This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2022.3200466

Finite-Time Distributed Algorithms for Verifying and Ensuring Strong Connectivity of Directed Networks

Made Widhi Surya Atman, *Member, IEEE,* and Azwirman Gusrialdi, *Member, IEEE*

Abstract—The strong connectivity of a directed graph associated with the communication network topology is crucial in ensuring the convergence of many distributed estimation/control/optimization algorithms. However, the assumption on the network's strong connectivity may not always be satisfied in practice. In addition, information on the overall network topology is often not available, e.g., due to privacy concerns or geographical constraints which calls for a distributed algorithm. This paper aims to fill a crucial gap in the literature due to the absence of a fully distributed algorithm to verify and ensure in finite-time the strong connectivity of a directed network. Specifically, inspired by the maximum consensus algorithm we propose distributed algorithms that enable individual node in a networked system to verify the strong connectivity of a directed graph and further, if necessary, augment a minimum number of new links to ensure the directed graph's strong connectivity. The proposed distributed algorithms are implemented without requiring information of the overall network topology and are scalable as they only require finite storage and converge in finite number of steps. Furthermore, the algorithms also preserve the privacy in terms of the overall network's topology. Finally, the proposed distributed algorithms are demonstrated and evaluated via numerical results.

Index Terms—Distributed algorithms, finite-time, max-consensus, strongly connected digraph, weakly connected digraph, link addition.

1 Introduction

1.1 Motivation and Literature Review

Distributed algorithm plays an important role in estimation [2], [3], optimization [4], [5], and control [6], [7], [8], [9] of networked systems. In contrast to centralized algorithms where all the computations are performed at a control center, the computations in distributed algorithms are locally performed at individual system and by exchanging information with a number of neighboring systems via a communication network. As a result, distributed algorithms have several potential advantages such as scalability to system's size, robustness with respect to failure of individual system, and also preservation of data privacy. Strong connectedness of a graph associated with the communication network topology of distributed systems is a crucial requirement in ensuring the convergence of the above mentioned distributed algorithms. Most of the work on distributed estimation, optimization, and control algorithms take for granted (i.e., assume) that the communication network topology is strongly connected. However, in practice the communication network topology of a networked systems may not always be strongly connected. Therefore, it is of importance to first verify and further ensure (e.g., by adding new links) the strong connectivity of a given communication network topology before executing any distributed estimation/optimization/control algorithms. More importantly, the procedure for verifying and ensuring strong connectivity of a communication network topology also needs to be performed in a distributed manner as the overall network topology is often not available due to privacy concerns or geographical constraints and also in order to comply with the feature of distributed algorithms that will be deployed in the networked systems.

Motivated by the above fundamental yet crucial issue, this paper focuses on the problem of distributively verifying and ensuring the strong connectivity of a directed graph. The communication of many real-world distributed systems is unidirectional whose overall communication network topology can be modelled as a directed graph. For example, in a broadcast-based communication scheme or publish-subscribe protocol (as can be found in Robot Operating System for robotic systems [10] and Open Field Message Bus for smart grid [11]) the receiver/subscriber can decide to use only a portion of all the broadcasted/published information due to their selected preferences or to limit the computational and/or communication cost. Other examples of unidirectional communication include connectivity in social network such as Twitter [12] and wireless network using directional antennae [13].

The problem of verifying a strongly connected directed graph (digraph) can be translated into the problem of computing strongly connected components of a given digraph. Existing algorithms to solve the computation include Tarjan [14], [15], Kosaraju–Sharir [16], and Gabow [17], [18] algorithm, which are based on depth-first-search approach, as well as the relation-transitive-closure-based Warshall algorithm [19]. On the other hand, the problem of ensuring

- *A preliminary version of this paper was presented at the IEEE Conference on Decision and Control as [1].*
- *The work was supported by the Academy of Finland under Academy Project decision number 330073.*
- *M. W. S. Atman and A. Gusrialdi are with Faculty of Engineering and Natural Sciences, Tampere University, Tampere 33014, Finland. E-mail: widhi.atman@tuni.fi, azwirman.gusrialdi@tuni.fi*

strong connectivity of a directed graph is often described as strong connectivity augmentation problem. The study on the augmentation problem was initiated by the work in [14], [15], followed by subsequent research in [20], [21], emphasizing that the problem is solvable in polynomial time. Note that while the problem of ensuring strong-connectivity problem is equivalent to constructing k-edge-connectivity topology with $k = 1$, various approaches for $k \geq 2$ in undirected graph topology has also been gaining interest to ensure robustness of the communication network, see for example [22], [23].

Despite the aforementioned approaches in verifying and ensuring strongly connected digraph, most of the solutions focus on the centralized or parallel computation and rely on the assumption that information/knowledge of the overall network topology is available or known beforehand. A fully distributed approach (i.e., without requiring knowledge of the overall network topology) to solve the problem is still limited in literature, with notable examples are presented in [9], [24]. The distributed algorithms in [9], [24] focus on verifying strong connectivity of a digraph after link removals. However, the algorithm still requires the initial graph before link removal to be strongly connected.

1.2 Statement of Contributions

The main contributions of this paper are twofold. First, we propose distributed algorithms for verifying strong connectivity of a directed graph. The proposed algorithms are inspired by the maximum consensus algorithm [25], [26]. Our second contribution is distributed algorithms to turn a non-strongly connected digraph into a strongly connected one by adding a minimum number of new links. This is achieved by first developing distributed link addition algorithms together with their optimality gap to ensure strong connectivity of a directed graph. A distributed method is then developed to check if the number of added links is minimum and further, if necessary, compute a new set of minimum number of links to make the digraph strongly connected. In addition to be fully distributed and without requiring information of the overall network topology, the proposed distributed algorithms are also scalable as they only require finite storage and converge in finite time steps. The completion in a finite number of steps allows the proposed algorithms to be easily implemented before executing any distributed estimation/control/optimization algorithms whose convergence require strong connectedness of the underlying communication network. Furthermore, the distributed algorithms are also able to preserve the privacy in terms of the global network topology.

Finally, in comparison to the preliminary version of our work on this problem [1], this paper considers link augmentation problem for not only weakly connected digraph but also disconnected digraph. The distributed algorithms in this paper also ensure strong connectivity of a digraph with minimum number of link addition. In addition, this paper includes all the proofs omitted in the preliminary version together with extensive simulations.

1.3 Organization

The remainder of this paper is organized as follows. In Section 2, we review the basic notions from graph theory and provide the problem settings. Section 3 presents the distributive algorithm to verify whether a given directed network is strongly connected. The distributed algorithm to estimate strongly connected components of a digraph is then presented in Section 4. Section 5 presents the distributed algorithm to strongly connect a directed graph. Numerical results is presented in Section 6 and followed with concluding remarks in Section 7. All the proofs of the theorems, propositions and lemmas are presented in the Appendix A. Illustrative examples to describe the procedure of the proposed algorithms are included as a supplementary material.

2 PROBLEM FORMULATION

In this section, we recall some basic notions of the fundamental theories such as graph theory and maximum consensus algorithm. Then, we define the problem settings within this paper.

2.1 Notation and Graph Theory

Information exchange between nodes in a network can be modeled by means of *directed graph (digraph)*. A directed graph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a set of nodes $\mathcal{V} = \{1, 2, \ldots, n\}$ and a set of edges (links) $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. A graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ is a *subgraph* of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if $\mathcal{V}_1 \subseteq \mathcal{V}$ and $\mathcal{E}_1 \subseteq \mathcal{E}$. Existence of an edge $(i, j) \in \mathcal{E}$ denotes that node j can obtain information from node i, or node i is accessible to node j. Here, node i is said to be an *in-neighbor* of node j while node j is the *out-neighbor* of node i. Within this paper, the set of all in-neighbors of node i is denoted by $\mathcal{N}_i^{\text{in}} = \{j \in \mathcal{V} \mid (j, i) \in \mathcal{E}\}$ while $\mathcal{N}_i^{\text{out}} = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$ denotes the set of all out-neighbors of node i. Let the set $\mathcal{K}$ consist of all 2-element subsets of $\mathcal{V}$, then the edge set $\mathcal{E}^C := \mathcal{K} \setminus \mathcal{E}$ denotes all possible edges that are not present in $\mathcal{G}$.

A *path* is a sequence of nodes $(i_1, i_2, \ldots, i_p), p > 1$, such that i_j is an in-neighbor of i_{j+1} for $j = i_1, \ldots, p-1$. An *elementary path* is a path in which no nodes appears more than once. A path is *closed* if $i_p = i_1$. A *cycle* is a closed path such that $i_1, i_2, \ldots, i_{p-1}$ are all distinct. A graph is *acyclic* if it has no cycles. A graph is said to be *strongly connected* if there is a path between any pair of distinct nodes and it is called *weakly connected* if the graph obtained by adding an edge (j, i) for every existing edge $(i, j) \in \mathcal{E}$ in the original graph is strongly connected. A *strongly connected component* of directed graph $\mathcal{G}$ is a subgraph of $\mathcal{G}$ that is strongly connected and maximal, as such no additional edges or vertices from $\mathcal{G}$ can be included in the subgraph without breaking its property of being strongly connected.

Within this paper, let $\mathbb{R}$ be the set of real numbers and $\mathbb{Z}_{\geq 0}$ be the set of non-negative integers. By $\mathbf{1}_n \in \mathbb{R}^n$ and $\mathbf{0}_n \in \mathbb{R}^n$, we denote the all ones vector and zeros vector in n-dimension, respectively. For a given set $\mathcal{N}$, $|\mathcal{N}|$ denotes the number of elements in this set. Vectors are denoted as boldface letters and matrices are denoted as capital letters in boldface. Finally, the state associated with node $i \in \mathcal{V}$ is represented by the subscript operator, for example state $\boldsymbol{a} \in \mathbb{R}^b$, $b > 1$ for node i is shown as $\boldsymbol{a_i}$ and the j-th element of vector $\boldsymbol{a_i}$ (with $j \leq b$) is denoted by $a_{i,j}$.

2.2 Max-Consensus Algorithm

Consider a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n nodes and let us assign state $y_i[t] \in \mathbb{R}$ to each node $i \in \mathcal{V}$. The max-consensus algorithm allows all nodes to distributively compute the maximum value of the initial conditions $y_i[0]$ for all $i \in \mathcal{V}$. Specifically, each node executes the following update rule [25]

$$y_i[t+1] = \max_{j \in \mathcal{N}_i^{\text{in}} \cup \{i\}} \{y_j[t]\}, \tag{1}$$

with t denotes the t-th communication event.

Definition 1 (Max-Consensus [25]). Given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, an initial states $y_i[0]$ for each node $i \in \mathcal{V}$ and the update law (1). Then, *max-consensus* is said to be achieved, if $\exists l \in \mathbb{Z}_{\geq 0}$ such that

$$y_i[k] = \max_{i \in \mathcal{V}} \{y_i[0]\} \quad \forall k \geq l, \ \forall i \in \mathcal{V}. \tag{2}$$

If (2) holds for all possible $y_i[0]$, we say that *strong max-consensus* is achieved. If (2) only holds for a subset of all possible $y_i[0]$, *weak max-consensus* is achieved.

Next, we recall the following results.

Lemma 1 (Max-Consensus [25]). Let $\mathcal{G}$ be a directed graph representing the communication topology of n nodes.

- *Strong max-consensus:* Given any initial value of $y_i[0]$, the necessary and sufficient condition for strong max-consensus is that there exist a path between any pair of nodes in $\mathcal{G}$, i.e., the digraph $\mathcal{G}$ is strongly connected.
- *Weak max-consensus:* Given partitions of all nodes based on the initial value of $y_i[0]$ as $\mathcal{V}_m := \{i \in \mathcal{V} \mid y_i[0] = \max_{i \in \mathcal{V}} \{y_i[0]\}\}$ and $\mathcal{V}_o := \mathcal{V} \setminus \mathcal{V}_m$. Then, the necessary and sufficient condition for weak max-consensus is that for any node $j \in \mathcal{V}_o$, there exist a path ending in j and starting in a node $k \in \mathcal{V}_m$.
- *Convergence speed:* The required number of communication instants is the maximum of the shortest path length between any pair of nodes in $\mathcal{G}$, i.e. $n-1$ in the worst case.

It will be demonstrated throughout the paper that max-consensus algorithm serves as a unified framework to solve our problem.

2.3 Problem Settings

Consider a network consisting of n nodes whose connections is given by a directed graph $\mathcal{G}^0 = \{\mathcal{V}, \mathcal{E}\}$, which also represents the communication network topology between the nodes. We make the following assumptions in the remaining of the paper:

Assumption 1. Assume that

1) The information of the overall network topology $\mathcal{G}^0$ is not available and each node i only knows the information on $\mathcal{N}_i^{\text{in}}$, $\mathcal{N}_i^{\text{out}}$, and n.
2) Each node is equipped with its own computational resources and is assigned with a unique identifier which can be mapped to its vertex number, i.e., $i \in \{1, \ldots, n\}$.

Note that the unique identifier is a standard assumption commonly used in designing distributed algorithm which can be realized e.g., by using MAC address, see for example [3], [7]. In addition to Assumption 1, it is also assumed that the communication between nodes occur in a synchronous manner. Furthermore, we consider a discrete-time case, where communication instants may either be defined by a clock or by the occurrence of external events. This can be realized, e.g., by allowing the node to have access to global/universal time and by having the execution timing and interval to be predetermined beforehand.

The objective of this paper is to develop distributed algorithms, under assumption 1, for solving the following problems:

Problem 1 (Connectivity Verification). Verify in a distributed manner if directed graph $\mathcal{G}^0$ is strongly connected.

Problem 2 (Connectivity Augmentation). For a directed graph $\mathcal{G}^0$, add a minimum number of additional edges $\Delta\mathcal{E}^+ \subseteq \mathcal{E}^C$ in a distributed manner to ensure that the resulting graph $\mathcal{G}^* = \{\mathcal{V}, \mathcal{E} \cup \Delta\mathcal{E}^+\}$ is strongly connected, i.e., to solve the following optimization problem

$$\begin{aligned} \min_{\Delta\mathcal{E}^+ \subseteq \mathcal{E}^C} \quad & |\Delta\mathcal{E}^+|, \\ \text{s.t.} \quad & \mathcal{G}^* \text{ is strongly connected} \end{aligned} \tag{3}$$

For the sake of readability, the notations used in this paper are summarized in Table 1. Each notation will be described in more detail when it is first used in the discussion.

3 DISTRIBUTED VERIFICATION OF A DIRECTED GRAPH'S STRONG CONNECTIVITY

In this section, we present a distributed algorithm to verify whether a given network is strongly connected. Here, for each node $i \in \mathcal{V}$, we introduce the state $\boldsymbol{x}_i[t] \in \mathbb{R}^n$ for checking if node i is reachable from any other nodes and state $f_i[t] \in \mathbb{R}$ for locally verifying if graph $\mathcal{G}^0$ is strongly connected. Within this paper, we refer $t \in \mathbb{Z}_{\geq 0}$ as the tth communication event. To this end, each node updates each row $j \in \mathcal{V}$ of its state $\boldsymbol{x}_i[t]$, i.e., $x_{i,j}[t]$, for n iterations according to the following max-consensus protocol

$$x_{i,j}[t+1] = \max_{k \in \mathcal{N}_i^{\text{in}} \cup \{i\}} x_{k,j}[t] \tag{4}$$

whose initial condition is chosen as

$$x_{i,j}[0] = \begin{cases} 1, & \text{if } j = i \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Given the initialization in (5), this approach allows individual node to estimate the existence of paths from all other nodes to itself as the value of $x_{i,j}[n] = 1$ for any $i \neq j$ implies that there exists a path from node j to node i while the value of $x_{i,j}[n] = 0$ signals the absence of that path [9]. The n iterations is selected to ensure x_i reach its steady state.

The following result establishes the relationship between the value of $\boldsymbol{x}_i[n]$ and the strong connectivity of directed graph $\mathcal{G}^0$.

Theorem 1. Given a digraph $\mathcal{G}^0$ and each node executes (4) for n iterations whose initial values are given in (5), the

TABLE 1: A List of Key Notations

Notations	Description
$\mathcal{G}^0$	original graph to test
$\mathcal{V}$, $\mathcal{E}$	set of nodes and initial edges in $\mathcal{G}^0$
n	number of nodes, i.e., $n = \|\mathcal{V}\|$
$\Delta\mathcal{E}^+$	set of new edges to strongly connect $\mathcal{G}^0$
$\boldsymbol{x}_i[t]$	node i's estimate of reachable nodes
$f_i[t]$	node i's estimate of strong connectivity
$\boldsymbol{c}_i[t]$	node i's estimate of other nodes' information number
$\boldsymbol{o}_i[t]$	node i's estimate of other SCC's outgoing edge
ζ_i	node i's information number, i.e. 1 plus total number of reachable nodes to i
$\mathcal{C}_i$	node i's estimate of its own member of SCC
$\mathcal{P}_i$	node i's estimate of reachable nodes outside of $\mathcal{C}_i$
m	index for number of link addition iteration
$\mathcal{G}^m$	the resulting graph after m-iteration of link addition
$\mathcal{V}^m_{\text{sour}}$	representative node of source-scc in graph $\mathcal{G}^m$
$\mathcal{V}^m_{\text{sink}}$	representative node of sink-scc in graph $\mathcal{G}^m$
$\mathcal{V}^m_{\text{isol}}$	representative node of isolated-scc in graph $\mathcal{G}^m$
d^m	number of disjoint subgraphs in graph $\mathcal{G}^m$
$\mathcal{S}^m_j$	set of node within $\mathcal{V}^m_{\text{sour}}$ that is accessible to $j \in \mathcal{V}^m_{\text{sink}}$
$\overline{\mathcal{G}}^m$	a condensed graph representation for $\mathcal{G}^m$
$\overline{\mathcal{V}}^m$	set of nodes in $\overline{\mathcal{G}}^m$, i.e., $\overline{\mathcal{V}}^m = \{\mathcal{V}^m_{\text{sour}}, \mathcal{V}^m_{\text{sink}}, \mathcal{V}^m_{\text{isol}}\} \subseteq \mathcal{V}$
$\overline{\mathcal{E}}^m$	set of edges in $\overline{\mathcal{G}}^m$. Edge $(i,j) \in \overline{\mathcal{E}}^m$ denotes the existence of path from $i \in \mathcal{V}^m_{\text{sour}}$ to $j \in \mathcal{V}^m_{\text{sink}}$ in $\mathcal{G}^m$
Δ^*	optimality gap between $\|\Delta\mathcal{E}^+\|$ and the minimum number of required link to strongly connect $\mathcal{G}^0$
v	an ordering for $\mathcal{V}^m_{\text{sour}}$, $\mathcal{V}^m_{\text{isol}}$ to compute minimum link
w	an ordering for $\mathcal{V}^m_{\text{sink}}$ to compute minimum link
p	an index to compute minimum link

graph $\mathcal{G}^0$ is strongly connected if and only if $\boldsymbol{x}_i[n] = \mathbf{1}_n$ for all $i \in \mathcal{V}$.

As a last step, each node needs to verify locally whether $\boldsymbol{x}_i[n] = \mathbf{1}_n$ for all $i \in \mathcal{V}$. To this end, each node updates its state $f_i[t]$ for n iterations according to

$$f_i[t+1] = \max_{j \in \mathcal{N}_i^{\text{in}} \cup \{i\}} f_j[t] \tag{6}$$

whose initial value is chosen as

$$f_i[0] = \begin{cases} 0, & \text{if } \boldsymbol{x}_i[n] = \mathbf{1}_n \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

Each node can then independently verify the strong connectivity of digraph $\mathcal{G}^0$ by observing its own value of $f_i[n]$ as shown in the following theorem.

Theorem 2. Given a digraph $\mathcal{G}^0$ and each node executes in sequence update rule (4) and (6) for n iterations each, with each initial values as in (5) and (7). The graph $\mathcal{G}^0$ is strongly connected if and only if $f_i[n] = 0$ for any $i \in \mathcal{V}$.

The pseudo code of distributed verification algorithm for solving problem 1 is summarized in Algorithm 1.

Remark 1 (Computational Complexity). Algorithm 1 finishes in $2n$ iterations i.e., its computational complexity is equal to $O(n)$.

Remark 2 (Privacy Preservation). From the retrieved information through Algorithm 1, each node only knows the existence of path from other nodes to itself (state $\boldsymbol{x}_i$) and the general notion of the strong connectivity of the graph $\mathcal{G}^0$ (state f_i). Thus, Algorithm 1 does not reveal the overall network topology.

Algorithm 1 Distributed Algorithm for Solving Problem 1

Input: network size n, in-neighbor set $\mathcal{N}_i^{\text{in}}$
Output: verification if $\mathcal{G}^0$ is strongly connected
1: initialize each row of $\boldsymbol{x}_i[0]$ as in (5)
2: for each j-th row of $\boldsymbol{x}_i$ ($j \in \{1, \ldots, n\}$), execute max-consensus update law (4) for n iterations.
3: assign $f_i[0]$ as in (7)
4: execute max-consensus update law (6) for n iterations
5: node i knows that graph $\mathcal{G}^0$ is strongly connected when $f_i[n] = 0$ and not strongly connected when $f_i[n] = 1$.

Now, assume that after running Algorithm 1 all nodes verify that the graph $\mathcal{G}^0$ is not strongly connected, i.e., $\mathcal{G}^0$ is either a weakly connected or a disconnected digraph. A distributed algorithm is then needed to add new edges to $\mathcal{G}^0$ so that the resulting graph becomes strongly connected. The problem can be reduced to a simpler one by converting $\mathcal{G}^0$ into a directed acyclic graph $\hat{\mathcal{G}}^0$ which contains one node for each strongly connected component (SCC) of $\mathcal{G}^0$. The resulting node in $\hat{\mathcal{G}}^0$ with no entering edge is called a source, and a node with no exiting edge is called a sink. The new edges to strongly connect $\hat{\mathcal{G}}^0$ can then be selected by connecting the existing sink to source following a certain ordering, as shown in [14], [15]. However, the computation for the solution in general is centralized which requires information of the overall network topology. In the following sections, given a non-strongly connected digraph we propose distributed algorithms which first estimate the strongly connected components that each node belongs to (Section 4) and then distributively add new links to make the digraph strongly connected (Section 5).

4 DISTRIBUTED ESTIMATION OF SCC

In the following, inspired by the max-consensus algorithm we propose distributive approaches for estimating the strongly connected component (SCC) of a directed graph. First, let us introduce the following definitions on different types of SCC.

Definition 2 (source-scc). *source strongly connected component* is a strongly connected component with no entering edges and one or more exiting edges.

Definition 3 (sink-scc). *sink strongly connected component* is a strongly connected component with no exiting edges and one or more entering edges.

Definition 4 (isolated-scc). *isolated strongly connected component* is a strongly connected component with no exiting edges and no entering edges.

An illustration of source-sccs, sink-sccs, and isolated-sccs is shown in Fig. 1. Note that a SCC which is neither sink-scc, source-scc, or isolated-scc can also exist (called as non-assigned SCC) within a directed graph, e.g., nodes 9 and 10 in Fig. 1.

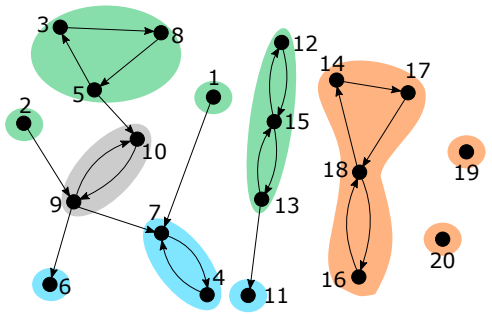


Fig. 1: Examples of source-sccs (green regions), sink-sccs (blue regions), isolated-sccs (orange regions), and non-assigned SCC (gray region)

The proposed distributed algorithms allow each node $i \in \mathcal{V}$ to estimate the following: 1) the existence of paths from other nodes to itself; 2) the SCC that it belongs to, namely the set $\mathcal{C}_i$; 3) the existence of entering or exiting edges of its own SCC; and 4) verify whether its own SCC is a source-scc, sink-scc, isolated-scc, or neither of these.

To that end, for each node $i \in \mathcal{V}$, let us assign states $\boldsymbol{x}_i[t] \in \mathbb{R}^n$, $\boldsymbol{c}_i[t] \in \mathbb{R}^n$, and $\boldsymbol{o}_i[t] \in \mathbb{R}^n$. State $\boldsymbol{x}_i[t]$ is used to check if node i is reachable from any other nodes. State $\boldsymbol{c}_i[t]$ then collect all accessible $\boldsymbol{x}_j[n]$ from other nodes for determining the set $\mathcal{C}_i$. Using the information on $\boldsymbol{c}_i[n]$, each node i also determines an additional set $\mathcal{P}_i$ consisting of all nodes which are reachable to $\mathcal{C}_i$. The determination of entering edge into $\mathcal{C}_i$ will rely on $\mathcal{P}_i$. Then, states $\boldsymbol{o}_i[t]$ estimates the exiting edges from $\mathcal{C}_i$. Finally, the characterization of its own SCC into source-scc, sink-scc, or isolated-scc will rely on the information $\mathcal{C}_i$, $\mathcal{P}_i$ and $\boldsymbol{o}_i[n]$ values.

4.1 Estimation of Paths and SCCs

As the first step, each node updates its state $\boldsymbol{x}_i[t]$ for n iterations according to the update rule (4) whose initial condition is chosen as in (5). Next, let us define the *information number* of node i, denoted as ζ_i, as the number of nodes that can reach node i, including node i itself. Noting that the existence of a path from node j to i is indicated by the value $x_{i,j}[n] = 1$, node i's information number is then equal to $\zeta_i = \mathbf{1}_n^T \boldsymbol{x}_i[n]$. In order to estimate the information number of other nodes which can reach node i, each node updates for n iterations each row $j \in \mathcal{V}$ of its own state $\boldsymbol{c}_i[t]$, i.e., $c_{i,j}[t]$, according to the following rule

$$c_{i,j}[t+1] = \max_{k \in \mathcal{N}_i^{\text{in}} \cup \{i\}} c_{k,j}[t] \tag{8}$$

whose initial condition is chosen as

$$c_{i,j}[0] = \begin{cases} \zeta_i, & \text{if } j = i \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

After n iterations, the information number of all nodes j that can reach node i will be given by the entry of $c_{i,j}[n]$.

We then have the following results on the information number:

Lemma 2. If node i is reachable from node j (i.e., $c_{i,j}(n) > 0$) and nodes i and j have the same information number (i.e., $c_{i,j}(n) = \zeta_i$), then nodes i and j are belonging to the same SCC (i.e., they are mutually reachable to each other).

Algorithm 2 Alternative Distributed Algorithm for Solving Problem 1

Input: directed graph $\mathcal{G}^0$, network size n, in-neighbor set $\mathcal{N}_i^{\text{in}}$

Output: verification whether graph $\mathcal{G}^0$ is strongly connected

1: initialize each row of $\boldsymbol{x}_i[0]$ as in (5)
2: for each j-th row of $\boldsymbol{x}_i$ ($j \in \{1, \dots, n\}$), execute max-consensus update law (4) for n iterations.
3: initialize each row of $\boldsymbol{c}_i[0]$ as in (9)
4: for each j-th row of $\boldsymbol{c}_i$ ($j \in \{1, \dots, n\}$), execute max-consensus update law (8) for n iterations
5: node i knows that graph $\mathcal{G}^0$ is strongly connected when $\boldsymbol{c}_i[n] = n\mathbf{1}_n$.

Lemma 3. For each node i, the other nodes in the set $\mathcal{P}_i$ have a smaller (positive) information number compared to node i (equivalently any nodes in $\mathcal{C}_i$). Specifically, the information number of node i satisfy $\zeta_i \geq |\mathcal{C}_i| + \max_{j \in \mathcal{P}_i} \zeta_j$.

As a direct result of Lemma 3, it is clear that within all the entries of $\boldsymbol{c}_i[n]$, its i-th element $c_{i,i}[n] = \zeta_i$ always has the highest number. Additionally, from Lemma 2 node i can estimate its own SCC, i.e., set $\mathcal{C}_i$, by identifying all nodes which have the same information number with itself, namely

$$\mathcal{C}_i := \{\forall j \in \mathcal{V} \mid c_{i,j}[n] = c_{i,i}[n]\}. \tag{10}$$

Furthermore, each node i can estimate the set $\mathcal{P}_i$ by collecting all nodes which have lower information number than itself, that is

$$\mathcal{P}_i := \{\forall j \in \mathcal{V} \mid 0 < c_{i,j}[n] < c_{i,i}[n]\}. \tag{11}$$

Here, $c_{i,j}[n] = 0$ represents the case where node j's information is inaccessible to i. Note that the node i's local estimation of $\mathcal{C}_i$ and $\mathcal{P}_i$ are identical to all the other nodes which belong to the same SCC (i.e,. $\mathcal{C}_j = \mathcal{C}_i$ and $\mathcal{P}_j = \mathcal{P}_i$ for all $j \in \mathcal{C}_i$).

It is easy to observe that the only SCC of a strongly connected graph is the graph itself. In fact, using this observation we can develop an alternative distributed algorithm to solve Problem 1 in which each node distributively checks the membership of its own SCC and verifies if it comprises of all nodes, i.e. $\mathcal{V}$, as shown in the following corollary.

Corollary 1. Given a digraph $\mathcal{G}^0$ and each node executes in sequence the update laws (4) and (8) for n iterations each, with initial conditions given in (5) and (9). Then, $\mathcal{G}^0$ is strongly connected if and only if for any $i \in \mathcal{V}$, $\boldsymbol{c}_i[n] = n\mathbf{1}_n$ (equivalent to $|\mathcal{C}_i| = n$ and $\mathcal{C}_i = \mathcal{V}$).

The pseudo code of alternative distributed verification algorithm for solving problem 1 is summarized in Algorithm 2.

4.2 Determination of Sink-scc, Source-scc, and Isolated-scc

Using Algorithm 2, node i can estimate the existence of paths from other nodes to itself and the SCC that it belongs

to, namely the set $\mathcal{C}_i$. In order to provide an effective strong connectivity augmentation which will be described later, it is important that each node is also able to characterize whether its own SCC is a source-scc, sink-scc, or isolated-scc. For this purpose, each node needs to identify the existence of entering or exiting edges of its own SCC. To that end, we introduce the following lemma.

Lemma 4. A SCC has no entering edges if and only if $\mathcal{P}_i = \emptyset$ for all node i in its membership.

With the estimated value of $\mathcal{P}_i$, each node i can determine the absence of an entering edge to its own SCC (i.e., set $\mathcal{C}_i$) based on Lemma 4, namely when $\mathcal{P}_i = \emptyset$.

On the other hand, in order to verify if there exists an edge from nodes i in $\mathcal{C}_i$ to any nodes $j \notin \mathcal{C}_i$, each node updates for n iterations each row $j \in \mathcal{V}$ of its state $\boldsymbol{o}_i[t]$, i.e., $o_{i,j}[t]$, according to the following rule

$$o_{i,j}[t+1] = \max_{k \in \mathcal{N}_i^{\text{in}} \cup \{i\}} o_{k,j}[t] \tag{12}$$

whose initial condition is chosen as

$$o_{i,j}[0] = \begin{cases} 1, & \text{if } j = i \text{ and } \exists k \in \mathcal{N}_i^{\text{out}} \ (k \notin \mathcal{C}_i) \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

In other words, the state $\boldsymbol{o}_i[n]$ collects the information from all nodes $k \in \mathcal{P}_i \cup \mathcal{C}_i$ on whether there exists an edge from node k to any nodes outside of its set $\mathcal{C}_k$.

We can then establish the following result which allows each node to distributively characterize its own SCC.

Proposition 1. Given a digraph $\mathcal{G}^0$ and each node executes in sequence the update rules (4), (8), and (12) for n iterations each, with initial values given in (5), (9), and (13), respectively. Node i can then determine the following to characterize its own SCC (i.e. the set $\mathcal{C}_i$):

1) All nodes in the set $\mathcal{C}_i$ is a source-scc if and only if $\mathcal{P}_i = \emptyset$ and there exist a node $j \in \mathcal{C}_i$ where $o_{i,j}[n] = 1$.
2) All nodes in the set $\mathcal{C}_i$ is a sink-scc if and only if $\mathcal{P}_i \neq \emptyset$ and $o_{i,j}[n] = 0, \ \forall j \in \mathcal{C}_i$.
3) All nodes in the set $\mathcal{C}_i$ is an isolated-scc if and only if $\mathcal{P}_i = \emptyset$ and $o_{i,j}[n] = 0, \ \forall j \in \mathcal{C}_i$.

Note that a non-assigned SCC will fall outside of the conditions 1)–3) in Proposition 1, i.e., $\mathcal{P}_i \neq \emptyset$ and there exist a node $j \in \mathcal{C}_i$ where $o_{i,j}[n] = 1$. The pseudo code for the proposed distributed estimation and characterization of SCC is presented in Algorithm 3.

Remark 3 (Computational Complexity). Algorithm 2 and 3 finishes in $2n$ and $3n$ iterations, respectively. Thus, both algorithms' computational complexity are equal to $O(n)$.

Remark 4 (Privacy Preservation). Using the information retrieved via Algorithm 2 and 3, each node only knows the existence of path from other nodes to itself (state $\boldsymbol{x}_i$), the information number of other nodes (state $\boldsymbol{c}_i$), and the other SCC's information regarding their exiting edges (state $\boldsymbol{o}_i$). Therefore, Algorithm 2 and 3 does not reveal the overall network topology.

Remark 5. In the case where there is no disjoint subgraphs in the directed graph $\mathcal{G}^0$, it is sufficient to know only the information on the upper bound of the number of nodes in the network (denoted by $\overline{n}$) for executing Algorithm 3. This is due to the fact that the state $\boldsymbol{x}_i$, $\boldsymbol{c}_i$, and $\boldsymbol{o}_i$ are reaching steady state at time step $t = n \leq \overline{n}$. Moreover, each node i can verify strong connectivity of the digraph by checking whether its own SCC is an isolated-scc, namely property 3) in Proposition 1.

Algorithm 3 Distributed Estimation and Characterization of SCC

Input: directed graph $\mathcal{G}^0$, network size n, neighbor set $\mathcal{N}_i^{\text{in}}$ and $\mathcal{N}_i^{\text{out}}$

Output: node i's associated SCC

1: step 1-4 in Algorithm 2
2: estimate $\mathcal{C}_i$ and $\mathcal{P}_i$ by (10) and (11), respectively
3: initialize each row of $\boldsymbol{o}_i[0]$ as in (13)
4: for each j-th row of $\boldsymbol{o}_i$ ($j \in \{1, \ldots, n\}$), execute max-consensus update law and (12) for n iterations
5: node i can determine whether $\mathcal{C}_i$ is a source-scc, sink-scc, isolated-scc or neither (Proposition 1).

5 DISTRIBUTED AUGMENTATION FOR A DIRECTED GRAPH'S STRONG CONNECTIVITY

In this section, we focus our discussion on the distributed strategies to solve Problem 2. We first propose a distributed algorithm together with its optimality gap in order to add new edges to a non-strongly connected directed graph $\mathcal{G}^0$ so that the resulting graph becomes strongly connected. Then, inspired by the centralized approach in [14], [15], we propose an algorithm to verify in a distributed manner whether the number of added edges is minimum and alternatively provide a solution for the minimum link addition problem. All the computations are performed in a distributed manner and without requiring information of the overall network topology $\mathcal{G}^0$. We start by introducing the following additional assumption.

Assumption 2. Each node can establish a communication link to any node in $\mathcal{G}^0$.

This assumption can be satisfied for the publish-subscribe protocol as found in Open Field Message Bus and in social network such as Twitter where a node can request a connection to any other nodes.

In order to simplify the discussion and presentation of the proposed algorithms, in the remaining of the section each sink-scc, source-scc, and isolated-scc is represented by a single node which is a member of their own SCC. To this end, let us denote $\mathcal{G}^m$ as the resulting graph after the m-iteration of link-addition. Let us define $\mathcal{V}_{\text{sour}}^m$, $\mathcal{V}_{\text{sink}}^m$, and $\mathcal{V}_{\text{isol}}^m$, as a set consisting of representative nodes respectively for source-scc, sink-scc, and isolated-scc in $\mathcal{G}^m$. Furthermore, let $\mathcal{S}_j^m$ denote the set of all the source-scc representative nodes accessible to representative node $j \in \mathcal{V}_{\text{sink}}^m$.

A condensed graph representation of a digraph $\mathcal{G}^m$ is then given by $\overline{\mathcal{G}}^m := \{\overline{\mathcal{V}}^m, \overline{\mathcal{E}}^m\}$ with $\overline{\mathcal{V}}^m = \{\mathcal{V}_{\text{sour}}^m, \mathcal{V}_{\text{sink}}^m, \mathcal{V}_{\text{isol}}^m\} \subseteq \mathcal{V}$ and $(i,j) \in \overline{\mathcal{E}}^m$ denotes the existence of path from node i to node j in the original graph $\mathcal{G}^m$. Note that all nodes within non-assigned SCC, together with non-representative nodes within source-scc, sink-scc, and isolated-scc, will not have special role during the distributed

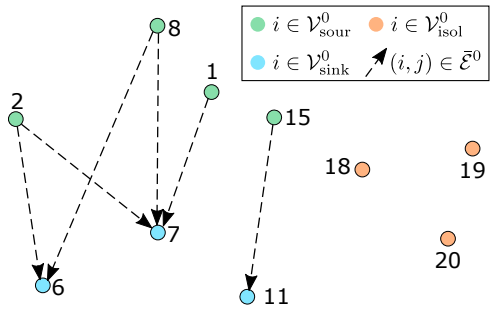


Fig. 2: A condensed graph representation $\bar{\mathcal{G}}^0$ for the digraph in Fig. 1. The nodes within non-assigned SCCs and non-representative nodes are omitted in the graph $\bar{\mathcal{G}}^0$. The graph $\bar{\mathcal{G}}^0$ is composed of 5 disjoint subgraphs, i.e., $d^0 = 5$, which are $\{1, 2, 6, 7, 8\}$, $\{11, 15\}$, $\{18\}$, $\{19\}$, and $\{20\}$.

link addition other than passing the information, hence they are omitted for the condensed graph representation.

To this end, the representative nodes can be selected by following a predefined rules, e.g., the node with the highest vertex (ID) number in each SCC is selected as the representative node. Alternatively, the nodes within the same SCC can locally coordinate over a certain decision variable, e.g., to select a node with the most number of out-neighbors, each node can share its own $\mathcal{N}_i^{\text{out}}$ and execute a max-consensus algorithm. For the above two examples, the selection of representative nodes will take no more than n iteration.

Moreover, we consider the representative nodes after each link addition, i.e., $\overline{\mathcal{V}}^m$, to be selected within $\overline{\mathcal{V}}^0$. To be precise, the selection of the representative node ensures that $\mathcal{V}^m_{\text{sour}} \subseteq \mathcal{V}^0_{\text{sour}} \cup \mathcal{V}^0_{\text{isol}}$, $\mathcal{V}^m_{\text{sink}} \subseteq \mathcal{V}^0_{\text{sink}} \cup \mathcal{V}^0_{\text{isol}}$, and $\mathcal{V}^m_{\text{isol}} \subseteq \mathcal{V}^0_{\text{sink}} \cup \mathcal{V}^0_{\text{isol}}$ are maintained. Additionally let us denote d^m as the number of disjoint subgraphs within $\mathcal{G}^m$. An example of this condensed graph is illustrated in Fig. 2. Note that the condensed graph information is introduced only for facilitating the discussion, and not necessarily known by each node in the original graph.

5.1 Distributed Link Addition Algorithm

Here, we present the algorithm to strongly connects $\mathcal{G}^0$ by utilizing the estimated SCCs obtained from the previous section. Recall that each node can use Algorithm 3 to estimate whether its own SCC is a source-scc, sink-scc, isolated-scc, or neither of these. Let us further assume that the procedure to select representative nodes for all SCCs have been established, and as a result we can present the discussion in terms of the condensed graph $\bar{\mathcal{G}}^m$.

To this end, the proposed algorithm will rely on the approach where each node $i \in \mathcal{V}^m_{\text{sour}}$ broadcasts its information to the rest of the network and accordingly each node $j \in \mathcal{V}^m_{\text{sink}}$ collects this information. This information broadcasting enables each sink-scc representative j to obtain the information about all the accessible source-scc representative $\mathcal{S}^m_j \subseteq \mathcal{V}^m_{\text{sour}}$. The broadcast of information can be distributively realized via another max-consensus update law which takes as many as n time-steps, that is by introducing a state $s_i[t] \in \mathcal{R}^n$ and initializing its element as $s_{i,i}[0] = 1$ if $i \in \mathcal{V}^0_{\text{sour}}$ and $s_{i,j}[0] = 0$ for $j \neq i$.

Algorithm 4 Distributed Algorithm to Strongly Connect A Weakly Connected Digraph

Input: weakly connected graph $\mathcal{G}^0$, network size n, neighbor set $\mathcal{N}_i^{\text{in}}$ and $\mathcal{N}_i^{\text{out}}$
Output: strongly connected graph $\mathcal{G}^m = \{\mathcal{V}, \mathcal{E} \cup \Delta\mathcal{E}^+\}$

```
set m = 0 and run Algorithm 3 {for G^m = {V, E}}
if G^m not strongly connected, i.e. C_i ≠ V then
    determine SCC's representative node within C_i
    m = m + 1
    if i ∈ V^{m-1}_sour then
        broadcast its own information
    else if i ∈ V^{m-1}_sink then
        S_i^{m-1} = ∅ and start collecting source information
    end if
    forward broadcast information for n iterations
    if i ∈ V^{m-1}_sink then
        add all members of S_i^{m-1} into N_i^out and establish
        new link from node i to all j ∈ S_i^{m-1}. {at the same
        time i is added into N_j^in and conceptually (i, j) is
        added to ΔE^+}
    end if
end if
```

5.1.1 *Distributed Algorithm for Weakly Connected Graph*

We first consider the case where the non-strongly connected digraph is given by a weakly connected digraph which has no isolated-sccs, i.e., $\mathcal{V}^0_{\text{isol}} = \emptyset$. Before proceeding, we introduce the following lemma.

Lemma 5. Given a weakly connected graph $\mathcal{G}^0$, adding edges (j, i) from each node $j \in \mathcal{V}^0_{\text{sink}}$ to all reachable nodes $i \in \mathcal{S}^0_j$, results in a strongly connected graph.

The above lemma provides a one-step strategy to strongly connect a weakly connected digraph, namely by adding a set of edges from each $j \in \mathcal{V}^0_{\text{sink}}$ to all reachable $i \in \mathcal{S}^0_j$. The pseudo code of the proposed algorithm is given in Algorithm 4. Next, let Δ^* denote the optimality gap between the added edges using Algorithm 4, denoted by $|\Delta\mathcal{E}^+|$ and the minimum number of required links to strongly connect the graph. We then have the following main result.

Theorem 3. Given a weakly connected digraph $\mathcal{G}^0 = \{\mathcal{V}, \mathcal{E}\}$, Algorithm 4 results in a strongly connected graph $\mathcal{G}^m = \{\mathcal{V}, \mathcal{E} \cup \Delta\mathcal{E}^+\}$. Furthermore, Algorithm 4 will finish in $5n$ iterations with one link-addition step ($m = 1$), whose optimality gap Δ^* is equal to

$$\Delta^* = |\Delta\mathcal{E}^+| - \max\left\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\right\} \tag{14}$$

where $|\Delta\mathcal{E}^+| = \sum_{i \in \mathcal{V}^0_{\text{sink}}} |\mathcal{S}^0_i| \leq |\mathcal{V}^0_{\text{sour}}||\mathcal{V}^0_{\text{sink}}|$.

Note that the resulting $|\Delta\mathcal{E}^+|$ also denotes the total number of elementary paths from any pair source-scc to sink-scc that exists in the initial graph. Furthermore, the following corollary shows a case where Algorithm 4 results in a minimum link addition.

Corollary 2. For a weakly connected digraph $\mathcal{G}^0$ with a single source-scc ($|\mathcal{V}^0_{\text{sour}}| = 1$) or a single sink-scc ($|\mathcal{V}^0_{\text{sink}}| = 1$), Algorithm 4 yields an optimal solution with minimum link addition.

Algorithm 5 Distributed Algorithm to Strongly Connect A Disconnected Digraph

Input: directed graph $\mathcal{G}^0$, network size n, neighbor set $\mathcal{N}_i^{\text{in}}$ and $\mathcal{N}_i^{\text{out}}$
Output: strongly connected graph $\mathcal{G}^m = \{\mathcal{V}, \mathcal{E} \cup \Delta\mathcal{E}^+\}$
1: set $m = 0$ and run Algorithm 3 {for $\mathcal{G}^m = \{\mathcal{V}, \mathcal{E}\}$}
2: **while** $\mathcal{G}^m$ is not strongly connected, i.e. $\mathcal{C}_i \neq \mathcal{V}$ **do**
3: determine representative node within $\mathcal{C}_i$
4: $m = m + 1$
5: run step 5-10 in Algorithm 4 {Broadcast source information}
6: **if** $i \in \mathcal{V}_{\text{isol}}^{m-1}$ **then**
7: randomly select a candidate node $j \notin \mathcal{C}_i$
8: add j into $\mathcal{N}_i^{\text{out}}$ and establish new link (i, j).
9: **else if** $i \in \mathcal{V}_{\text{sink}}^{m-1}$ **then**
10: add all members $\mathcal{S}_i^{m-1}$ into $\mathcal{N}_i^{\text{out}}$ and establish new links from node i to all $j \in \mathcal{S}_i^{m-1}$.
11: **end if**
12: run Algorithm 3 {for $\mathcal{G}^m = \{\mathcal{V}, \mathcal{E} \cup \Delta\mathcal{E}^+\}$}
13: **end while**

5.1.2 Distributed Algorithm for Disconnected Digraph

Next, we present distributed algorithm to strongly connect $\mathcal{G}^0$, given that $\mathcal{G}^0$ is a disconnected graph which separates group of nodes into several disjoint subgraphs, i.e. $d^0 > 1$. The main idea for the proposed distributed link addition algorithm comprises of two main steps (extending from ideas in Algorithm 4), namely to strongly connect each weakly-connected subgraph and to connect all disconnected subgraphs. Specifically, each link-addition step adds the following new links: (i) from each $i \in \mathcal{V}_{\text{sink}}^m$ to all $j \in \mathcal{S}_i^m$ and (ii) from each $i \in \mathcal{V}_{\text{isol}}^m$ to a random node $j \notin \mathcal{C}_i$. The pseudo-code of the distributed algorithm is given in Algorithm 5 and its performance is summarized in the following theorem.

Theorem 4. Given a disconnected digraph $\mathcal{G}^0 = \{\mathcal{V}, \mathcal{E}\}$, then Algorithm 5 results in a strongly connected graph $\mathcal{G}^m = \{\mathcal{V}, \mathcal{E} \cup \Delta\mathcal{E}^+\}$ by adding at most $(2d^0 + \sum_{i \in \mathcal{V}_{\text{sink}}^0} |\mathcal{S}_i^0|)$ new edges. Furthermore, Algorithm 5 will finish in $3n + 5nm$ iterations with the worst case $m = 2\lceil \log_2 d^0 \rceil$, whose optimality gap Δ^* is upper-bounded by

$$\Delta^* \leq 2d^0 + \sum_{i \in \mathcal{V}_{\text{sink}}^0} |\mathcal{S}_i^0| - (\max\{|\mathcal{V}_{\text{sour}}^0|, |\mathcal{V}_{\text{sink}}^0|\} + |\mathcal{V}_{\text{isol}}^0|). \tag{15}$$

Remark 6. Note that for a weakly connected digraph, the link addition procedure in Algorithm 5 is identical to Algorithm 4, i.e., $m = 1$. However, Algorithm 5 introduces additional $3n$ iterations for strong connectivity verification (Algorithm 3 in line 12), thus finishes in $8n$.

Remark 7 (Alternative Algorithm). An earlier version of algorithm is presented in [1] without the need to broadcast the source information. However, it may results in a longer computation time as the computation complexity is $O(n^2)$.

Remark 8. Analogous to Remark 5, given a weakly connected graph $\mathcal{G}^0$, the Algorithm 4 and 5 can be executed only with the information of the upper bound of number of nodes $\overline{n} \geq n$ by modifying the step 2 into checking whether $\mathcal{C}_i$ reflects an isolated-scc, namely property 3) in Proposition 1. Moreover, the exact number of nodes, i.e., n, can be inferred at the end of Algorithm 5 as $n = |\mathcal{C}_i|$.

5.2 Verifying and Enforcing Minimum Link Addition

In the previous subsection, we have presented distributed link addition algorithms to ensure a strongly connected graph. However, as summarized in Theorems 3 and 4, the resulting number of added links is not always guaranteed to be minimum. In the following, we present the procedure to verify whether the number of added links is minimum, and additionally compute a new set of edges to ensure minimum link augmentation by first removing the previously augmented edges $\Delta\mathcal{E}^+$. The computation will be conducted by a single node called a virtual leader. The virtual leader can be selected among any node $i \in \mathcal{V}_{\text{sink}}^0 \cup \mathcal{V}_{\text{isol}}^0$ where $(i, j) \in \Delta\mathcal{E}^+$ for some nodes j.

The verification of minimum link addition is conducted once the execution of Algorithm 5 is finished. The strong connectivity of the graph is required in order to collect the information for the minimum link verification algorithm as well as to solve the minimum link augmentation problem. A solution to the minimum link augmentation problem itself is presented in [14], [15], where a $\max\{|\mathcal{V}_{\text{sour}}^0|, |\mathcal{V}_{\text{sink}}^0|\} + |\mathcal{V}_{\text{sink}}^0|$ number of links can be added once the information on $\mathcal{V}_{\text{isol}}^0$, $\mathcal{V}_{\text{sour}}^0$, $\mathcal{V}_{\text{sink}}^0$, and $S_i^0, \forall i \in \mathcal{V}_{\text{sink}}^0$ are known.

Adopting the approach in [14], [15] to our current setup, the virtual leader needs to collect the following information: 1) original sinks $\mathcal{V}_{\text{sink}}^0$; 2) reachable sources S_i^0 for each $i \in \mathcal{V}_{\text{sink}}^0$; and 3) number of added links $|\Delta\mathcal{E}^+|$. Note that the set $\mathcal{V}_{\text{sour}}^0$ can be reconstructed from $\cup_{i \in \mathcal{V}_{\text{sink}}^0} S_i^0$. These information can be obtained by having all nodes $i \in \mathcal{V}_{\text{sink}}^0 \cup \mathcal{V}_{\text{isol}}^0$ to broadcast their own information and the number of links which they added. Using the above information, the virtual leader can then verify if the added link $|\Delta\mathcal{E}^+|$ is minimum. If the number of added links is not minimum, the virtual leader then constructs a new set of $\Delta\mathcal{E}^+$ which ensure the minimal link augmentation.

The procedure to compute the minimum link augmenting set, as shown in [14], [15], requires an index p and an ordering $v(1), \ldots, v(|\mathcal{V}_{\text{sour}}^0| + |\mathcal{V}_{\text{isol}}^0|)$ and $w(1), \ldots, w(|\mathcal{V}_{\text{sink}}^0|)$. The ordering w contains all nodes in $\mathcal{V}_{\text{sink}}^0$, while the ordering v contains a combination of $\mathcal{V}_{\text{sour}}^0$ and $\mathcal{V}_{\text{isol}}^0$. The index p and the orderings need to ensure the following properties:

1) there is a path from $v(i)$ to $w(i)$ for $1 \leq i \leq p$;
2) for each source $v(i)$, $p + 1 \leq i \leq |\mathcal{V}_{\text{sour}}^0|$ there is a path from $v(i)$ to some $w(j)$, $1 \leq j \leq p$; and
3) for each sink $w(j)$, $p+1 \leq j \leq |\mathcal{V}_{\text{sink}}^0|$ there is a path from some $v(i)$, $1 \leq i \leq p$ to $w(j)$.

Additionally, the ordering $v(|\mathcal{V}_{\text{sour}}^0| + 1), \ldots, v(|\mathcal{V}_{\text{sour}}^0| + |\mathcal{V}_{\text{isol}}^0|)$ contains all nodes from $\mathcal{V}_{\text{isol}}^0$. Given the existing information in the virtual leader, the ordering can be constructed by following the steps in Algorithm 6.

Lemma 6. Given a list of pairings of sinks and their reachable sources, the Algorithm 6 finds an index p and an ordering of v and w satisfying Properties 1–3.

Once the p and the ordering of v and w are known, the augmenting set can be constructed as a combination of the

$$\Delta\mathcal{E}^+ = \{(w(i), v(i+1)) \mid 1 \le i < p\} \cup \{(w(i), v(i)) \mid p+1 \le i \le \min\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\}\}$$
$$\cup \begin{cases} (w(p), \mathcal{V}^*), & \text{if } |\mathcal{V}^0_{\text{sour}}| = |\mathcal{V}^0_{\text{sink}}| > 0; \\ (w(p), w(|\mathcal{V}^0_{\text{sour}}|+1)) \cup \{(w(i), w(i+1)) \mid |\mathcal{V}^0_{\text{sour}}|+1 \le i < |\mathcal{V}^0_{\text{sink}}|\} \\ \cup\, (w(|\mathcal{V}^0_{\text{sink}}|), \mathcal{V}^*), & \text{if } |\mathcal{V}^0_{\text{sour}}| < |\mathcal{V}^0_{\text{sink}}|; \\ (w(p), v(|\mathcal{V}^0_{\text{sink}}|+1)) \cup \{(v(i), v(i+1)) \mid |\mathcal{V}^0_{\text{sink}}|+1 \le i < |\mathcal{V}^0_{\text{sour}}|\} \\ \cup\, (v(|\mathcal{V}^0_{\text{sour}}|), \mathcal{V}^*), & \text{if } |\mathcal{V}^0_{\text{sour}}| > |\mathcal{V}^0_{\text{sink}}|; \end{cases} \tag{16}$$
$$\cup \begin{cases} \{(v(i), v(i+1)) \mid (|\mathcal{V}^0_{\text{sour}}|+1) \le i < (|\mathcal{V}^0_{\text{sour}}| + |\mathcal{V}^0_{\text{isol}}|)\} \cup (v(|\mathcal{V}^0_{\text{sour}}| + |\mathcal{V}^0_{\text{isol}}|), v(1)) & \text{if } |\mathcal{V}^0_{\text{isol}}| \neq 0; \\ \emptyset & \text{if } |\mathcal{V}^0_{\text{isol}}| = 0; \end{cases}$$
$$\text{with } \mathcal{V}^* = \begin{cases} v(|\mathcal{V}^0_{\text{sour}}|+1) & \text{if } |\mathcal{V}^0_{\text{isol}}| \neq 0; \\ v(1) & \text{if } |\mathcal{V}^0_{\text{isol}}| = 0; \end{cases}.$$

Algorithm 6 Ordering for Sinks and Sources

Input: $\mathcal{V}^0_{\text{sour}}$, $\mathcal{V}^0_{\text{sink}}$, $\mathcal{V}^0_{\text{isol}}$, and $\mathcal{S}^0_i$ for all $i \in \mathcal{V}^0_{\text{sink}}$
Output: p, v and w following Properties 1–3

$p = 0$
for each $i \in \mathcal{V}^0_{\text{sink}}$ **do**
  for each $j \in \mathcal{S}^0_i$ **do**
    if $j \notin \{v(1), \ldots, v(p)\}$ **and** $i \notin \{w(1), \ldots, w(p)\}$ **then**
      $p = p + 1$
      add j to $v(p)$ and add i to $w(p)$
    end if
  end for
end for
add the remaining $\mathcal{V}^0_{\text{sour}}$ into $v(p+1), \ldots, v(|\mathcal{V}^0_{\text{sour}}|)$
add the remaining $\mathcal{V}^0_{\text{sink}}$ into $w(p+1), \ldots, w(|\mathcal{V}^0_{\text{sink}}|)$
add all $\mathcal{V}^0_{\text{isol}}$ into $v(|\mathcal{V}^0_{\text{sour}}|+1), \ldots, v(|\mathcal{V}^0_{\text{sour}}| + |\mathcal{V}^0_{\text{isol}}|)$

following edges shown in (16). Note that the formulated links in (16) is modified from the original formulation in [14], [15] to circumvent the need to flip the direction of the graph for the case of $|\mathcal{V}^0_{\text{sour}}| > |\mathcal{V}^0_{\text{sink}}|$. The information about the augmenting set can then be distributed to all nodes to reconfigure the new edges, that is by locally removing existing $\Delta\mathcal{E}^+$ and replacing it with the new one.

The complete pseudo-code of algorithm for verifying and enforcing minimum link addition is presented in Algorithm 7. The results can be formally stated in the following theorem.

Theorem 5. Consider a disconnected digraph $\mathcal{G}^0$. Given an index p, and an ordering of v and w following Properties 1-3, then the set of edges $\Delta\mathcal{E}^+$ in (16) makes the resulting graph $\mathcal{G}^* = \{\mathcal{V}, \mathcal{E} \cup \Delta\mathcal{E}^+\}$ strongly connected. In addition, the number of links added is $|\Delta\mathcal{E}^+| = \max\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\} + |\mathcal{V}^0_{\text{isol}}|$.

Remark 9 (Privacy Preservation). In addition to the existing information as stated in Remark 4, by executing Algorithm 4, 5, or 7, each node can also retrieved the information of $\mathcal{V}^m_{\text{sour}}$, $\mathcal{V}^0_{\text{sink}}$, and $\mathcal{V}^0_{\text{isol}}$ from the broadcasted information. While this information provide a general existence of paths between source-sccs and sink-sccs, it is still not sufficient for each node to reveal the overall network topology, thus preserving the privacy.

Algorithm 7 Distributed Algorithm for Solving Problem 2

Input: directed graph $\mathcal{G}^0$, network size n, neighbor set $\mathcal{N}^{\text{in}}_i$ and $\mathcal{N}^{\text{out}}_i$
Output: strongly connected graph $\mathcal{G}^m = \{\mathcal{V}, \mathcal{E} \cup \Delta\mathcal{E}^+\}$ with minimum number of $\Delta\mathcal{E}^+$

run Algorithm 5
determine virtual leader
if node $i \in \mathcal{V}^0_{\text{sink}} \cup \mathcal{V}^0_{\text{isol}}$ **then**
  if node i is not virtual leader **then**
    broadcast i, added edges, and accessible sources $\mathcal{S}^0_i$ (if applicable)
  else
    start collecting other's information
  end if
end if
forward broadcast information for n iterations
if node i is virtual leader **then**
  construct $\mathcal{V}^0_{\text{sour}}$, $\mathcal{V}^0_{\text{sink}}$, $\mathcal{V}^0_{\text{isol}}$ and $\Delta\mathcal{E}^+$
  if $|\Delta\mathcal{E}^+| > \max\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\} + |\mathcal{V}^0_{\text{isol}}|$ **then**
    construct Tarjan's ordering as Algorithm 6
    construct minimum link augmentation as in (16)
    broadcast the optimal link to reforge new $\Delta\mathcal{E}^+$
  else
    broadcast that link is already optimal
  end if
end if
save broadcasted information and forward it for n iterations
process the information, re-establish new links if previously not optimal

Remark 10 (Computational Complexity). Algorithm 7 finishes in $3n$ additional iterations from Algorithm 4 due to the broadcasting procedures, totaling to $6n + 10n(\lceil \log_2 d^0 \rceil)$ iterations. Hence, Algorithm 7 computational complexity is equal to $O(n \log n)$.

6 NUMERICAL SIMULATION

In this section, we provide numerical simulations where we test Algorithm 7 as it covers all the main functionalities presented in Algorithms 1-6. The distributed computation of Algorithm 7 including the information exchange are simulated in a single PC using

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2022.3200466

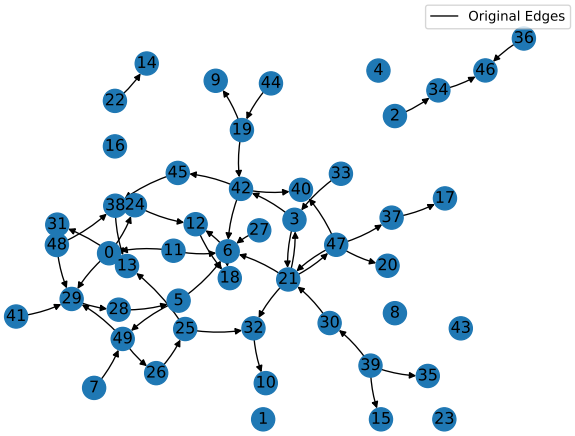


Fig. 3: A disconnected graph $\mathcal{G}_L$ with 50 nodes.

TABLE 2: Parameter of the tested graph and the theoretical bounds from Theorem 4 and 5

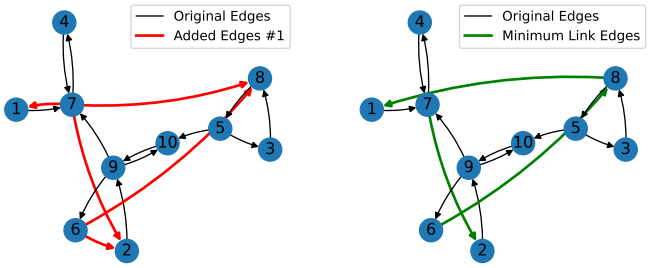
Parameter	$\mathcal{G}_B$	$\mathcal{G}_A$	$\mathcal{G}_L$
$\lvert\mathcal{V}^0_{\text{sour}}\rvert$	3	4	11
$\lvert\mathcal{V}^0_{\text{sink}}\rvert$	2	3	11
$\lvert\mathcal{V}^0_{\text{isol}}\rvert$	0	3	6
$\lvert\bar{\mathcal{E}}^0\rvert$	5	6	34
d^0	1	5	9
min required links $\lvert\Delta\mathcal{E}^+\rvert$	3	7	17
guaranteed max links $\lvert\Delta\mathcal{E}^+\rvert$	7	16	52
max optimality gap Δ^*	4	9	35
max number of link addition m	2	6	8
max time step iteration	$16n$	$36n$	$46n$

python programming language. The source code is available in the following link `https://github.com/TUNI-IINES/dist-strong-connectivity`.

For the simulations we consider three different graphs, namely $\mathcal{G}_A, \mathcal{G}_B, \mathcal{G}_L$. The graph $\mathcal{G}_A = (\mathcal{V}_A, \mathcal{E}_A)$ consists of 5 disjoint subgraphs and is shown in Fig. 1. Digraph $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{E}_B)$ is a weakly connected graph consists of 10 nodes as depicted in Fig. 4. Finally, digraph $\mathcal{G}_L$ is a disconnected graph of 50 nodes as shown in Fig. 3. The detailed parameters for each graph and the theoretical bounds presented in Theorems 4 and 5 are summarized in Table 2.

Since the distributed link addition for the weakly connected graph provides a unique solution, it is sufficient to run a single numerical simulation for $\mathcal{G}_B$. Furthermore, we conduct 400 and 2500 number of simulations for $\mathcal{G}_A$ and $\mathcal{G}_L$ respectively in order to ensure sufficient samples (n^2) are collected for verifying our theoretical results in Theorem 4 and 5, as some new links are selected randomly.

All the results of the numerical simulation show that the distributed link addition algorithms result in strongly connected digraph and if the number of added links is not minimum, the algorithms will further enforce a minimum number of added links. Hence, the results are aligned with Theorem 5. More detailed results are presented in the subsequent discussions to further verify the required number of time steps and the number of augmented link before the minimum link reconfiguration in comparison to its theoretical bound given in Theorem 4.



(a) Result of Algorithm 5 (b) Result of Algorithm 7

Fig. 4: Numerical results for graph $\mathcal{G}_B$ at the end of: a) distributed link addition algorithm and b) verification and enforcing minimum link addition algorithm.

6.1 Weakly Connected Graph $\mathcal{G}_B$ with 10 nodes

The results for graph $\mathcal{G}_B$ is illustrated in Fig. 4. The algorithm finishes in $11n$ time steps, where it first introduces 5 new edges (Fig. 4a) to strongly connect graph $\mathcal{G}_B$ before minimum link addition is enforced with 3 new edges (Fig. 4b). Note that for the weakly connected graph, the link addition procedure in Algorithm 5 is identical to the one in Algorithm 4, which ensures strong connectivity in $5n$ iteration ($m = 1$). In addition, Theorem 3 guarantee a smaller optimality gap with $\Delta^* = 2$, which is aligned with the observation shown in Fig. 4.

6.2 Disconnected Graph $\mathcal{G}_A$ with 20 nodes

An example of the results for graph $\mathcal{G}_A$ is illustrated in Fig. 5 while the results for $n^2 = 400$ repetitions are summarized in Fig. 6. For all the repetitions, the data shows that the algorithm finishes in $21n$ time steps, which is equivalent to link addition with $m = 3$ steps. The number of added links for all the repetitions are between 12 to 14 new edges. Both the number of iterations and number of augmented links are within the expected bounds as shown in Table 2.

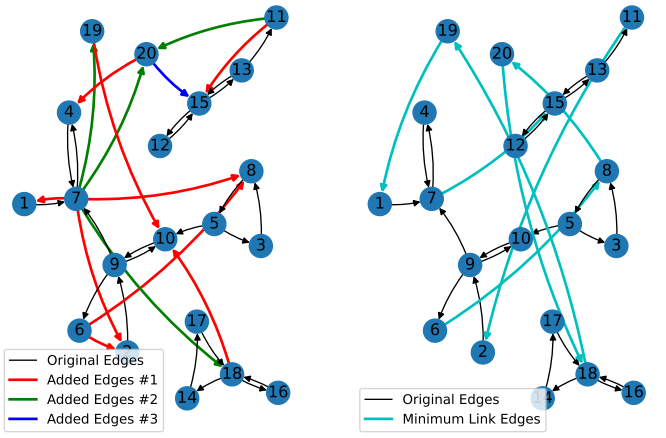
6.3 Disconnected Graph $\mathcal{G}_L$ with 50 nodes

Finally, an example of the results for graph $\mathcal{G}_L$ is illustrated in Fig. 7 while the results for $n^2 = 2500$ repetitions are summarized in Fig. 8. The data is divided into two groups, with the majority (1362 results) finishes in $26n$ time steps and the remaining (1138 results) finishes in $21n$ time steps, which are equivalent to link addition with $m = 4$ and $m = 3$ steps, respectively. The number of added links for all the results are between 46 to 50 new edges. Both the number of iterations and number of augmented links are within the expected bounds as shown in Table 2. The results verify the theoretical bounds given in Theorem 4 and 5.

7 CONCLUSIONS

This paper proposes distributed and finite time algorithms to verify strongly connected property of a directed graph and to make a directed graph strongly connected with a minimum number of link addition. The strategy is inspired by maximum consensus algorithm which is known

This article has been accepted for publication in IEEE Transactions on Network Science and Engineering. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNSE.2022.3200466



(a) Result of Algorithm 5. (b) Result of Algorithm 7

Fig. 5: An example of numerical results for graph $\mathcal{G}_A$ at the end of: a) distributed link addition algorithm and b) verification and enforcing minimum link addition algorithm.

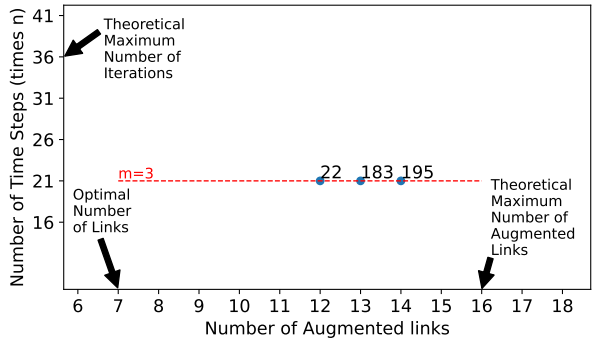
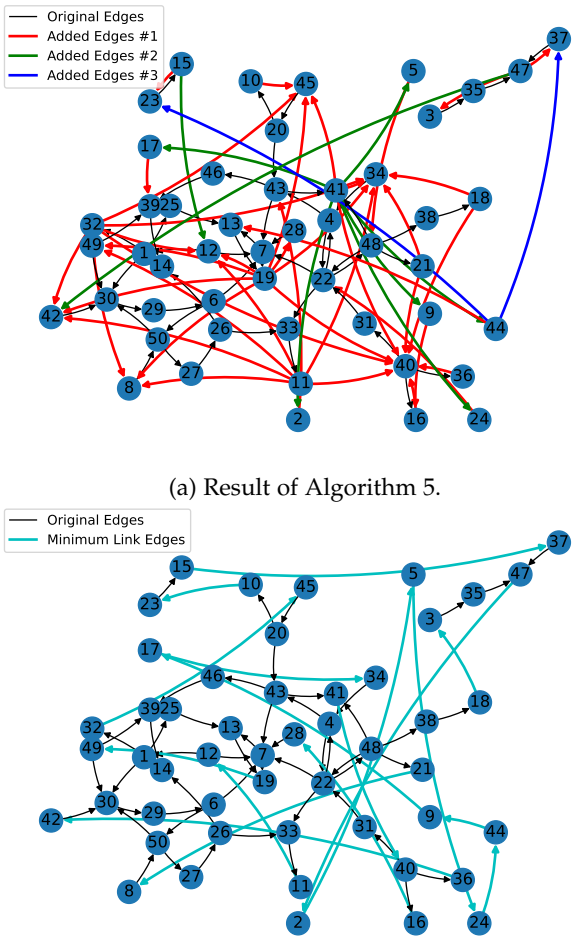


Fig. 6: Numerical results of disconnected graph $\mathcal{G}_A$ with 20 nodes for 400 repetitions. The number beside each data point describe the number of accumulated occurrences on that data point.

to have finite computation time. The proposed strategies provide the solutions without requiring knowledge of the overall network topology and further preserve the privacy in terms of the overall network's topology. Strong connectivity is a graph property that is commonly assumed or required in many distributed systems and is crucial in guaranteeing convergence of many distributed estimation/optimization/control algorithms. Hence, the proposed distributed strategy has broad applications.

Future work will aim towards the asynchronous implementation of the proposed algorithms and to relax the assumption where only upper bound of the number of nodes is known. In addition, several application specific use-cases will be considered, e.g., towards minimizing network's end-to-end delay or a case where a communication link can only established with nodes within a certain communication range.



(a) Result of Algorithm 5.

(b) Result of Algorithm 7

Fig. 7: An example of numerical results for graph $\mathcal{G}_L$ at the end of: a) distributed link addition algorithm and b) verification and enforcing minimum link addition algorithm.

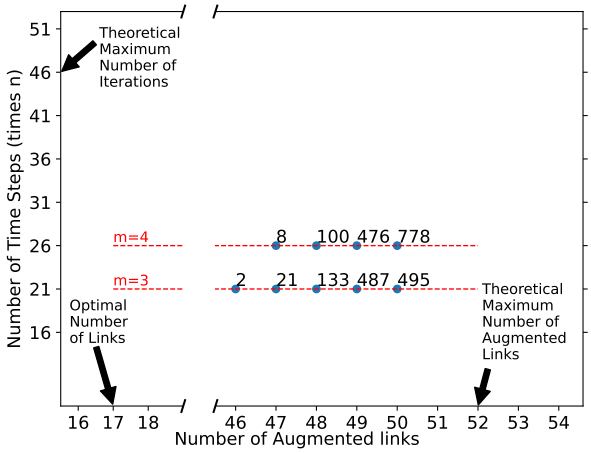


Fig. 8: Numerical results of disconnected graph $\mathcal{G}_L$ with 50 nodes for 2500 repetitions. The number beside each data point describe the number of accumulated occurrences on that data point.

APPENDIX A
PROOFS

A.1 Proof of Theorem 1

We start by showing the necessity ($\Rightarrow$). From Lemma 1, since the graph $\mathcal{G}^0$ is strongly connected, each element in x_i namely $x_{i,j}$ will converge to $\max_i x_{i,j}[0] = 1$ (strong max-consensus) for all $i, j \in \mathcal{V}$ within the worst-case of $n-1$ iterations. Thus, $\boldsymbol{x}_i[n] = \mathbf{1}_n$ is fulfilled for all $i \in \mathcal{V}$. Next, we show the sufficiency ($\Leftarrow$) through contradiction. We first assume that graph $\mathcal{G}^0$ is not strongly connected, i.e., there exists no path from a certain node i to j. However, as we have $x_{i,j}[n] = 1$ under update law (4) for all j-th row in $\boldsymbol{x}_i[n]$ and for all nodes i in the network, this means that there exist path from any node j to any node i. Hence the graph $\mathcal{G}^0$ is strongly connected, which contradicts the assumption.

A.2 Proof of Theorem 2

Let us divide all nodes into set $\mathcal{V}_0 := \{\forall i \in \mathcal{V} \mid f_i[0] = 0\}$ and $\mathcal{V}_1 := \{\forall i \in \mathcal{V} \mid f_i[0] = 1\}$. Then, we can rewrite Theorem 1 as graph $\mathcal{G}^0$ is strongly connected if and only if $\mathcal{V}_0 = \mathcal{V}$ and $\mathcal{V}_1 = \emptyset$, equivalently $f_i[n] = 0,\ \forall i \in \mathcal{V}$.

For a non-strongly connected graph $\mathcal{G}^0$, under update law (6), the value of f_i will converge to $\max_i f_i[0] = 1,\ \forall i \in \mathcal{V}$ (weak maximum consensus) if for any node $i \in \mathcal{V}_0$ there exists path ending in i and starting in $j \in \mathcal{V}_1$ [25]. Note that this condition is satisfied as any node $i \in \mathcal{V}_0$ is reachable from all nodes. This ensures that $f_i[n] = f_j[n], \forall i, j \in \mathcal{V}$.

A.3 Proof of Lemma 2

As there exist a path between any distinct nodes within a SCC, this means that all information from one node can reach the other, which results in an equal information number.

A.4 Proof of Lemma 3

Node i can be reached by all nodes in $\mathcal{P}_i$ as well as its own SCC, i.e. $\mathcal{C}_i$, thus ensures a higher information number than all nodes in $\mathcal{P}_i$. Hence, node i's information number is lower bounded by $\max_{j \in \mathcal{P}_i} |\mathcal{C}_i| + \zeta_j$, noting that node i's SCC can have multiple entering edges.

A.5 Proof of Corollary 1

We start by showing the necessity ($\Rightarrow$). Since the graph $\mathcal{G}^0$ is strongly connected, Theorem 1 ensures that $\boldsymbol{x}_i[n] = \mathbf{1}_n$ for all $i \in \mathcal{V}$. Hence, all node i's information number is equal to n, initializing $c_{i,i}[0] = n$. Strong connectivity of $\mathcal{G}^0$ and update law (8) ensures maximum consensus protocol [25] for each element in c_i namely $c_{i,j}$ will converge to $\max_i c_{i,j}[0] = n$ for all $i, j \in \mathcal{V}$. Thus, $\boldsymbol{c}_i[n] = n\mathbf{1}_n$. Note that with (10), the above condition is equivalent to each node i ends up with $\mathcal{C}_i = \mathcal{V}$ (alternatively $|\mathcal{C}_i| = |\mathcal{V}| = n$) for all $i, j \in \mathcal{V}$. The sufficiency ($\Leftarrow$) through contradiction follows similar arguments with Theorem 1.

A.6 Proof of Lemma 4

We can show the proof by contradiction, assume a given node i where its SCC (i.e. set $\mathcal{C}_i$) has no entering edge and $\mathcal{P}_i \neq \emptyset$. The fact that $\mathcal{P}_i \neq \emptyset$ implies that there exist at minimum one node outside of $\mathcal{C}_i$ which can reach node $i \in \mathcal{C}_i$. Hence, there exist an entering edge to its own SCC which contradict the original assumption.

A.7 Proof of Proposition 1

The three statements follows directly from Definitions 2-4 as results from update rules (4), (8), and (12). The condition $\mathcal{P}_i \neq \emptyset$ denotes that there exist at least one node in $\mathcal{P}_i$ that can reach a node in $\mathcal{C}_i$, hence the existence of at least an entering edge to node i's SCC. Conversely, the absence of entering nodes is denoted by $\mathcal{P}_i = \emptyset$. The existence of at least an exiting edge is denoted by any $o_{i,j}[n] = 1$ for all node $j \in \mathcal{C}_i$, while the absence of exiting edge is denoted by $o_{i,j}[n] = 0,\ \forall j \in \mathcal{C}_i$.

A.8 Proof of Lemma 5

Each new edge (j, i) creates a cycle containing all nodes within the elementary path from $i \in \mathcal{S}_j^0 \subseteq \mathcal{V}_{\text{sour}}^0$ to the $j \in \mathcal{V}_{\text{sink}}^0$, merging the corresponding SCCs into a single SCC. As it occurs simultaneously for all sink-sccs towards all existing source-sccs, this ensure there exist a path from node j to node i for every original edge $(i, j) \in \mathcal{E}$. Hence, by the definition of weakly connected graph, the resulting graph is strongly connected.

A.9 Proof of Theorem 3

The Algorithm 4 reflects the described step in Lemma 5, hence strongly connects the whole graph.

Upper bound of the added links: The link addition procedure in step 12 introduces $|\mathcal{S}_i^0| \leq |\mathcal{V}_{\text{sour}}^0|$ number of new links for each $i \in \mathcal{V}_{\text{sink}}^0$. Thus, the number of added links will be $\sum_{i \in \mathcal{V}_{\text{sink}}^0} |\mathcal{S}_i^0|$ and is upper-bounded by $|\mathcal{V}_{\text{sour}}^0||\mathcal{V}_{\text{sink}}^0|$.

Computational complexity: The Algorithm 3 in step 1 runs in $3n$ iterations, while the link addition step (steps 3-13) requires $2n$ steps due to the selection of representative nodes and information broadcast. Hence, by a simple calculation, the execution of Algorithm 4 requires a total $5n$ iterations.

Optimality gap: The minimum number of edges that must be added to strongly connect a weakly connected digraph is equal to $\max\{|\mathcal{V}_{\text{sour}}^0|, |\mathcal{V}_{\text{sink}}^0|\}$, see [14], [15]. This stems from the fact that we need to introduce at least one exiting edge on sink-scc and at least one entering edge on source-scc. Then, the number of the new edges added through Algorithm 4 is $|\Delta\mathcal{E}^+| \geq \max\{|\mathcal{V}_{\text{sour}}^0|, |\mathcal{V}_{\text{sink}}^0|\}$. Thus, the optimality gap can be calculated as $\Delta^* = |\Delta\mathcal{E}^+| - \max\{|\mathcal{V}_{\text{sour}}^0|, |\mathcal{V}_{\text{sink}}^0|\}$. As the number of added link is $\sum_{i \in \mathcal{V}_{\text{sink}}^0} |\mathcal{S}_i^0|$, the optimality gap is $\Delta^* = \sum_{i \in \mathcal{V}_{\text{sink}}^0} |\mathcal{S}_i^0| - \max\{|\mathcal{V}_{\text{sour}}^0|, |\mathcal{V}_{\text{sink}}^0|\}$.

A.10 Proof of Corollary 2

When $|\mathcal{V}_{\text{sour}}^0| = 1$ or $|\mathcal{V}_{\text{sink}}^0| = 1$, we can write the right hand side of inequality (14) as

$$\sum_{i \in \mathcal{V}_{\text{sink}}^0} |\mathcal{S}_i^0| - \max\{|\mathcal{V}_{\text{sour}}^0|, |\mathcal{V}_{\text{sink}}^0|\} \leq |\mathcal{V}_{\text{sour}}^0||\mathcal{V}_{\text{sink}}^0| - \max\{|\mathcal{V}_{\text{sour}}^0|, |\mathcal{V}_{\text{sink}}^0|\} = 0. \tag{17}$$

Hence, the optimality gap $\Delta^* = 0$, i.e., number of links obtained from Algorithm 4 is minimum.

A.11 Proof of Theorem 4

As stated in Lemma 5 the combination of new links from each $i \in \mathcal{V}^m_{\text{sink}}$ to all $j \in \mathcal{S}^m_i$ strongly connects the subgraph where node i belongs to (Step 10). In addition, the new links from each $i \in \mathcal{V}^m_{\text{isol}}$ to a random node $j \notin \mathcal{P}_i \cup \mathcal{C}_i$ (Step 8) connect the associated disjoint subgraphs to form either new weakly connected subgraphs or new isolated-sccs (from multiple isolated-sccs forming a cycle).

Now, consider a case where the original graph consists of d^0 weakly connected subgraphs and assume d^0 is even. For $m = 1$, only nodes $i \in \mathcal{V}^0_{\text{sink}}$ add new links following step 10, resulting in d^0 isolated-sccs. Next, consider an extreme condition in the subsequent link addition ($m = 2$), where any distinct pair of isolated-sccs connect to each other to form a new isolated-sccs. This reduces the number of disjoint subgraphs into $d^0/2$.

Note that we can consider the above as the worst-case scenario, given the following arguments: (a) when isolated-scc exists in the original graph, the number of disjoint graphs will already be reduced after $m = 1$, (b) in many cases several isolated-sccs can chain together, thus further reducing the number of disjoint graph after $m = 2$, and (c) the extreme case for odd d^0 after $m = 2$ will be analogous to the above (even) case as the last single isolated-scc need to add link to a pair of connected isolated-sccs. Hence, it is guaranteed that after 2 number of link additions the number of disjoint subgraph will be less than half of the original, i.e., $d^{m+2} \leq d^m/2$. Thus, the proposed Algorithm 5 guarantees the reduction of the number of disjoint subgraphs which results in a strongly connected graph under finite number of link-addition steps.

Upper bound of the added links: Since the number of disjoint subgraphs is guaranteed to be less than half of its original after 2 steps of link addition, i.e., $d^{m+2} \leq d^m$, Algorithm 5 will finish at most within $2\lceil \log_2 d^0 \rceil$ link additions.

In the first link addition, i.e., $m = 1$, each node $i \in \mathcal{V}^0_{\text{sink}}$ adds $|\mathcal{S}^0_i| \leq |\mathcal{V}^0_{\text{sour}}|$ number of new links (step 10). At the same time each node $i \in \mathcal{V}^0_{\text{isol}}$ adds a new link towards other disjoint subgraphs (step 8). Note that after each link addition, any weakly connected subgraph becomes an isolated-sccs while any existing isolated-sccs add a new exiting edge. From the above observations, we can infer that any new weakly connected subgraph that is created after adding new links will only have a single sink-scc. Note that the subsequent new edges from this single sink-scc to all source-sccs, is still within the considered worst-case scenario where pair of isolated-sccs are selecting each other to add new links. Hence, with the exception of $\sum_{i \in \mathcal{V}^0_{\text{sink}}} |\mathcal{S}^0_i|$ number of links from step 10 at $m = 0$, the number of new links to connect the disjoint subgraphs via Algorithm 5 can be upper-bounded by $d^0 + d^0/2 + d^0/4 + \cdots \leq 2d^0$. Thus, in total the number of added links is upper-bounded by $\sum_{i \in \mathcal{V}^0_{\text{sink}}} |\mathcal{S}^0_i| + 2d^0$.

Computational complexity: The Algorithm 3 runs in $3n$ iterations, while each link addition steps will need $2n$ steps due to the selection of representative node and information broadcast. Then by simple calculation, each link addition step requires $5n$ iterations. In total, as $m \leq 2\lceil \log_2 d^0 \rceil$, Algorithm 5 requires at maximum $3n + 10n\lceil \log_2 d^0 \rceil$) iterations.

Optimality gap: The minimum number of edges that must be added to strongly connect a disconnected digraph is $\max\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\} + |\mathcal{V}^0_{\text{isol}}|$, as shown in [14], [15]. This stems from the fact that we need to introduce at least one exiting edge on sink-sccs and isolated-sccs and at least one entering edge on source-sccs and isolated-sccs. Thus, the optimality gap can be calculated as $\Delta^* = |\Delta\mathcal{E}^+| - (\max\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\} + |\mathcal{V}^0_{\text{isol}}|)$. with the upper-bound is given by $\Delta^* \leq \sum_{i \in \mathcal{V}^0_{\text{sink}}} |\mathcal{S}^0_i| + 2d^0 - (\max\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\} + |\mathcal{V}^0_{\text{isol}}|)$.

A.12 Proof of Lemma 6

Algorithm 6 constructs the ordering by iterating and checking each pairing of sink and reachable sources. Note that each pairings already ensures that there is a path from any selected source $j \in \mathcal{S}^0_i \subseteq \mathcal{V}^0_{\text{sour}}$ to the given $i \in \mathcal{V}^0_{\text{sink}}$, which is a requirement for Property 1. Initially, at the first iteration we can add any $i \in \mathcal{V}^0_{\text{sink}}$ and its respective $j \in \mathcal{S}^0_i$. Then, in the subsequent iterations, assuming that the pairing list contains no duplicate sink, the order can be updated as long as there exist a reachable source that has not been listed from a given pairing, i.e., $j \notin \{v(1), \ldots, v(p)\}$. If all sources $j \in \mathcal{S}_i$ already included in the ordering, i.e., $j \notin \{v(1), \ldots, v(p)\}$, then the given sink already satisfy the condition for Property 3, thus can be added later into $w(p+1), \ldots, w(|\mathcal{V}^0_{\text{sink}}|)$ after finished inspecting all the pairing list. In a similar argument, all the sources that is not selected during iterations, $\forall j \mid j \in \mathcal{V}^0_{\text{sour}} \setminus \{v(1), \ldots, v(p)\}$, satisfy the condition for Property 2 and can be assigned into $v(p+1), \ldots, v(|\mathcal{V}^0_{\text{sour}}|)$.

A.13 Proof of Theorem 5

The proof follow analogously to the existing result in [14], [15]. First, let us consider $|\mathcal{V}^0_{\text{isol}}| = 0$ and focus on all the augmented edges with the exception of $(w(i), v(i)) \mid p+1 \leq i \leq \min\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\}$. These in total introduces $p + |||\mathcal{V}^0_{\text{sour}}| - |\mathcal{V}^0_{\text{sink}}||$ new edges. Observe that by augmenting these edges, the nodes $v(1), \ldots, v(p), w(1), \ldots, w(p)$ and either $v(|\mathcal{V}^0_{\text{sink}}|), \ldots, v(|\mathcal{V}^0_{\text{sour}}|)$ for $|\mathcal{V}^0_{\text{sink}}| < |\mathcal{V}^0_{\text{sour}}|$ or $w(|\mathcal{V}^0_{\text{sour}}|), \ldots, w(|\mathcal{V}^0_{\text{sink}}|)$ for $|\mathcal{V}^0_{\text{sour}}| < |\mathcal{V}^0_{\text{sink}}|$ are on a directed cycle (denoted by $\mathcal{C}$) and thus strongly connected. Thus, all of these nodes and their respective sccs are mutually reachable.

By property (2) there is a path from $v(i),\ p+1 \leq i \leq \min\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\}$ to some vertex in $w(j),\ 1 \leq i \leq p$ and hence to all vertices on $\mathcal{C}$. Then, from property (3) and the addition of edge $(w(i), v(i))$ for $p+1 \leq i \leq \min\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\}$, there is a path from every node in the cycle $\mathcal{C}$ to each $v(i),\ p+1 \leq i \leq \min\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\}$. A similar argument shows that there is a directed path from the nodes in the cycle $\mathcal{C}$ to each $w(i),\ p+1 \leq i \leq \min\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\}$ and from each $w(i),\ p+1 \leq i \leq \min\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\}$ to the nodes in cycle $\mathcal{C}$.

To this end, the set $(w(i), v(i))$ for $p+1 \leq i \leq \min\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\}$ introduces $\min\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\} - p$ new edges. In addition to the $p + |||\mathcal{V}^0_{\text{sour}}| - |\mathcal{V}^0_{\text{sink}}|||$ number of edges introduced previously, the total number of augmented set in $\Delta\mathcal{E}^+$ is $\min\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\} + |||\mathcal{V}^0_{\text{sour}}| - |\mathcal{V}^0_{\text{sink}}||| = \max\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\}$.

Note that for the case of $|\mathcal{V}^0_{\text{isol}}| > 0$, the modification in (16) adds $|\mathcal{V}^0_{\text{isol}}|$ new edges by chaining the nodes $v(|\mathcal{V}^0_{\text{sour}}|+1), \ldots, v(|\mathcal{V}^0_{\text{sour}}|+|\mathcal{V}^0_{\text{isol}}|)$ in the directed cycle $\mathcal{C}$. The rest of the proof follows the previous discussion when $|\mathcal{V}^0_{\text{isol}}| = 0$, which ensures all nodes are mutually reachable. The additional modification for connecting isolated-sccs results in the total number of augmented set introduced in $\Delta\mathcal{E}^+$ as $\max\{|\mathcal{V}^0_{\text{sour}}|, |\mathcal{V}^0_{\text{sink}}|\} + |\mathcal{V}^0_{\text{isol}}|$.

REFERENCES

[1] M. W. S. Atman and A. Gusrialdi, "Distributed algorithms for verifying and ensuring strong connectivity of directed networks," in *2021 60th IEEE Conference on Decision and Control (CDC)*, Dec. 2021, pp. 4798–4803.

[2] A. Gusrialdi and Z. Qu, "Distributed estimation of all the eigenvalues and eigenvectors of matrices associated with strongly connected digraphs," *IEEE Control Systems Letters*, vol. 1, no. 2, pp. 328–333, 2017.

[3] T. Charalambous, M. G. Rabbat, M. Johansson, and C. N. Hadjicostis, "Distributed finite-time computation of digraph parameters: Left-eigenvector, out-degree and spectrum," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 2, pp. 137–148, Jun. 2016.

[4] V. S. Mai and E. H. Abed, "Distributed optimization over directed graphs with row stochasticity and constraint regularity," *Automatica*, vol. 102, pp. 94–104, 2019.

[5] Q. Yang and G. Chen, "Primal-dual subgradient algorithm for distributed constraint optimization over unbalanced digraphs," *IEEE Access*, vol. 7, pp. 85 190–85 202, 2019.

[6] T. Charalambous, Y. Yuan, T. Yang, W. Pan, C. N. Hadjicostis, and M. Johansson, "Distributed finite-time average consensus in digraphs in the presence of time delays," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 4, pp. 370–381, 2015.

[7] L. Sabattini, C. Secchi, and N. Chopra, "Decentralized estimation and control for preserving the strong connectivity of directed graphs," *IEEE Transactions on Cybernetics*, vol. 45, no. 10, pp. 2273–2286, 2014.

[8] Z. Qu and M. A. Simaan, "Modularized design for cooperative control and plug-and-play operation of networked heterogeneous systems," *Automatica*, vol. 50, no. 9, pp. 2405–2414, 2014.

[9] A. Gusrialdi, "Distributed algorithm for link removal in directed networks," in *International Conference on Complex Networks and Their Applications*. Springer, 2020, pp. 509–521.

[10] "Ros-robot operating system," https://www.ros.org/, accessed: 2022-07-03.

[11] "Open field message bus," https://openfmb.ucaiug.org/, accessed: 2022-07-03.

[12] H. Efstathiades, D. Antoniades, G. Pallis, M. D. Dikaiakos, Z. Szlávik, and R.-J. Sips, "Online social network evolution: Revisiting the twitter graph," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 626–635.

[13] S. Dobrev, E. Kranakis, D. Krizanc, J. Opatrny, O. M. Ponce, and L. Stacho, "Strong connectivity in sensor networks with given number of directional antennae of bounded angle," in *International Conference on Combinatorial Optimization and Applications*. Springer, 2010, pp. 72–86.

[14] K. P. Eswaran and R. E. Tarjan, "Augmentation problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 653–665, Dec. 1976.

[15] S. Raghavan, "A note on eswaran and tarjan's algorithm for the strong connectivity augmentation problem," in *The Next Wave in Computing, Optimization, and Decision Technologies*, ser. Operations Research/Computer Science Interfaces Series, B. Golden, S. Raghavan, and E. Wasil, Eds. Boston, MA: Springer US, 2005, pp. 19–26.

[16] M. Sharir, "A strong-connectivity algorithm and its applications in data flow analysis," *Computers & Mathematics with Applications*, vol. 7, no. 1, pp. 67–72, Jan. 1981.

[17] H. N. Gabow, "Path-based depth-first search for strong and biconnected components," *Information Processing Letters*, vol. 74, no. 3, pp. 107–114, May 2000.

[18] P. Lammich, "Verified efficient implementation of gabow's strongly connected component algorithm," in *Interactive Theorem Proving*, ser. Lecture Notes in Computer Science, G. Klein and R. Gamboa, Eds. Cham: Springer International Publishing, 2014, pp. 325–340.

[19] Z. Wang, Y. Wu, Y. Xu, and R. Lu, "An Efficient Algorithm to Determine the Connectivity of Complex Directed Networks," *IEEE Transactions on Cybernetics*, pp. 1–8, 2020.

[20] T. Watanabe and A. Nakamura, "Edge-connectivity augmentation problems," *Journal of Computer and System Sciences*, vol. 35, no. 1, pp. 96–144, Aug. 1987.

[21] K. V. Klinkby, P. Misra, and S. Saurabh, "Strong connectivity augmentation is FPT," in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, ser. Proceedings. Society for Industrial and Applied Mathematics, Jan. 2021, pp. 219–234.

[22] X. Bao, L. Han, C. Deng, H. Zhang, and W. Tan, "Robust topology construction method with radio interface constraint for multi-radio multi-channel wireless mesh network using directional antennas," *International Journal of Distributed Sensor Networks*, vol. 12, no. 9, p. 1550147716668062, Sep. 2016.

[23] N. Chen, T. Qiu, Z. Lu, and D. O. Wu, "An Adaptive Robustness Evolution Algorithm With Self-Competition and its 3D Deployment for Internet of Things," *IEEE/ACM Transactions on Networking*, vol. 30, no. 1, pp. 368–381, Feb. 2022.

[24] A. Gusrialdi, Z. Qu, and S. Hirche, "Distributed link removal using local estimation of network topology," *IEEE Transactions on Network Science and Engineering*, vol. 6, no. 3, pp. 280–292, Jul. 2019.

[25] B. M. Nejad, S. A. Attia, and J. Raisch, "Max-consensus in a max-plus algebraic setting: The case of fixed communication topologies," in *2009 XXII International Symposium on Information, Communication and Automation Technologies*, Oct. 2009, pp. 1–7.

[26] ——, "Max-consensus in a max-plus algebraic setting: The case of switching communication topologies," *IFAC Proceedings Volumes*, vol. 43, no. 12, pp. 173–180, Jan. 2010.

Made Widhi Surya Atman (S'18–M'20) received the BEng and MSc degrees in electrical engineering from the Institut Teknologi Bandung, Indonesia, in 2011 and 2014, respectively. In 2017 and 2020, he received the MEng degree in mechanical and control engineering and the DEng degree in systems and control engineering from the Tokyo Institute of Technology. Since 2020, he has been a postdoctoral research fellow with the Automation Technology and Mechanical Engineering, Tampere University. His research interests include human–swarm interaction, passivity-based control and distributed control of networked system. He is a member of IEEE.

Azwirman Gusrialdi (S'08–M'12) received the BEng and MEng degrees in mechanical & control engineering from the Tokyo Institute of Technology, Japan, in 2006 and 2008 respectively, and the Dr.-Ing. degree in control engineering from Technische Universität at München, Germany, in 2012. He was a postdoctoral researcher at the Department of Electrical and Computer Engineering, University of Central Florida (UCF), Orlando. Since 2019, he has been an assistant professor with the Automation Technology and Mechanical Engineering unit, Tampere University, Finland and leading the Intelligent Networked Systems group. His research interests include design of resilient networked systems, cooperative control and distributed optimization for networked cyber-physical systems. He is a member of the IEEE.