseudo-code in Algorithm 1). *A node j cannot be eligible if $\hat{d}_j(t) \geq R_j(t)$.* A node becomes a leader if it *does not have an eligible neighbor.*

What about nodes with eligible neighbors?

(a) In this case (iii) ensures that $\{\mathcal{N}(i) \wedge \hat{d}_j(t) < R_j(t)\} \neq \varnothing$. Further (ii) ensures that $\sigma_j(t) < i$ for some $j \in \{\mathcal{N}(i) \wedge \hat{d}_j(t) < R_j(t)\}$. Thus, the lexicographically smallest of the second tuples in (5) wins. With j the index of this winning tuple,

$$\sigma_i(t+1) = \sigma_j(t), \hat{d}_i(t+1) = \hat{d}_j(t)+1, R_i(t+1) = R_j(t). \quad (7)$$

The j chosen above is called i's *constraining node.* It is chosen from the set of eligible neighbors

$$\mathcal{E}_i(t+1) = \{j \in \mathcal{N}(i) \mid \hat{d}_j(t) < R_j(t) \wedge \sigma_j(t) < i\}. \quad (8)$$

Specifically, a constraining node is any node in

$$\mathcal{E}_i^R(t+1) = \{\arg \min_{j \in \mathcal{E}_i(t+1)} \{(\sigma_j(t), \hat{d}_j(t)+1, -R_j(t))\}\}$$

More explanation is in (A)-(D).

(b) Through (7), i thus inherits its constraining node's leader and radius of influence. (One hop away.)

(c) It sets its hop count estimate from its new leader to one more than its constraining node's estimate.

Thus, the R_i chosen by leaders and *broadcast to followers moving from one constraining node to the next (see (7)).*

How does node i select its constraining node $j = c_i(t+1)$ whose leader it inherits?

Note that a leader is its own constraining node. For others the it must be in $\mathcal{E}_i^R(t+1)$, i.e. must satisfy:

(A) It must be an eligible neighbor, that *follows the highest priority leader* among all nodes in $\mathcal{E}_i(t+1)$ (first variables in the second tuples in (5)).

(B) In case of ties it must have the *smallest distance estimate* (second variable).

(C) In case of further ties it must have the *largest radius of influence* (third variable).

(D) For further ties, $c_i(t+1)$ has the smallest ID.

Note (D) just helps settle on a unique number if $\mathcal{E}_i^R(t+1)$ has multiple members. *We reiterate that a j for which $\hat{d}_j(t) \geq R_j(t)$ cannot be eligible and be a constraining*

node. Even with resolution using (D) the state in (7) is unique. While the C-block *collects* pseudo-diameter estimates by implementing (4), the G-block in (5), performs the remaining operations. The set of leaders at t is $S(t) = \{i \mid \sigma_i(t) = i\}$. Then with $\mathcal{E}_i^R(t+1)$ defined in (a) the constraining node is:

$$c_i(t) = \begin{cases} i & \sigma_i(t) = i \\ \min \mathcal{E}_i^R(t) & \sigma_i(t) \neq i \end{cases}. \quad (9)$$

Of course (5) remains valid without (9), as it defines the values associated with j and not j itself. With the correct distance of i from 1, and $\mathcal{D}$ the true pseudo-diameter (see (1) and Definition 2), the desired stationary point is

$$(\sigma_i(t), \hat{d}_i(t), R_i(t)) \equiv (1, d_i, g(\mathcal{D})), \forall i \in V \text{ and } D_1(t) = \mathcal{D}. \quad (10)$$

Note in (10) every node has 1 as leader and radius of influence $g(\mathcal{D})$. At steady state $D_1 = \mathcal{D}$. However, as explained later, not all steady state D_i will equal $\mathcal{D}$. Henceforth we will call a node i, converged at t_1 if $\sigma_i(t) = 1$ and $\hat{d}_i(t) = d_i$, for all $t \geq t_1$.

3.2 Insights and convergence mechanism

Note in the lexicographic minimization $i = \sigma_j(t)$ still resolves for the first tuple in (5) as 0, the second component of $(i, 0, g(D_i(t)))$, is less than $\hat{d}_j(t) + 1$, due to $\hat{d}_j(0) \geq 0$. The role of (4) is straightforward: i chooses its pseudo-diameter estimate as the maximum of its own new distance estimate and the pseudo-diameter estimate at nodes previously estimated as further from their leader than i. It follows that

$$D_i(t) \geq \hat{d}_i(t). \quad (11)$$

The inequality will be strict iff the neighbor's distance estimate at t exceeds i's current estimate.

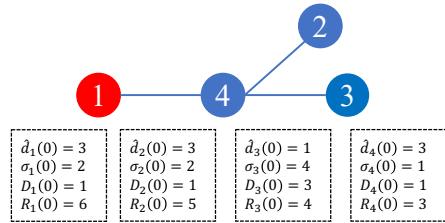| $\hat{d}_1(0) = 3$	$\hat{d}_2(0) = 3$	$\hat{d}_3(0) = 1$	$\hat{d}_4(0) = 3$
$\sigma_1(0) = 2$	$\sigma_2(0) = 2$	$\sigma_3(0) = 4$	$\sigma_4(0) = 1$
$D_1(0) = 1$	$D_2(0) = 1$	$D_3(0) = 3$	$D_4(0) = 1$
$R_1(0) = 6$	$R_2(0) = 5$	$R_3(0) = 4$	$R_4(0) = 3$

Fig. 2. An example graph.

To illustrate the convergence mechanism consider Figure 2 with $g(x) = 2x + 1$. Observe, 2 is the only current leader ($\sigma_2(0) = 2$), 1 should eventually be the sole leader. Node 1 is following 2, ($\sigma_1(0) = 2$) while 3 follows 4 and 4 follows 1. Node 1 has only one neighbor 4 which follows 1. Thus in the lexicographic minimization with $i = 1$, $(i, 0, -g(D_i(0))) = (1, 0, -3)$ wins out over $(\sigma_4(0), \hat{d}_4(0) + 1, -R_4(0)) = (1, 4, -3)$ and $(\sigma_1(1), \hat{d}_1(1), -R_1(1)) = (1, 0, -3)$. In any case 4 is not

Algorithm 1: Pseudo-code of the leader election algorithm

Input : Connected, undirected graph $\mathcal{G} = (V, E)$ and node IDs;
Output: i's leader σ_i;
States : The distance estimate $\hat{d}_i$ of i from i's leader; Radius of influence, R_i at i; Pseudo-diameter estimate D_i at i; The leader of i, σ_i;

---

1   $t = 0$;
2   Initialize $\hat{d}_i(0), \sigma_i(0), R_i(0), D_i(0)$ for all $i \in V$, set function g;
3   /* line 5 to 16: the G block; line 18: the C block   */
4   **foreach** $i \in V$ *and* $t \geq 0$ **do**
5   /* The set of neighbors following leaders with priority higher than i and having distance estimates less than radii of influence. Elements of this set are called *eligible* neighbors.   */
      $\mathcal{E}_i(t+1) = \{j \in \mathcal{N}(i) \mid \hat{d}_j(t) < R_j(t) \wedge \sigma_j(t) < i\}$;
6   **if** $\mathcal{E}_i(t+1) = \varnothing$ **then**
7   /* If it has no *eligible* neighbors, then i elects itself a leader, sets its distance estimate to zero and its radius of influence to $g(D_i)$.   */
        $\sigma_i(t+1) = i, \hat{d}_i(t+1) = 0, R_i(t+1) = g(D_i(t))$;
8   **else**
9   /* Set of *eligible* neighbors following the highest priority leaders.   */
        $\mathcal{E}_i^{\sigma}(t+1) = \{j \mid j = \arg\min_{\ell \in \mathcal{E}_i(t+1)}\{\sigma_\ell(t)\}\}$;
10   /* If $\mathcal{E}_i^{\sigma}$ has multiple members find those with the smallest distance estimates.   */
        $\mathcal{E}_i^{d}(t+1) = \{j \mid j = \arg\min_{\ell \in \mathcal{E}_i^{\sigma}(t+1)}\{\hat{d}_\ell(t)\}\}$;
11   /* If $\mathcal{E}_i^{d}$ has multiple members find those with the largest radii of influence.   */
        $\mathcal{E}_i^{R}(t+1) = \{j \mid j = \arg\max_{\ell \in \mathcal{E}_i^{d}(t+1)}\{R_\ell(t)\}\}$;
12   /* Find j, the highest priority member of this reduced set. Call it the *constraining node of i*.   */
13      $j = \min \mathcal{E}_i^{R}(t+1)$;
14   /* i follows the constraining node j's leader, sets its distance estimate to one more than j's distance estimate, adopts j's radius of influence as its own.   */
15      $\sigma_i(t+1) = \sigma_j(t), \hat{d}_i(t+1) = \hat{d}_j(t)+1, R_i(t+1) = R_j(t)$;
16   **end**
17   /* Find i's pseudo-diameter estimate.   */
18     $D_i(t+1) = \max\left\{\hat{d}_i(t+1), \{D_j(t) \mid j \in \mathcal{N}(i) \wedge \hat{d}_j(t) > \hat{d}_i(t+1)\}\right\}$;
19     $t = t+1$;
20 **end**

---

an eligible neighbor as $\hat{d}_4(0)$, the distance estimate of 4 is not less than $R_4(0)$, 4's radius of influence. Thus 2 and 3 become leaders at $t = 1$.

As $\sigma_3(0) = 4$, 3 is not 4's eligible neighbor, but 1 and 2 are as $\hat{d}_i(0) < R_i(0)$ and $\sigma_i(0) < 4$ for each. Thus, in (5) one among the second tuples wins. As $\sigma_1(0) = \sigma_2(0)$, in the lexicographic minimization there is a tie in the first variable. As $\hat{d}_1(0) = \hat{d}_2(0)$, there is a further tie in the second variable. Then 1 wins out as $R_1(0) > R_2(0)$ and 1 is the constraining node of 4, i.e. $c_4(1) = 1$. Consequently $(\sigma_4(1), \hat{d}_4(1), -R_4(1)) = (2, 4, -6)$. As 2 and 3 are leaders at $t = 1$, $(\sigma_2(1), \hat{d}_2(1), -R_2(1)) = (2, 0, -3)$ and $(\sigma_3(1), \hat{d}_3(1), -R_3(1)) = (3, 0, -7)$.

Turn now to (4). As $\hat{d}_i(1) = 0 < \hat{d}_4(0)$, for all $i \in \{1, 2, 3\}$, $D_i(1) = D_4(0) = 1$. On the other hand the set $\{j \mid \hat{d}_j(0) > \hat{d}_4(1)\} = \varnothing$. Thus, $D_4(1) = \hat{d}_4(1) = 4$. As 1, is the highest priority node, $\sigma_1(t) = 1$ and $\hat{d}_1(t) = 0, \forall\, t > 0$. Thus 1 is an eligible neighbor of and being a leader with higher priority than 4, remains 4's leader and propagates its radius of influence to 4: $(\sigma_4(2), \hat{d}_4(2), -R_4(2)) = (1, 1, -3)$. Henceforth due to (11), $R_1(t) = 2D_1(t) + 1 > \hat{d}_1(t)$, i.e. 1 remains a valid neighbor of 4, ensuring that 4 converges at $t = 1$, i.e., $\sigma_4(t) = 1$ and $\hat{d}_4(t) = 1$ for $t \geq 1$.

For $t \geq 2$ because of (11) $D_4(t) \geq \hat{d}_4(t)$. Further, for $t \geq 2$, as $\hat{d}_4(t-1) > \hat{d}_1(t) = 0$, it follows from (4) that $D_1(t) = D_4(t) \geq \hat{d}_4(t)$. Thus for $R_1(t) = 2D_1(t) + 1 = 2D_4(t) + 1 > \hat{d}_4(t)$. As 4 inherits 1's radius of influence at the next iteration, for $t \geq 4$, 4 is an eligible neighbor of both 2 and 3, whereupon both converge.

The mechanism of convergence is thus: Node 1 converges at $t = 1$, after which convergence proceeds in waves. Suppose all nodes within a distance d have converged.

(I) **Diameter estimates are sent back to leaders:** In an inward wave, using (4), all followers including the converged ones, transmit back to 1 their pseudo-diameter estimates. If a converged node i is $d_i = \hat{d}_i(t) = d$ away from 1, then due to (4) $D_i(t) \geq \hat{d}_i(t) = d$. On reaching 1 in d steps, $D_1(t) \geq d$. As $g(x) > x$, 1's radius of influence $R_1(t) = g(D_1(t)) > d$ exceeds the distance estimate of any such i.

(II) $R_1(t)$ **is broadcast by 1 to its followers:** In an outward wave, 1 broadcasts to its converged followers $R_1(t)$. A converged follower j, d hops from 1 receives this in d steps and at this point j's radius of influence $R_j(t) = R_1(t) > d \geq d_j = \hat{d}_j(t)$.

(III) **Converged nodes within d of 1, cause more nodes to converge:** Because of (II) every converged node j at a distance d from 1, has $\hat{d}_j(t) < R_j(t)$. As it also follows the highest priority node 1 it is an eligible neighbor for any neighbor k with $d_k = d+1$. This induces k to converge with, $\hat{d}_k(t) = d+1 = d_k$. The radius of convergence of k also becomes $R_k(t) = R_1(t) = g(D_1(t)) \geq g(d)$. As long as $d+1 = d_k < g(d)$, k becomes eligible, and induces its neighbors at $d+2$ hops from 1 to converge. As (1) ensures that every node at distance $d+1$ from 1 must have at least one neighbor d away from 1, nodes $d_j \leq g(d)$ hops from 1 converge in d_j+1 steps after 1 acquires the pseudo-diameter $D_1(t) \geq d$.

In (5), with ties in $\sigma_j(t)$, choosing a constraining node j with the smallest $\hat{d}_j(t)$, and with further ties, the largest $R_j(t)$, speeds j's becoming eligible. If converged these nodes can induce neighbors to converge quicker.
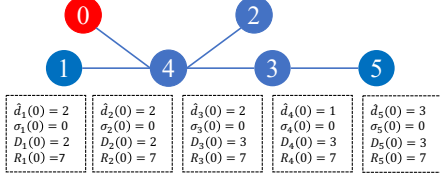
Fig. 3. Resilience: 0 the global leader is lost at $t = 0$.

3.3 The role of radii of influence in resilience

Consider Fig. 3 where the algorithm has converged at $t = 0$ when it loses its erstwhile leader 0. All nodes have $\hat{d}_i(0) < R_i(0)$ and constrain one or more of their neighbors. For example, 1 and 4 keep constraining each other. The result is that their distance estimates keep increasing while the radii of influence remain constant as 1 and 4 keep exchanging them. Eventually $R_4(t) = \hat{d}_4(t)$ and 1 lacking an eligible neighbor becomes a leader. Other fake (leader) IDs are similarly flushed out and recovery occurs.

In contrast consider a *min-consensus algorithm*: $\sigma_i(t + 1) = \min_{\{j \in \mathcal{N}(i) \cup \{i\}\}} \sigma_j(k)$. This will eventually provide each node with $\sigma_i(t) = 1$. Should 1 disappear thereafter, the other nodes will continue to exchange 1, as it is smaller than their IDs, *with no new leader emerging.*

Note a tension between resilience and convergence speed. A small $g(\cdot)$ leads to smaller R_i and in the example above, $R_i(t) = \hat{d}_i(t)$ occurs quicker. A large $g(\cdot)$ leads to faster convergence as more converged nodes become eligible quicker. Section 5.3 provides a $g(\cdot)$ that balances the two. The lemma below is key to proving resiliency.

Lemma 1. *Under (5), there holds:*
$$\hat{d}_i(t) \leq R_i(t), \quad \forall i \in V \text{ and } \forall t \geq 1. \quad (12)$$
Proof. Consider $t > 0$. If i is a leader, $\hat{d}_i(t) = 0 \leq R_i(t)$ by (6) and (3). If it is not a leader it must have an eligible neighbor which its constraining node $j = c_i(t+1)$ obeys $\hat{d}_j(t - 1) < R_j(t - 1)$. From (7) $\hat{d}_i(t) = \hat{d}_j(t - 1) + 1$ and $R_i(t) = R_j(t - 1)$. Hence $\hat{d}_i(t) = \hat{d}_j(t - 1) + 1 \leq R_j(t - 1) = R_i(t)$. $\square$

4 Analysis Framework and Stationary Point

The following is our standing set of assumptions.

Assumption 1. *The graph $\mathcal{G}$ is connected and undirected, $t_0 = 0$ is the initial time, $d_i, \mathcal{N}(i), c_i(t+1), \sigma_i(t), \hat{d}_i(t), R_i(t), D_i(t), g(\cdot)$ are defined as in Section 3, and for all $i \in V$, $\hat{d}_i(0), R_i(0), D_i(0)$ are all in $\mathbb{N}$.*

Note that we *do not* assume that $\sigma_i(0) \in V$. The case of $\sigma_i(0) \notin V$ models the situation where due to the loss of previous leaders, some nodes *have invalid leader ID* at $t = 0$. Even though lost leaders may have had IDs below 1, we allow such *fake IDs* to be any number not in V.

Theorem 1 shows that the desired state, with the single leader 1, is the unique stationary point of the algorithm.

Theorem 1. *Suppose Assumption 1 holds and d_i is as in (1), $g(\cdot)$ is as in Definition 3 and $\mathcal{D}$ as in Definition 2. Then the unique stationary point of the algorithm in (5) and (4) is given by (10) and*
$$D_i(t) \equiv D_i^* = \max\left\{d_i, \{D_j^* \mid j \in \mathcal{N}(i) \wedge d_j > d_i\}\right\}. \quad (13)$$
Proof. Because of (4), (10) implies (13). To see that (13) implies $D_1^* = \mathcal{D}$, note that as the graph is connected there is a set of nodes $\mathcal{C} = \{1 = i_0, \cdots, i_{\mathcal{D}}\}$ such that i_k is the true constraining node of i_{k+1} (Definition 1). Note $d_{i_k} = k$. Use induction to show that $\forall j \in \mathcal{C}, D_j^* = \mathcal{D}$. As $d_{i_{\mathcal{D}}} = \mathcal{D} \geq d_l, \forall l \in V$, (13) implies $D_{i_{\mathcal{D}}}^* = \mathcal{D}$. If $D_{i_k}^* = \mathcal{D}$ for some $k > 0$, then as $i_k \in \mathcal{N}(i_{k-1})$ and $d_{i_k} > d_{i_{k-1}}$, $D_{i_{k-1}}^* = \max\left\{d_{i_{k-1}}, \mathcal{D}\right\} = \mathcal{D}$, proving $D_1^* = \mathcal{D}$.

Suppose for all $i \in V$, $(\sigma_i(t), \hat{d}_i(t), R_i(t)) = (1, d_i, g(\mathcal{D}))$. Then from (4), $D_1(t+1) = \mathcal{D}$. Further, from (6) we have $\hat{d}_1(t+1) = 0 = d_1 = \hat{d}_1(t)$ and $R_1(t+1) = g(\mathcal{D}) = R_1(t)$.

Because of (3), the definition of $\mathcal{D}$ and (6),
$$\hat{d}_i(t) = d_i \leq \mathcal{D} < g(\mathcal{D}) = R_i(t). \quad (14)$$

Thus as each $i > 1$ there is a neighbor $j \in \mathcal{N}(i)$ with $\sigma_j(t) = 1 < i$, i has eligible neighbors, and from (7), $R_i(t+1) = R_j(t)$. Thus, $R_i(t+1) = g(\mathcal{D}) = R_i(t)$. From (1), for all $i > 1$, $\min_{j \in \mathcal{N}(i)}\{d_j = \hat{d}_j(t)\} = d_i - 1$. As all $\sigma_i = 1$, in (5) the neighbor with the smallest $\hat{d}_j(t)$ resolves the tie and $\hat{d}_i(t + 1) = d_i - 1 + 1 = d_i$. From (13), $D_i(t + 1) = D_i^*$, i.e. (10, 13) is a stationary point.

Define $\mathcal{S}_l = \{i | d_i = l\}$. To prove uniqueness, we show using induction that for all $l \in \{0, \cdots, \mathcal{D}\}$ and $i \in \mathcal{S}_l$ a stationary point must obey
$$\{\sigma_i(t), \hat{d}_i(t)\} \equiv \{1, d_i\}, R_i(t) \geq g(l), D_i(t) \geq l. \quad (15)$$

As $\mathcal{S}_0 = \{1\}$, because of (5) $\sigma_1(t) = \sigma_1(t + 1)$ must imply $\{\sigma_1(t), \hat{d}_1(t), R_1(t)\} = \{1, 0, g(D_1(t))\}$. As $D_i(t)$ are non-negative and $g(0) > 0$, (15) holds for $l = 0$. Suppose it holds for all $l \in \{0, \cdots, L - 1\}$ for some $L \leq \mathcal{D}$. Consider $i \in \mathcal{S}_L$. There must be a $j \in \mathcal{S}_{L-1}$ that is in $\mathcal{N}(i)$. From the induction hypothesis $\sigma_j(t) = 1$, $\hat{d}_j(t) = d_j = L - 1$, $R_j(t) \geq g(L - 1)$ and $D_j(t) \geq L - 1$. Thus as $\hat{d}_j(t) = L - 1 < g(L - 1) \leq R_j(t)$, from (5) $\sigma_i(t + 1) = \sigma_i(t)$ implies $\sigma_i(t + 1) = \sigma_j(t) = 1$ and $\hat{d}_i(t + 1) = d_j + 1 = d_i = L$. Further, from (4) we have $D_i(t+1) \geq d_i = L$. This proves (15). From (4) this also means $D_i(t + 1) = D_i^*$. As $D_1^* = \mathcal{D}$, $R_1(t) = g(\mathcal{D})$, and from (7) $R_i(t + 1) = g(\mathcal{D})$, for all $i \in \mathcal{S}_L$. $\square$

This is a desired stationary point as 1 is the sole leader. Not all $D_i^* = \mathcal{D}$. Node 2 in Figure 3 is an example: $d_2 = 2$, $d_4 = 1$ and $D_4 = \mathcal{D} = 3$. As $d_2 > d_4$, from (13), $D_2 = 2$. However, all radii of influence do equal $g(\mathcal{D})$. *Observe from (14) that the proof of (10,13) being a stationary point critically relies on (3).*

5 Stability and Resilience Analysis

In our framework, at $t = 0$ nodes may carry fake leader IDs due to the loss of a prior leader or the past advent of a fake leader. Section 5.1 proves that these fake IDs eventually disappear. Section 5.2 proves convergence. While a faster growing $g(\cdot)$ delays the disappearance of fake leaders, it speeds subsequent convergence. Section 5.3 seeks to balance these two competing requirements. Section 5.4 illustrates the *competitiveness* of convergence time.

5.1 Resilience to fake or lost leaders

We demonstrate the *resilience of the algorithm to the loss of leaders, temporary or otherwise, or of other nodes.* As noted earlier this is modeled as nodes carrying fake IDs. These IDs reflect nodes following leaders that have been lost. This thus permits recovery from the loss of multiple nodes (subject to continued connectivity) many of which may have been temporary leaders themselves. We define the set of *unrooted* nodes and prove that only unrooted nodes can carry fake leader IDs.

We assume that the loss occurs before $t = 0$ and at $t = 0$, even though this node is nonexistent, existing nodes may continue following it.

Definition 4 (Unrooted Node Set)**.** *Define the set of unrooted nodes $\mathcal{U}(t)$ as $\mathcal{U}(t + 1) = \{i \in V \mid i \neq c_i(t + 1) \in \mathcal{U}(t)\}$ and $\mathcal{U}(0) = V$. Thus $\mathcal{U}(t + 1)$ comprises non-leader nodes constrained by members of $\mathcal{U}(t)$. Let $\mathcal{L}(t) = \{\sigma_i(t) \mid i \in \mathcal{U}(t)\}$ be the set of unrooted leaders and $\mathcal{U}^k(t)$ be the set of unrooted nodes with leader k:*

$$\mathcal{U}^k(t) = \{i \in \mathcal{U}(t) \mid \sigma_i(t) = k\}.$$

Furthermore, define $\hat{d}^k_{\min}(t) = \min\{\hat{d}_i(t) \mid i \in \mathcal{U}^k(t)\}$ and $R^k_{\max}(t) = \max\{R_i(t) \mid i \in \mathcal{U}^k(t)\}$.

Recall a fake ID refers to a lost leader ID and is $\sigma_i(t) \notin V$. We show below that all fake IDs are in $\mathcal{L}(t)$. In Figure 3 after the loss of the leader 0, $\mathcal{U}(0) = V$ and $R_i(0) = 7 > \hat{d}_i(0)$ for all $i \in V$. As only 0 is a leader $\mathcal{L}(0) = \{0\}$, only $\mathcal{U}^0(0) = V$ is nonempty, $R^0_{\max}(t) = 7$ and $d^0_{\min}(t) = 1$. As $\hat{d}_i(0) < R_i(0)$ and each node follows 0, a higher priority ID than all $i \in V$, each node has an eligible leader and none leaves $\mathcal{U}(0)$. At $t = 1$ the $\hat{d}_i(t)$ is one greater than at least one of its neighbors. At the same time, as each i receives $R_i(1) = 7$ from a neighbor its radius of influenced is unchanged. Thus $\mathcal{U}(t)$, $\mathcal{L}(t)$ and $\mathcal{U}^0(t)$ are unchanged until $\hat{d}_4(5) = 7 = R_4(5) = \hat{d}_5(5) = R_5(5)$. Thus at $t = 5$, 1, 2 and 3 no longer have viable neighbors and at $t = 6$ each becomes a leader, i.e. $c_i(6) = i \; \forall \; i \in \{1, 2, 3\}$. Hence from the definition of $\mathcal{U}$, 1,2 and 3 leave $\mathcal{U}(6)$ but 4 and 5 remain and continue following 0 and $\mathcal{L}(6) = \{0\}$. At $t = 6$, both 4 and 5 have eligible neighbors with leaders (the neighbors themselves) with priority higher than them. Each starts following one among 1,2 and 3, $\mathcal{U}(t)$ empties, all fake IDs are flushed out and all nodes become rooted.

In this example and as proved below, $\mathcal{L}(t)$ does not grow and only unrooted nodes carry fake IDs, as IDs carried by rooted nodes are in V. *Thus, all fake IDs are in $\mathcal{L}(t)$.*

Lemma 2 ($\mathcal{L}(t)$ Does Not Grow)**.** *Consider (4,5) under Assumption 1. Then: (i) $\mathcal{L}(t)$ in Definition 4 obeys $\mathcal{L}(t + 1) \subseteq \mathcal{L}(t)$; (ii) $\sigma_i(t) \in V$ if $i \in V \setminus \mathcal{U}(t)$.*

Proof. Suppose $j \in \mathcal{L}(t + 1)$. By definition of $\mathcal{L}(t + 1)$, there exists $i \in \mathcal{U}(t + 1)$ such that $\sigma_i(t + 1) = j$. Thus from Definition 4 there exists a k such that $i \neq k = c_i(t + 1) \in \mathcal{U}(t)$, and $\sigma_k(t) = j$ by (7). Then $j \in \mathcal{L}(t)$ by definition of $\mathcal{L}(t)$, proving (i).

Use induction on t to prove (ii). For $t = 0$, (ii) is true as there are no rooted nodes $V = \mathcal{U}(0)$. From Definition 4, the constraining node of $i \in V \setminus \mathcal{U}(t+1)$ can only be such that either $c_i(t + 1) = i$ or $c_i(t + 1) = j \notin \mathcal{U}(t)$. In the former case, $\sigma_i(t + 1) = i \in V$. In the latter, $\sigma_i(t + 1) = \sigma_j(t)$, which is in V by the induction hypothesis. $\quad\square$

Recall the intuition behind resilience given in Section 3.2 through Figure 3: $\hat{d}_i(t)$ of nodes following a lost leaders grow, while their estimated R_i do not making them eventually ineligible neighbors. Lemma 3 together with Lemma 2 shows that this is true for all unrooted leaders i.e. members of $\mathcal{L}(t)$.

Lemma 3. *Under the conditions of Lemma 2, suppose at a $t \geq 0$, $\mathcal{L}(t)$ is nonempty and $k \in \mathcal{L}(t)$. Then, $\hat{d}^k_{\min}(t) \geq \hat{d}^k_{\min}(0) + t$ and $R^k_{\max}(t) \leq R^k_{\max}(0)$.*

Proof. By Lemma 2, $k \in \mathcal{L}(t)$ implies $k \in \mathcal{L}(0)$ and $\hat{d}^k_{\min}(0)$, $R^k_{\max}(0)$ are well defined. Use induction on t. For $t = 0$ the result is trivial. Assume that $t > 0$ and $i \in \mathcal{U}^k(t)$. By Definition 4, $j \in \mathcal{U}(t-1)$ and $j \neq i$. From (7), $\sigma_i(t) = \sigma_j(t - 1) = k$, thus $j \in \mathcal{U}^k(t - 1)$. By induction hypothesis and (7), $R_i(t) = R_j(t - 1) \leq R^k_{\max}(0)$, and $\hat{d}_i(t) = \hat{d}_j(t - 1) + 1 \geq \hat{d}^k_{\min}(0) + (t - 1) + 1$. $\quad\square$

We now prove resilience by showing that all fake ID's disappear, i.e. for some T_F, and all $t \geq T_F$ and $i \in V$, $\sigma_i(t) \in V$. In view of Lemma 2, this requires us to show that no member of $\mathcal{L}(t)$ that is not in V can lead any member of V after a certain time T_F. This follows from Lemma 3 consequent to which after a time all members of $\mathcal{U}(t)$ following a member of $\mathcal{L}(t)$ have radii of influence no greater than their distance estimates, i.e. are ineligible neighbors. This is also consistent with the mechanism described in Section 3.2 through Figure 3. As for $t \geq T_F$, 1 does not have a neighbor carrying a fake ID, 1 immediately becomes a leader. These time bounds are conservative and recovery may be faster. In fact, as shown in the lemma after $t = T_0 \leq T_F$ node has higher ranked leader than 1. This is so as T_0 is the time at which nodes with IDs less than 1 become ineligible. Thus 1 has the highest priority after this all will eventually follow 1.

Lemma 4 (Fake Leader IDs Disappear)**.** *Suppose the conditions of Lemma 3 hold. (i) No node follows a fake leader after $t = T_F$ below, i.e. $\sigma_i(t) \in V$ for all $i \in V$ and*

7

$t \geq T_F = 1 + \max\{0, R_{\max}^k(0) - \hat{d}_{\min}^k(0) \mid k \in \mathcal{L}(0) \setminus V\}.$

(ii) In fact for all $i \in V$ $\sigma_i(t) \geq 1$, $\forall\, t \geq T_0$ and (iii) $\sigma_1(t) = 1$, $\forall\, t \geq T_0$, *where*

$T_0 = 1 + \max\{0, R_{\max}^k(0) - \hat{d}_{\min}^k(0) \mid k \in \mathcal{L}(0) \wedge k < 1\}.$

Proof. Suppose there are $i \in V$ and $t \geq T_F$ such that $\sigma_i(t) = k \notin V$. By Lemma 2, $k \in \mathcal{L}(0)$ and $i \in \mathcal{U}^k(t)$ since $k \notin V$. By Lemma 3, $\forall\, t \geq T_F$:

$$\hat{d}_i(t) \geq \hat{d}_{\min}^k(t) \geq \hat{d}_{\min}^k(0) + t \geq \hat{d}_{\min}^k(0) + T_F \qquad (16)$$
$$\geq \hat{d}_{\min}^k(0) + R_{\max}^k(0) - \hat{d}_{\min}^k(0) + 1 = R_{\max}^k(0) + 1.$$

From Lemma 3 and the definition of $R_{\max}^k(t)$ we obtain: $R_i(t) \leq R_{\max}^k(t) \leq R_{\max}^k(0)$. Thus, $\hat{d}_i(t) > R_{\max}^k(0) \geq R_i(t)$, contradicting (12) as $T_F > 0$. Similarly, (ii) is proved by noting that $\min V = 1$. Thus if $\sigma_i(t) = \ell < 1$, then $\ell \in \mathcal{L}(0)$ and for $t \geq T_0$, (1) is again violated. Finally, (iii) follows because of (6) and the facts that $1 \leq T_0 \in \mathbb{N}$ and for $t \geq T_0$, $\sigma_i(t) \geq 1$, for all $i \in V$. $\square$

Thus fake IDs vanish after $T_F \geq T_0$. As no fake leader with ID greater than 1 exists at T_0, node 1 becomes a leader at T_0 even before all fake IDs vanish.

5.2 Global Uniform Asymptotic Stability

We now show that a node following the desired leader eventually cannot have an underestimated distance.
Lemma 5 (Underestimates Decay). *Consider (4, 5) under Assumption 1. For every $i \in V$ and $t \geq 0$ such that $\sigma_i(t) = 1$, we have $\hat{d}_i(t) \geq \min(d_i, t)$.*

Proof. Use induction on $t \geq 0$. As $\hat{d}_i(t) \geq 0$, the result holds for $t = 0$. Suppose it holds for some $t \geq 0$, and let i be such that $\sigma_i(t+1) = 1$. If $i = 1$ then from (6), the result holds as $d_1 = 0 = \hat{d}_i(t+1)$. If $i \neq 1$, then from (7) there is a $j \in \mathcal{N}(i)$ such that $\sigma_j(t) = 1$ and $\hat{d}_i(t+1) = \hat{d}_j(t) + 1$. By induction hypothesis and (1),

$$\hat{d}_i(t+1) \geq \min(d_j, t) + 1 \geq \min\left(\min_{k \in \mathcal{N}(i)}\{d_k\} + 1, t+1\right)$$
$$= \min(d_i, t+1). \qquad \square$$

The next lemma documents the inward flow of pseudo-diameter estimates explained in (I) of Section 3.2. Recall diameter estimates are no smaller than distance estimates. Thus if all nodes at a distance x from 1 have converged at time T_x then it takes the diameter estimate, which is greater than x at the furthest converged node, at most x iterations to reach 1. As $g(x) > x$, from (6), $R_1(t) = g(x) > x$ for all $t \geq T_x + x$.
Lemma 6 (Diameter Collection). *Under the conditions of Lemma 4, suppose T_x for $x \leq \mathcal{D}$ is such that every device $i \in V$ with $d_i \leq x$ obeys $\sigma_i(t) = 1$, $\hat{d}_i(t) = d_i$ for all $t \geq T_x$. Then $D_1(t) \geq x$ for all $t \geq T_x + x$.*

Proof. We prove by induction on k that for all $t \geq T_x + k$ and $0 \leq k \leq x$, there exists i such that $d_i = x - k$ and $D_i(t) \geq x$. Then the result will follow from $k = x$, as 1 is the only node i such that $d_i = 0$. Consider $k = 0$, $t \geq T_x$ and any i with $d_i = x \leq \mathcal{D}$. From (4), $D_i(t)$ equals

$\max\{\hat{d}_i(t), \{D_j(t-1) \mid j \in \mathcal{N}(i) \wedge \hat{d}_j(t-1) > \hat{d}_i(t)\}\}$
$\geq \hat{d}_i(t) = d_i = x.$

Suppose this is true for some $0 \leq k < x$. Consider $t \geq T_x + k + 1$. By induction hypothesis, there is an i such that $d_i = x - k$ and $D_i(t-1) \geq x$. With j a true constraining node of i, $\hat{d}_i(t-1) = d_i = x - k = d_j + 1$. Thus $x - (k+1) = d_j = \hat{d}_j(t)$. Then the result follows as $D_j(t)$ equals

$\max\{\hat{d}_j(t), \{D_\ell(t-1) \mid \ell \in \mathcal{N}(j) \wedge \hat{d}_\ell(t-1) > \hat{d}_j(t)\}\}$
$\geq D_i(t-1) \geq x.$ $\qquad \square$

Thus if all nodes with $d_i \leq x$ converge at $t = T_x$, then $R_1(t) = g(x) > x$ for all $t \geq T_x + x$. The next lemma documents the outward flow of radii of influence from 1, leading to more converged nodes, as explained in (II) and (III) of Section 3.2. Consider any node $d_i < g(x)$ away from 1. Once $R_1(t) = g(x) > d_i$, nodes converge and receive R_1 one hop at a time, becoming eligible neighbors as long as their distances are less than $g(x)$. They then induce their neighbors one hop further away from 1, to converge. Thus, all nodes with $d_i < g(x)$ converge in $T_x + x + d_i + 1$ steps.
Lemma 7 (Radius Broadcast). *Under the conditions of Lemma 4, suppose $T \geq T_0$ is such that $D_1(t) \geq x$ for $t \geq T$. Consider $i \in V$ with $d_i \leq g(x)$. Then for all $t \geq T + d_i + 1$, $\sigma_i(t) = 1$, $\hat{d}_i(t) = d_i$, and $R_i(t) \geq g(x)$.*

Proof. Use induction on y, with $0 \leq d_i = y \leq g(x)$. If $y = 0$, then $i = 1$. From Lemma 4, for $t \geq T_0$, $\sigma_i(t) = 1$, $\hat{d}_i(t) = d_i = 0$. From (6) and Definition 2, $R_i(t+1) = g(D_1(t)) \geq g(x)$, initiating the induction.

Suppose the result holds for some $0 \leq y < g(x)$ and every $m \in V$ with $d_m = y$. Consider i with $d_i = y + 1$ and $t \geq T + y + 2$. From Definition 1, $j \in \mathcal{N}(i)$ is a true constraining node of i iff $d_j = y$. Thus by induction hypothesis for any j a true constraining node of i, $\sigma_j(t-1) = 1$, $\hat{d}_j(t-1) = d_j < g(x)$, $R_j(t-1) \geq g(x)$. Thus $j \in \mathcal{E}_i(t)$ given in (8).

Suppose $k = c_i(t)$ is the constraining node of i. Because of (5), $1 \leq \sigma_k(t-1) \leq \sigma_j(t-1) = 1$, and $y \leq \hat{d}_k(t-1) \leq \hat{d}_j(1-1) = y$. Thus $\sigma_k(t-1) = 1$ and $\hat{d}_k(t-1) = y$. As $k \in \mathcal{N}(i)$, $d_k \geq y$. Thus from Lemma 5, $y = \hat{d}_k(t-1) \geq \min(d_k, T + y + 2)$ i.e., $y \geq d_k \geq d_i - 1 = y$ and k is a true constraining node of i. Thus, from (7), $\hat{d}_i(t) = y + 1$, $\sigma_i(t) = 1$ and $R_i(t) = R_k(t-1) \geq g(x)$. $\qquad \square$

Combining the last two lemmas, Theorem 2 recursively

characterizes the overall convergence time T_x for nodes at distance x. It uses the definition below.

Definition 5 (Discrete Inverse). *Define $g^{-1}(D)$ as the smallest $x \in \mathbb{N}$ such that $g(x) \geq D$. Note $g^{-1}(\cdot) : \mathbb{N} \to \mathbb{N}$.*

As from Definition 3, $g(x) > x$, $g^{-1}(x) < x$. Thus

$$g^{-1}(x) \leq x - 1, \; g^{-1}(1) = 0 \text{ and } g\left(g^{-1}(x)\right) \geq x. \quad (17)$$

Theorem 2 (Convergence). *Under the conditions of Lemma 4 with T_0 given there,*

$$\sigma_i(t) = 1, \quad \text{and } \hat{d}_i(t) = d_i, \quad (18)$$

for all i with $0 \leq d_i \leq x \leq \mathcal{D}$ and

$$t \geq T_x = T_{g^{-1}(x)} + g^{-1}(x) + x + 1. \quad (19)$$

Proof. Use induction. The result is true for $x = 0$ from Lemma 4, as for all $t \geq T_0$, $\sigma_1(t) = 1$ and $\hat{d}_1(t) = 0 = d_1$. Because $g^{-1}(x) < x$, sustain the induction by assuming that for some x and all i such that $0 \leq d_i \leq g^{-1}(x) < \mathcal{D}$ and $t \geq T_{g^{-1}(x)}$ given by (19), (18) holds. By Lemma 6, $D_1(t) \geq g^{-1}(x)$ for $t \geq T_{g^{-1}(x)} + g^{-1}(x)$. From Lemma 7, (18) holds for all $t \geq T_{g^{-1}(x)} + g^{-1}(x) + x + 1 = T_x$ and i such that $d_i \leq g\left(g^{-1}(x)\right)$. As $g\left(g^{-1}(x)\right) \geq x$, (18) holds for all i such that $d_i \leq x$ and $t \geq T_x$ in (19). $\quad\square$

Thus $\hat{d}_i(t)$ and $\sigma_i(t)$ converge by $T_{\mathcal{D}}$. With initial time t_0, this shifts to $t_0 + T_{\mathcal{D}}$, making stability uniform. Notice that the previous theorem does not prove convergence for the hidden variables $D_i(t)$ and $R_i(t)$. However, once $\hat{d}_i(t)$ and $\sigma_i(t)$ converge, another application of Lemmas 6 and 7 proves convergence also for them in an additional $2\mathcal{D}$ time ($\mathcal{D}$ for collection, reaching the limit $D_i(t)$ values, then $\mathcal{D}$ for broadcast to propagate the final $R_i(t)$ values).

Suppose $L(x)$ is the smallest integer for which $g^{-L(x)}(x) = 0$. Then for all $0 \leq x \leq \mathcal{D}$ and $x \in \mathbb{N}$, (19) becomes

$$T_x = T_0 + x + \sum_{k=1}^{L(x)} (2g^{-k}(x) + 1). \quad (20)$$

Indeed, from (19) and (17), $L(1) = 1$ and we have $T_1 = T_0 + 2 = T_0 + 1 + \sum_{k=1}^{L(1)} (2g^{-k}(1) + 1)$. Furthermore $L(g^{-1}(x)) = L(x) - 1$. Thus assuming (20) to be true for all nonnegative integer $x = y - 1 \geq 1$, from (17):

$$T_{g^{-1}(y)} = T_0 + g^{-1}(y) + \sum_{k=1}^{L(g^{-1}(y))} \left(2g^{-k}(g^{-1}(y)) + 1\right).$$

Thus

$$T_y = T_0 + 2g^{-1}(y) + \sum_{k=1}^{L(y)-1} \left(2g^{-(k+1)}(y) + 1\right) + y + 1$$

$$= T_0 + y + \sum_{k=1}^{L(y)} (2g^{-k}(y) + 1).$$

5.3 Tuning $g(x)$ for optimality

A slow growing $g(\cdot)$ improves T_0, the time to recovery from fake and lost leaders, while the remaining terms in $T_{\mathcal{D}}$ decline with a fast growing $g(\cdot)$. We now tune $g(\cdot)$ to balance these requirements. For the slowest growing $g(x) = x + 1$, $g^{-k}(x) = x - k$ for $x \geq k$, $L(x) = x$. Thus

$$T_x = T_0 + x + \sum_{k=1}^{x} (2(x - k) + 1) = T_0 + x(x + 1).$$

Hence $T_{\mathcal{D}}$ is quadratic in $\mathcal{D}$, and thus not time optimal.

Contrast this to the setting of convergence after loss of the true leader. Relabel the highest priority node as 1, and reset the the time at which recovery starts as 0. From Theorem 1, $D_1(0) = \mathcal{D}$ and the time to recovery T_0 in Lemma 4, it is readily seen that

$$\max\{0, R^k_{\max}(t) - \hat{d}^k_{\min}(t) \mid k \in \mathcal{L}(t) \wedge k < 1\} \leq g(\mathcal{D}).$$

and from Lemma 4 in the worst case $T_0 = 1 + g(\mathcal{D})$. Accordingly, to *balance convergence and recovery times* we consider the choice of $g(\cdot)$ that leads to minimizing

$$T_x = 1 + g(x) + x + \sum_{k=1}^{L(x)} (2g^{-k}(x) + 1) \quad (21)$$

with $x = \mathcal{D}$. Due to the second term, it is important to choose $g(x)$ to be at least asymptotically linear. Thus we characterize the best coefficient $P = \lim_{x \to \infty} \frac{g(x)}{x}$ under the implicit assumption that as $\mathcal{D}$ tends to infinity

$$g(\mathcal{D}) = P\mathcal{D} + o(\mathcal{D}). \quad (22)$$

This implies that as $\mathcal{D}$ tends to infinity

$$g^{-k}(\mathcal{D}) = \frac{\mathcal{D}}{P^k} + o(\mathcal{D}). \quad (23)$$

As $g : \mathbb{N} \to \mathbb{N}$, as $\mathcal{D}$ tends to infinity

$$L(\mathcal{D}) = \log_P(\mathcal{D}) + o(\log \mathcal{D}). \quad (24)$$

Then we have the following lemma.

Lemma 8. *Assume that $1 < P = \lim_{x \to \infty} \frac{g(x)}{x}$ is finite. Then in (21), $\lim_{\mathcal{D} \to \infty} \frac{T_{\mathcal{D}}}{\mathcal{D}} = \frac{P^2 + 1}{P - 1}$.*

Proof. As $\mathcal{D}$ tends to infinity, from (21)-(24),

$$T_{\mathcal{D}} = (P + 1)\mathcal{D} + 2\sum_{k=1}^{L(\mathcal{D})} \frac{\mathcal{D}}{P^k} + o(\mathcal{D})$$

$$= (P - 1)\mathcal{D} + 2\mathcal{D}\frac{P - 1/P^{L(\mathcal{D})}}{P - 1} + o(\mathcal{D})$$

$$= (P - 1)\mathcal{D} + 2\mathcal{D}\frac{P - 1/\mathcal{D}}{P - 1} + o(\mathcal{D})$$

$$= \mathcal{D}\frac{P^2 + 1}{P - 1} + o(\mathcal{D}). \quad\square$$

Minimization of $(P^2 + 1)/(P - 1)$ yields the following.

Theorem 3. *Under the conditions of Lemma 8, the smallest* $\lim_{\mathcal{D}\to\infty} \frac{T_{\mathcal{D}}}{\mathcal{D}}$ *is* $2+2\sqrt{2}$*, attained for* $P = 1+\sqrt{2}$*.*

Thus for large $\mathcal{D}$, the optimal $g(x) = (1 + \sqrt{2})x + o(x)$, leading to an asymptotic convergence time of $2(1+\sqrt{2})\mathcal{D}$. This fact allows us to calculate the overall message complexity of the algorithm before convergence.

Corollary 1. *The total message complexity of (4,5) is* $O(D\mathcal{E})$*, where D and $\mathcal{E}$ are the diameter and the number of edges of $\mathcal{G}$, respectively.*

Proof. At each round, there are message exchanges between the linked nodes. Then the message complexity per round is $O(\mathcal{E})$ and the total message complexity is $O(D\mathcal{E})$ as GCF will converge within $O(\mathcal{D})$ rounds and $\mathcal{D}$ is upper bounded by the diameter of $\mathcal{G}$ by Definition 2. □

5.4 Competitiveness with optimal

The $T_{\mathcal{D}}$ in Section 5.3 is only asymptotically optimal. For finite diameters, we define K-competitive algorithms.

Definition 6. *Given O_x the shortest possible time to converge with knowledge of the diameter x, the algorithm is K-competitive (resp. for all $x \geq a$) if $T_x \leq KO_x$ for all $x \in \mathbb{N}$ (resp. $\forall\, x \geq a$).*

As O_x is unknown we use an *optimistic* estimate for it:

$$O_x = 2x + 1. \tag{25}$$

This is so as if $\mathcal{D}$ is *known*, the loss of a leader will require at least $\mathcal{D}$ iterations for its successor to learn of its disappearance (e.g., after not receiving messages from the lost leader), one iteration for the new leader to appoint itself, and $\mathcal{D}$ more steps to apprise others.

Through (25) *we are comparing with an algorithm which knows $\mathcal{D}$, which ours does not.* Nevertheless we settle on an incontrovertible, but conservative value. A candidate algorithm that achieves this bound is obtained by anchoring $R_i(t)$ to $g(\mathcal{D})$ in (5). Theorem 3 precludes the possibility of K-competitiveness with $K < 1 + \sqrt{2}$, as $\lim_{x\to\infty} T_x/O_x \geq \frac{1}{2}\lim_{x\to\infty} T_x/x \geq 1 + \sqrt{2}$. We argue in the sequel that our algorithm *is $\frac{32}{13}$-competitive* after a leader loss.

To show this we first present an algorithm that verifies K-competitiveness for GCF for a given K.[3] The GCF algorithm is K-competitive iff there is a progressive, monotonic $g(x)$ such for all $x \in \mathbb{N}$

$$g(x) \leq \lfloor KO_x \rfloor - 1 - x - \sum_{k=1}^{L(x)}(2g^{-k}(x) + 1). \tag{26}$$

---

[3] The C++ code for this algorithm is available at `https://github.com/fcpp-experiments/near-optimal-election` with a full license to reuse.

As all terms in the summation can be obtained from $g(y)$, with $y < x$, this provides a recursive computation of $g(x)$ that leads to K-competitiveness. Selecting the largest $g(x)$ that satisfies (26) at a given x is the best option, as it reduces the summation for larger values of x. To ensure monotonicity, if the obtained $g(x)$ is such that $g(x) < g(x - 1)$ we need to reset $g(x - 1) = g(x)$ (and possibly recursively other smaller values). This does not violate the previous constraints since $g^{-1}(x)$ is unchanged (since $g(x) > x$, any $z < x$ with $g(z) \geq x$). For a candidate K, K-competitiveness is false if for that K and some $x \in \mathbb{N}$, (26) yields $g(x) \leq x$. On the other hand if (4,5) is K-competitive then this recursion may not stop.

To circumvent such a potential non-termination, the theorem below provides a sufficient condition that helps finds a $g(x)$ that ensures K-competitiveness. It assumes that the recursion yields a progressive function $g(x)$, a K and an x_0 such that $T_x \leq KO_x$ for all $x \leq x_0$. It then provides a sufficient condition on $g(x)$ and x_0, so that a continuation of $g(x)$ can be found to satisfy $T_x \leq KO_x$ for all $x > x_0$.

Theorem 4. *Consider T_x as in (21) amd O_x as in (25). Suppose there is an $x_0 > 1$, $K \geq \sqrt{2}+1$ and a progressive monotonic function $g : \mathbb{N} \to \mathbb{N}$ such that $T_x \leq KO_x$, for all $x \leq x_0$. Assume, $g(x_0) > g(x_0 - 1)$. Define,*

$$\gamma = \max_{\substack{g^{-1}(x_0)<x\leq x_0, \\ g(x)>g(x-1)}} \{x - (\sqrt{2}-1)(g(x-1) + 1)\}, \tag{27}$$

$$\delta = \max\{\gamma, g^{-1}(x_0) - (\sqrt{2}-1)x_0\} \tag{28}$$

and $\alpha = (1 - \delta)(\sqrt{2} + 1)$. Suppose

$$x_0 \geq \frac{(2K - \sqrt{2})\delta + 2 + \sqrt{2}}{2\sqrt{2}\left((\sqrt{2} - 1)K - 1\right)}. \tag{29}$$

Then with

$$g(x) = \left\lfloor(\sqrt{2} + 1)x + \alpha\right\rfloor \ \forall\, x > x_0, \ x \in \mathbb{N}, \tag{30}$$

$T_x \leq KO_x$ *for all $x \in \mathbb{N}$.*

Proof. We first prove that

$$g^{-1}(x) \leq (\sqrt{2} - 1)x + \delta \ \forall\, x > x_0. \tag{31}$$

To this end we consider three cases that between them cover all $x > x_0$.:

Case I: $x_0 < x \leq g(g^{-1}(x_0))$**.**

Observe, $g^{-1}(g(g^{-1}(x_0))) = g^{-1}(x_0)$. Thus $x_0 < x \leq g(g^{-1}(x_0))$ implies $g^{-1}(x_0) \leq g^{-1}(x) \leq g^{-1}(x_0)$, i.e. $g^{-1}(x_0) = g^{-1}(x)$. Then from (28) we obtain:

$$g^{-1}(x) = g^{-1}(x_0) = (\sqrt{2} - 1)x_0 + g^{-1}(x_0) - (\sqrt{2} - 1)x_0$$
$$\leq (\sqrt{2} - 1)x_0 + \delta \leq (\sqrt{2} - 1)x + \delta.$$

Case II: $g(g^{-1}(x_0)) < x \leq g(x_0)$.

In this case, $g^{-1}(x_0) + 1 \leq z = g^{-1}(x) \leq x_0$. Further from (17), $z = g^{-1}(x)$ means $g(z) \geq x > g(z-1)$. From (27), it follows that $z - (\sqrt{2}-1)(g(z-1)+1) \leq \gamma$. As $\gamma \leq \delta$ by (28),

$$z = g^{-1}(x) \leq (\sqrt{2}-1)(g(z-1)+1) + \delta$$
$$\leq (\sqrt{2}-1)x + \delta \qquad (32)$$

where (32) uses the fact that $x \geq g(z-1) + 1$.

Case III: $x > g(x_0)$.

In this case, $g^{-1}(x) > x_0$, hence it only depends on the continuation of g defined in (30). Notice that given $w = (x - \alpha)(\sqrt{2}-1)$,

$$g(\lceil w \rceil) = \left\lfloor (\sqrt{2}+1)\lceil w \rceil + \alpha \right\rfloor$$
$$\geq \left\lfloor (\sqrt{2}+1)w + \alpha \right\rfloor = \lfloor x - \alpha + \alpha \rfloor = x.$$

As $y = g^{-1}(x)$ is the smallest integer such that $g(y) \geq x$, we have that $g^{-1}(x) \leq \lceil w \rceil$, thus

$$g^{-1}(x) \leq w + 1 = (x - \alpha)(\sqrt{2}-1) + 1$$
$$= (\sqrt{2}-1)x - (1-\delta)(\sqrt{2}+1)(\sqrt{2}-1) + 1$$
$$= (\sqrt{2}-1)x + \delta.$$

Thus as the three cases between them cover all $x > x_0$, (31) holds for all $x > x_0$. Use induction on x to prove the theorem. The result is true for $0 < x \leq x_0$. For $x > x_0$,

$$T_x = T_{g^{-1}(x)} + g^{-1}(x) + x + 1$$
$$< T_{g^{-1}(x)} + g^{-1}(x) + x + 1 + g(x) - x$$
$$\leq K(2g^{-1}(x)+1) + g^{-1}(x) + 1 + g(x) \quad (33)$$
$$= (2K+1)g^{-1}(x) + (K+1) + g(x)$$
$$\leq (2K+1)\left((\sqrt{2}-1)x + \delta\right) + (K+1) +$$
$$+ (\sqrt{2}+1)x + (1-\delta)(\sqrt{2}+1) \quad (34)$$
$$= K + (2K - \sqrt{2})\delta + 2 + \sqrt{2}$$
$$+ \left(2K(\sqrt{2}-1) + 2\sqrt{2}\right)x$$
$$< K + \left(2\sqrt{2}((\sqrt{2}-1)K - 1)\right)x_0$$
$$+ \left(2K(\sqrt{2}-1) + 2\sqrt{2}\right)x \quad (35)$$
$$< K + \left(2\sqrt{2}((\sqrt{2}-1)K - 1)\right)x$$
$$+ \left(2K(\sqrt{2}-1) + 2\sqrt{2}\right)x \quad (36)$$
$$= K(2x+1) = KO_x$$

where (33) uses the induction hypothesis that $T_{g^{-1}(x)} \leq K(2g^{-1}(x)+1)$, (34) uses (31) and (30), (35) uses (29), and (36) uses that $x > x_0$. $\qquad \square$
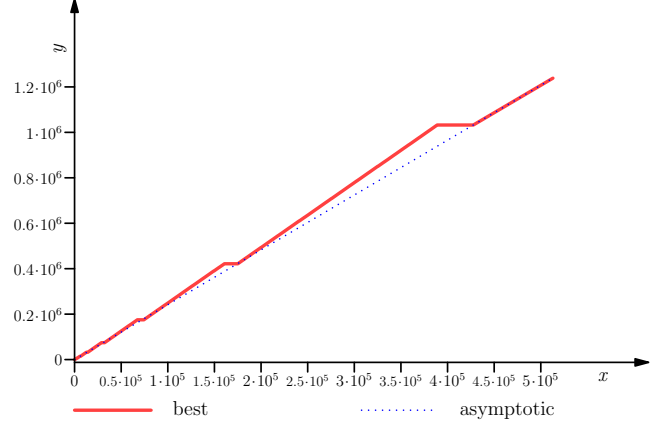

Fig. 4. Plot of the $\frac{32}{13}$-competitive function $g(x)$.

A g thus constructed has the best asymptotic behavior. To make $g(x)$ progressive we need $x_0 \geq -\frac{\alpha}{\sqrt{2}} = \frac{\delta-1}{2-\sqrt{2}}$. The algorithm in the repository shows that the algorithm is not $K-$competitive for $K < 32/13$. However, $g(x) = \lfloor \sqrt{2}x + 5.51 \rfloor$, $x \geq 427534$, with the first 427534 custom values given in the repository (depicted in Figure 4), is $\frac{32}{13}-$competitive. A variation of Theorem 4 also shows that $g(x) = \max(\lfloor (1+\sqrt{2})x + 4.6 \rfloor, 6)$ is $(\sqrt{2}+1)$-competitive for $x \geq 3$. This can be considered better, since graphs with $\mathcal{D} \leq 2$ are trivial.

6 Simulations

We performed simulations assessing the performance of the proposed algorithm through the FCPP simulator (Audrito 2020) implementing the field calculus language (Audrito *et al.* 2019)[4] . We compared the performance of the GCF algorithm introduced in this paper with the algorithm proposed by (Datta *et al.* 2011b)— knowledge-free leader election algorithms. For GCF, we set the parameter $g(x) = \max(\lfloor (1+\sqrt{2})x + 4.6 \rfloor, 6)$. We also considered *filtered* versions of both algorithms, which changes a node's leader only after the new leader persists for $\tau = 4$ rounds.

We placed 254 nodes uniformly distributed in a 20×2 rectangular area with unit disc communication, so that every node has approximately 20 neighbors on average. The initial conditions are: $\forall i \in V$, $\hat{d}_i(0) = D_i(0) = 0$, and $\sigma_i(0) = i$, and at $t = 100$ the leader is lost together with 10% of the other nodes. We considered both a synchronous scheduling, and an asynchronous scheduling (where devices appear at a random time between 0 and 20, and have a 25% error in round lengths). We varied device movement speed, from 0% to 60% of the connection radius per round, and also considered varying "crash probability" per round ranging from 0% to 6%.

---

[4] The source code for these simulations is available at `https://github.com/fcpp-experiments/near-optimal-election`, with a license to reuse.

When a device crashes, we assume it to be unavailable for 20 time units before becoming available again after a reboot. We ran several instances varying random seeds, averaging the results.

Figure 5 shows the results for the synchronous case, and Figure 6 for the asynchronous case. The results in both cases are very similar, showing that both GCF and Datta's algorithms are resilient to scheduling perturbations. Overall, the filtered versions succeed in reducing the total number of leaders, and in particular of spurious leaders (leaders that have never been the correct leader in the past), while on the other hand slowing down convergence (by τ rounds). In the simulations with device speed and crash probability, Datta's algorithm fails to converge at all, with the majority of the network retaining spurious leaders until the simulation end. On the other hand, GCF retains good performance even in the presence of movement and crashes, with only a slight lengthening of the convergence time. Even without speed and crashes, GCF significantly outperforms Datta's algorithm in the number of leaders (especially spurious one), and in the convergence time from scratch. The overall recovery time of both algorithms is similar, although with a different dynamics: Datta's algorithm starts immediately to recover, then proceeding slower; while GCF stays still for a while, then recovering faster.

Figure 7 shows the effect of speed and crashes for the synchronous case (results for the asynchronous case are omitted but similar, as shown by Figure 6). GCF shows very little degradation of performance increasing device speeds up to 60% of the communication radius per round, and crash probabilities up to 6% per round for each device. Datta's algorithm, instead, degrades exponentially, becoming practically unusable for speeds larger than 10% and crash probabilities larger than 2%. The observations made for Figure 5 reflect in these plots as well: filtering reduces spurious leaders and also slightly the number of nodes with the correct leader (due to convergence slowdown); and the different dynamics of GCF implies a lower number of correct nodes for zero speed and crash probability (although corresponding to an identical convergence time).

7   Conclusion

We have introduced a resilient leader election algorithm involving a feedback interconnection of aggregate computing building blocks. It assumes no prior knowledge about the network, and is not only globally uniformly and asymptotically stable, but also resilient to transient perturbations. The design function $g(\cdot)$ tunes the convergence and resilience rates. It has been shown that the proposed leader election algorithm is $\frac{32}{13}$-competitive with an *optimistic* optimal and stabilizes in $(2 + 2\sqrt{2})\mathcal{D}$ rounds after the loss of leaders.

References

Aggarwal, S. & S. Kutten (1993). Time optimal self-stabilizing spanning tree algorithms. In 'FSTTCS'. pp. 400–410.

Altisen, K., A. Cournier, S. Devismes, A. Durand & F. Petit (2017). 'Self-stabilizing leader election in polynomial steps'. *Information and Computation* **254**, 330–366.

Audrito, G. (2020). FCPP: an efficient and extensible field calculus framework. In 'IEEE ACSOS'. pp. 153–159.

Audrito, G., M. Viroli, F. Damiani, D. Pianini & J. Beal (2019). 'A higher-order calculus of computational fields'. *ACM Transactions on Computational Logic* **20**(1), 5:1–5:55.

Awerbuch, B., S. Kutten, Y. Mansour, B. Patt-Shamir & G. Varghese (1993). Time optimal self-stabilizing synchronization. In 'STOC'. pp. 652–661.

Baillieul, J. & A. Suri (2003). Information patterns and hedging brockett's theorem in controlling vehicle formations. In 'IEEE CDC'. Vol. 1. IEEE. pp. 556–563.

Beal, J. & M. Viroli (2014). Building blocks for aggregate programming of self-organising applications. In 'IEEE SASO Workshops'. pp. 8–13.

Beal, J., D. Pianini & M. Viroli (2015). 'Aggregate programming for the internet of things'. *Computer* **48**(9), 22–30.

Burman, J. & S. Kutten (2007). Time optimal asynchronous self-stabilizing spanning tree. In 'International Symposium on Distributed Computing'. pp. 92–107.

Cao, M., A. S. Morse & B. D. O. Anderson (2008). 'Agreeing asynchronously'. *IEEE Transactions on Automatic Control* **53**(8), 1826–1838.

Casteigts, A., Y. Métivier, J. M. Robson & A. Zemmari (2016). Deterministic leader election in $O(D + \log n)$ time with messages of size $O(1)$. In 'International Symposium on Distributed Computing'. pp. 16–28.

Dasgupta, S., B. D. O. Anderson, C. Yu & T. Summers (2011). Controlling rectangular formations. In '2011 Australian Control Conference'. IEEE. pp. 44–49.

Datta, A. K., L. L. Larmore & P. Vemula (2011*a*). 'An O(n)-time self-stabilizing leader election algorithm'. *Journal of Parallel and Distributed Computing* **71**(11), 1532–1544.

Datta, A. K., L. L. Larmore & P. Vemula (2011*b*). 'Self-stabilizing leader election in optimal space under an arbitrary scheduler'. *Theoretical Computer Science* **412**(40), 5541–5561.

Dubois, S. & R. Guerraoui (2013). Introducing speculation in self-stabilization: an application to mutual exclusion. In 'Proceedings of the 2013 ACM symposium on Principles of distributed computing'. pp. 290–298.

Fidan, B., S. Dasgupta & B. D. O. Anderson (2013). 'Adaptive range-measurement-based target pursuit'. *International Journal of Adaptive Control and Signal Processing* **27**(1-2), 66–81.

Kravchik, A. & S. Kutten (2013). Time optimal synchronous self stabilizing spanning tree. In 'International Symposium on Distributed Computing'. pp. 91–105.

Kutten, S., G. Pandurangan, D. Peleg, P. Robinson & A. Trehan (2015). 'On the complexity of universal leader election'. *Journal of the ACM (JACM)* **62**(1), 7.

Mo, Y., G. Audrito, S. Dasgupta & J. Beal (2020). 'A resilient leader election algorithm using aggregate computing blocks'. *IFAC-PapersOnLine* **53**(2), 3336–3341. 21st IFAC World Congress.
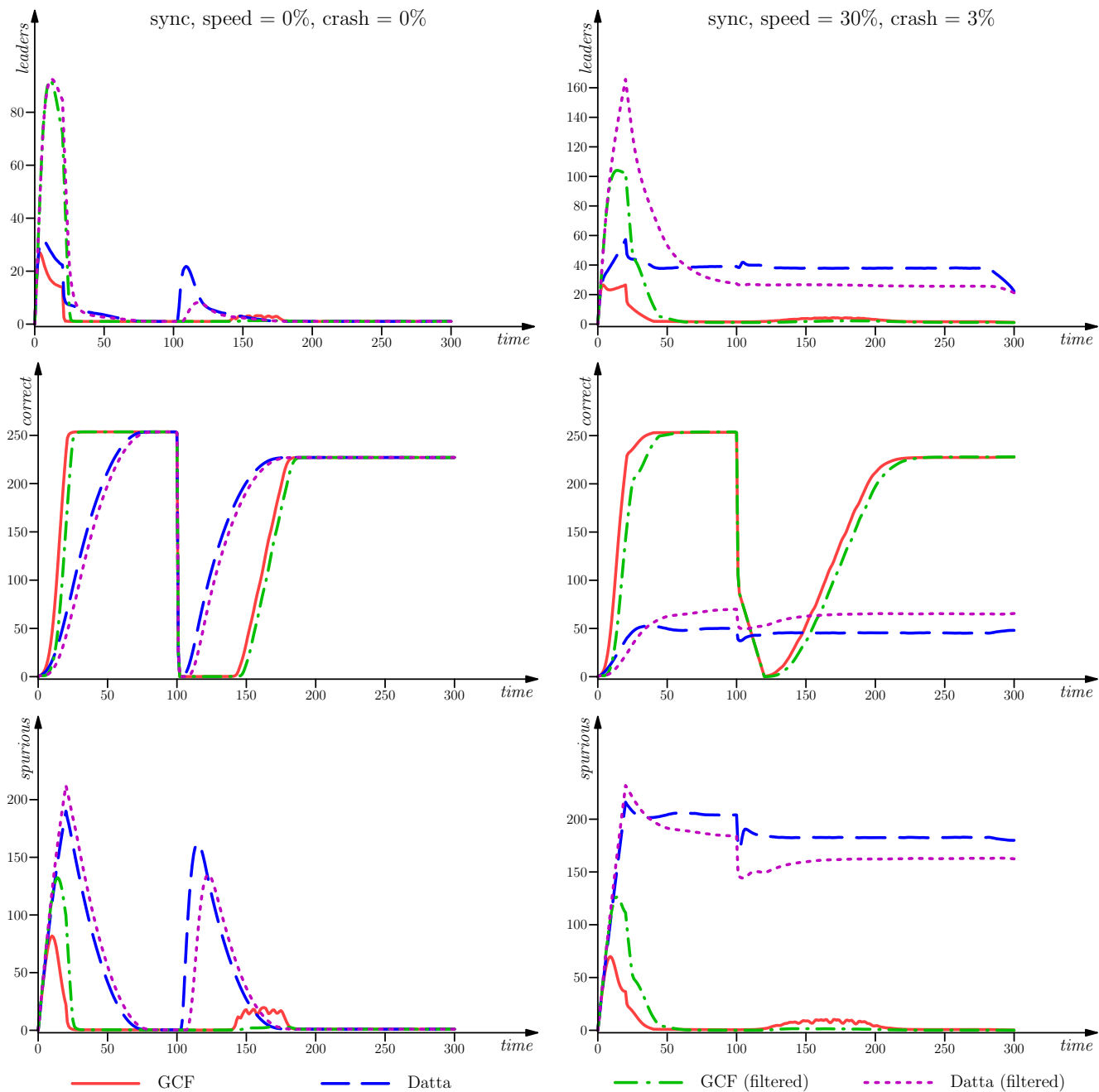
Fig. 5. Performance of GCF vs Datta's algorithm in a synchronous setting over time; with no device speed and crash probability (left) and with 30% speed and 3% crash probability (right); showing leader count (top), number of nodes with a correct leader (middle) and with a spurious leader (bottom). Values are averaged across 5000 different random seeds, and the total number of devices is 254.
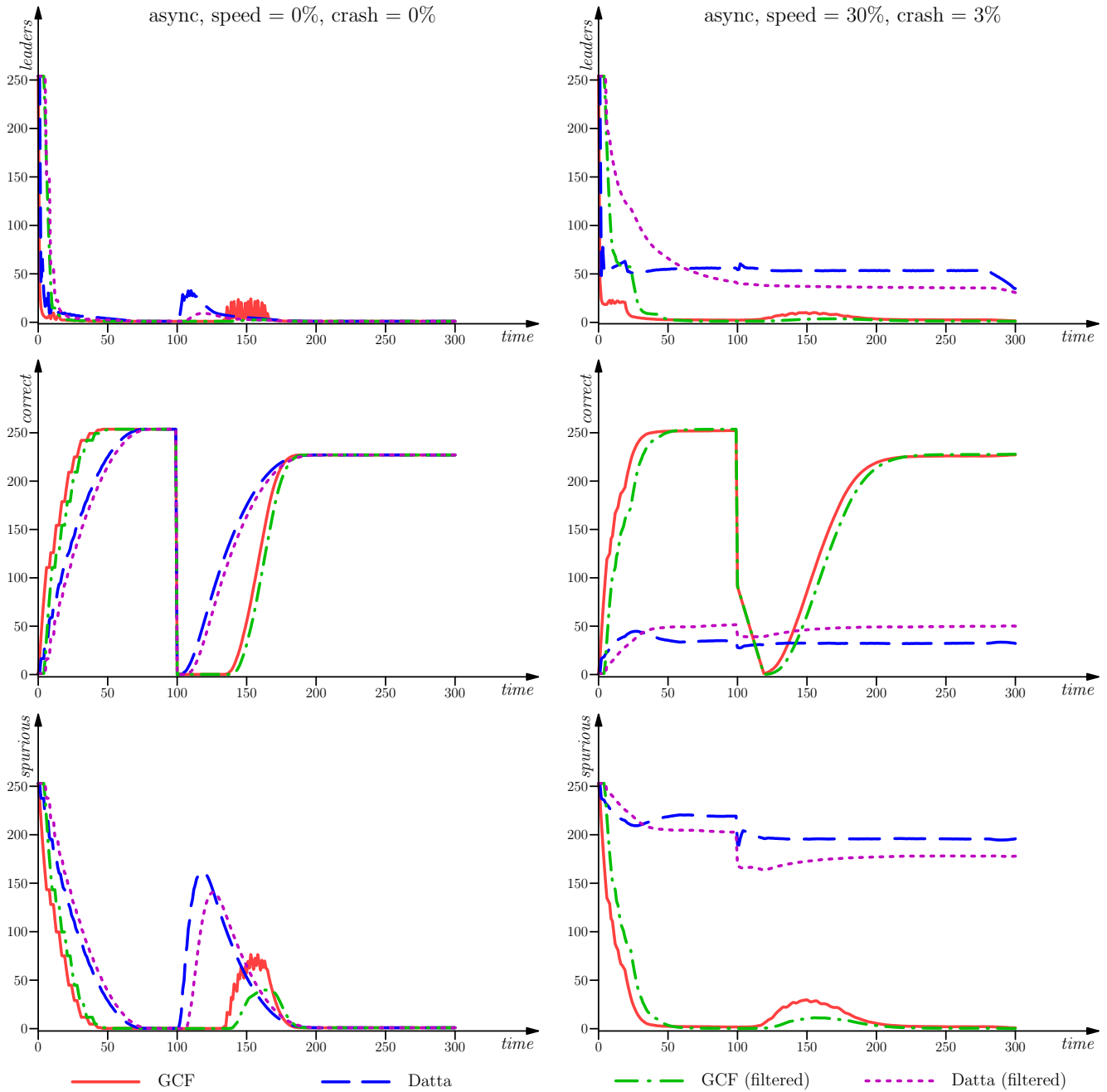
13

Fig. 6. Performance of GCF vs Datta's algorithm in an asynchronous setting over time; with no device speed and crash probability (left) and with 30% speed and 3% crash probability (right); showing leader count (top), number of nodes with a correct leader (middle) and with a spurious leader (bottom). Values are averaged across 5000 different random seeds, and the total number of devices is 254.
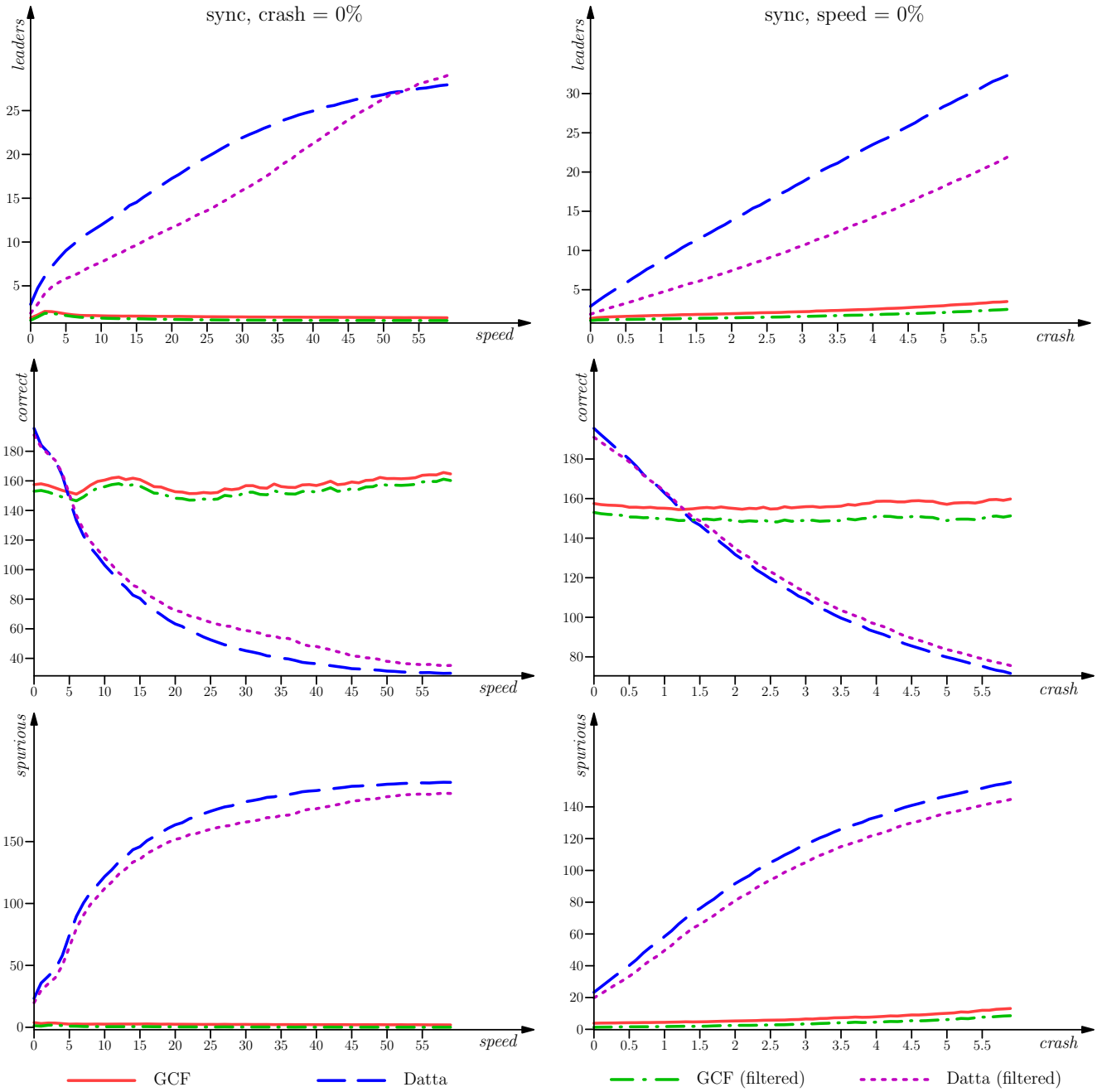
Fig. 7. Recovery performance of GCF vs Datta's algorithm in a synchronous setting; varying device speed with no crash probability (left) and varying crash probability with no device speed (right); showing leader count (top), number of nodes with a correct leader (middle) and with a spurious leader (bottom). Values are averaged across the recovery time 100-300s and 500 different random seeds per point. The total number of devices is 254; device speeds range from 0% to 60% of the connection radius per round, and crash probability per round ranges from 0% to 6%.

Mo, Y., S. Dasgupta & J. Beal (2019). 'Robustness of the adaptive Bellman-Ford algorithm: Global stability and ultimate bounds'. *IEEE Transactions on Automatic Control* **64**(10), 4121– 4136.

Olfati-Saber, R., J. A. Fax & R. M. Murray (2007). 'Consensus and cooperation in networked multi-agent systems'. *Proceedings of the IEEE* **95**(1), 215–233.

Peleg, D. (1990). 'Time-optimal leader election in general networks'. *Journal of Parallel and Distributed Computing* **8**(1), 96–99.

Pianini, D., J. Beal & M. Viroli (2016). Improving gossip dynamics through overlapping replicates. In 'International Conference on Coordination Languages and Models'. Springer. pp. 192–207.

Summers, T., C. Yu, B. D. O. Anderson & S. Dasgupta (2009). Formation shape control: Global asymptotic stability of a four-agent formation. In 'IEEE CDC'. pp. 3002–3007.

Summers, T., C. Yu, S. Dasgupta & B. D. O. Anderson (2011). 'Control of minimally persistent leader-remote-follower and coleader formations in the plane'. *IEEE Transactions on Automatic Control* **56**(12), 2778–2792.

Viroli, M., G. Audrito, J. Beal, F. Damiani & D. Pianini (2018). 'Engineering resilient collective adaptive systems by self-stabilisation'. *ACM Transactions on Modelling and Computer Simulation (TOMACS)* **28**(2), 16:1–16:28.

Yu, C., B. D. O. Anderson, S. Mou, J. Liu, F. He & A. S. Morse (2017). 'Distributed averaging using periodic gossiping'. *IEEE Transactions on Automatic Control* **62**(8), 4151 – 4158.
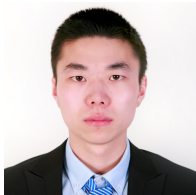
Yuanqiu Mo was born in Yangzhou, China, in 1991. He received the Ph.D. degree in electrical and computer engineering at the University of Iowa, in 2019. He is currently an associate researcher with the Southeast University. His research interests include distributed algorithm design and stability theory. Dr. Mo has been awarded the 2018 CDC Outstanding Student Paper Award. He was also a finalist of the Young Author Award in the IFAC World Congress 2020.

Giorgio Audrito, Ph.D. in Mathematics from the University of Torino, is a research fellow at the Computer Science Department of the same university. His research interests include distributed computing, programming languages, distributed algorithms and graph algorithms. He leads the development of FCPP, a C++-based implementation of the field calculus. Home page: http://giorgio.audrito.info/#!/research.

Soura Dasgupta received the B.E. degree (Hons. I) in Electrical Engineering from the University of Queensland (Australia) in 1980, and the Ph.D. in Systems Engineering from the Australian National University, in 1985. He is currently F. Wendell Miller Distinguished Professor in the Department of Electrical and Computer Engineering at the University of Iowa, U.S.A.

In 1981, he was a Junior Research Fellow at the Indian Statistical Institute, Calcutta. He has held visiting appointments at the University of Notre Dame, University of Iowa, Universite Catholique de Louvain-La-Neuve, Belgium, Tata Consulting Services, Hyderabad, the Australian National University, Shandong Academy of Sciences and National ICT Australia.

He is a past Associate Editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, IEEE Control Systems Society Conference Editorial Board, and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS- II. He is a co-recipient of the Gullimen Cauer Award for the best paper published in the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS in the calendar years of 1990 and 1991, a past Presidential Faculty Fellow (a predecessor of PECASE), a past subject editor for the International Journal of Adaptive Control and Signal Processing, and a member of the editorial board of the EURASIP Journal of Wireless Communications. In 2012 he was awarded the University Iowa Collegiate Teaching award. In the same year he was selected by the graduating class for excellence in teaching and commitment to student success. From 2016-18 he was a 1000 Talents Scholar in the People's Republic of China. His research interests are in Controls, Signal Processing, Communications and Parkinson's Disease. He was elected a Fellow of the IEEE in 1998.

Jacob Beal is a scientist at Raytheon BBN Technologies in Cambridge, Massachusetts. His research focuses on the engineering of robust adaptive systems, particularly on the problems of aggregate-level modeling and control for spatially distributed systems like pervasive wireless networks, robotic swarms, and natural or engineered biological cells. Dr. Beal received a PhD in electrical engineering and computer science from MIT. He is an associate editor of ACM Transactions on Autonomous and Adaptive Systems, is on the steering committee of the IEEE International Conference on Self-Adapting and Self-Organizing Systems (SASO), and is a founder of the Spatial Computing Workshop series. He is a Senior Member of IEEE. Contact him at jakebeal@ieee.org