Distributed Algorithms for Verifying and Ensuring Strong Connectivity of Directed Networks

Made Widhi Surya Atman and Azwirman Gusrialdi

Abstract— **This paper considers the problem of distributively verifying and ensuring strong connectivity of directed networks. Strong connectivity of a directed graph associated with the communication network topology is crucial in ensuring the convergence of many distributed algorithms. Specifically, inspired by maximum consensus algorithm, we first propose a distributed algorithm that enables nodes in a networked system to verify strong connectivity of a directed graph. Then, given an arbitrary weakly connected directed graph, we develop a distributed algorithm to augment additional links to ensure the directed graph's strong connectivity. Both algorithms are implemented without requiring information of the overall network topology and are scalable (linearly with the number of nodes) as they only require finite storage and converge in finite number of steps. Finally, the proposed distributed algorithms are demonstrated via several examples.**

Index Terms— **Distributed algorithms, finite-time, strongly connected digraph, max-consensus.**

I. INTRODUCTION

Distributed algorithm plays an important role in estimation, optimization, and control of networked systems [1]–[5]. In contrast to centralized algorithms where all computations are performed at a control center, computations in distributed algorithms are locally performed at individual system by exchanging information with a subset of other 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 preservation of data privacy. Strong connectivity of the graph associated with the communication network topology is crucial in ensuring the convergence of the above mentioned distributed algorithms. Most of the work on distributed estimation, optimization, and control algorithms take for granted (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 necessary to first verify and further ensure (e.g., by adding new links) strong connectivity of a given communication network topology before executing the distributed estimation/optimization/control algorithms. In addition, verifying and ensuring strong connectivity of a communication network topology needs to be performed in a distributed manner to comply with the feature of distributed algorithms that will be deployed in the networked system.

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. Emails: widhi.atman@tuni.fi and azwirman.gusrialdi@tuni.fi

The problem of verifying a strongly connected directed graph (digraph) can be translated into the problem of computing strongly connected components of the given digraph using e.g., Tarjan [6], [7], Kosaraju–Sharir [8], or Gabow [9] algorithm, which are based on depth-first-search approach, as well as the relation-transitive-closure-based Warshall algorithm [10]. Likewise, the problem of ensuring strong connectivity of a directed graph is often described as strong connectivity augmentation problem, which was initiated by [6], [7] and followed by subsequent research in [11], [12] emphasizing that the problem is solvable in polynomial time. Despite the aforementioned approaches, most of the solutions focus on centralized computation and rely on the assumption that information of the overall network topology is known beforehand. A fully distributed approach to solve the problems are scarce in literature, with notable examples are [5], [13] which focuses on maintaining strong connectivity of a digraph after link removals. Nonetheless, the algorithm in [5], [13] still requires the initial graph before link removal to be strongly connected. To the authors' knowledge, there is still no work in the literature which focuses on verifying and ensuring strong connectivity of a directed network in a distributed manner.

The main contribution of the paper is development of distributed algorithms for verifying and ensuring strong connectivity of a digraph. Specifically, we propose a distributed algorithm to verify the strong connectivity of a directed graph, inspired by the maximum consensus algorithm [14]. In addition, we propose a distributed algorithm, together with its optimality gap, to turn a weakly connected graph into a strongly connected digraph by adding new links. The proposed algorithms require finite storage and converge in finite time, which increase linearly with the number of nodes. This feature allows the proposed algorithm to be easily implemented before executing any distributed algorithms whose convergence require strong connectivity property of the network.

The remainder of this paper is organized as follows. In Section II, we review the basic notions from graph theory and provide the problem settings. Section III presents the distributive algorithm to verify whether a given directed network is strongly connected. The distributed algorithm for ensuring strong connectivity of a weakly connected directed graph is then presented in Section IV. Illustrative examples are presented throughout this paper to demonstrate the proposed algorithms. Finally, Section V presents concluding remarks. Due to space limitation, the proof of lemmas are omitted and will be included in the extended version of this paper.

II. PROBLEM FORMULATION

In this section, we recall some basic notions from graph theory and we define the problem settings within this paper.

A. 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 directly obtain information from node i, or node i is directly 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. The set of all in-neighbors of node i is denoted by $\mathcal{N}_i^{in} = \{j \in \mathcal{V} \mid (j, i) \in \mathcal{E}\}$ while $\mathcal{N}_i^{out} = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$ denotes the set of all out-neighbors of node i.

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 $j = 1, \ldots, p-1$. 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 (i, j) for every existing edge (j, i) in the original graph is strongly connected. A *strongly connected component* (or *SCC*) 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 for node $i \in \mathcal{V}$ is represented by 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 $a_{i,j}$.

B. Problem Settings

In this paper, we consider a network of n nodes whose communication network topology can be represented by a directed graph $\mathcal{G}_0 = \{\mathcal{V}, \mathcal{E}_0\}$. Furthermore, it is assumed that there exists no isolated nodes or group of nodes in $\mathcal{G}_0$. Here, we introduce the following assumptions.

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 $\mathcal{N}_i^{in}$, $\mathcal{N}_i^{out}$, and n.
2) Each node is equipped with its own computational resources and is assigned with a unique identifier that 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]. In addition to Assumption 1, we consider a discrete-time case and it is also assumed that the communication between nodes occur in a synchronous manner that may either be defined by a clock or by the occurrence of external events.

This paper's objective is to develop distributed algorithms, under assumption 1, for solving the following problems:

Problem 1: Verify in a distributed manner if directed graph $\mathcal{G}_0$ is strongly connected.

Problem 2: For a weakly connected graph $\mathcal{G}_0$, add additional edges $\Delta\mathcal{E}^+$ in a distributed manner to ensure that the resulting graph $\bar{\mathcal{G}}_m = \{\mathcal{V}, \mathcal{E}_0 \cup \Delta\mathcal{E}^+\}$ is strongly connected.

III. 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 state $\boldsymbol{x_i}[t] \in \mathbb{R}^n$ to estimate 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. The variable $t \in \mathbb{Z}_{\geq 0}$ denotes the t-th time step or the t-th communication event. Furthermore, for notation convenience and readability, it is assumed that t resets to zero when executing new update law.

To this end, each node updates its state $\boldsymbol{x_i}[t]$ for n iterations according to the following rule

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

whose initial condition is chosen as

$$x_{i,j}[0] = \begin{cases} 1, & \text{if } j = i \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Update law (1) is based on the maximum consensus algorithm in [14]. Given the initialization in (2), this approach allows each node i to estimate the existence of paths from any node j to itself when the value of $x_{i,j}[n] = 1$, otherwise $x_{i,j}[n] = 0$ [5]. The n iterations is selected to ensure x_i reach its steady state. We then have the following result which establishes the relationship between $x_i[n]$ and the strong connectivity of directed graph G_0.

Theorem 1: Given a digraph $\mathcal{G}_0$ and each node executes (1) for n iterations whose initial values as in (2), the graph $\mathcal{G}_0$ is strongly connected if and only if $\boldsymbol{x_i}[n] = \mathbf{1}_n$, $\forall i \in \mathcal{V}$.

Proof: We start by showing the necessity ($\Rightarrow$). Since the graph $\mathcal{G}_0$ is strongly connected, under update law (1), which is a maximum consensus algorithm [14], each element in $\boldsymbol{x_i}$ namely $x_{i,j}$ will converge to $\max_i x_{i,j}[0] = 1$ for all $i, j \in \mathcal{V}$. The required number of communication to reach maximum consensus is the maximum of the elementary path between any pair nodes in the graph, i.e., $n-1$ in the worst case. 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 (1) for all j-th row in $\boldsymbol{x_i}[n]$ and for all nodes i in the network, this means that there exist

Algorithm 1 Distributed Algorithm for Solving Problem 1

```
Require: network size $n$, in-neighbor set $\mathcal{N}_i^{in}$
1: initialize each element of $\boldsymbol{x_i}[0]$ as in (2)
2: for each $k$-th element of $\boldsymbol{x_i}$ ($k \in \{1, \dots, n\}$), execute update law (1) for $n$ iterations.
3: assign $f_i[0]$ as in (4)
4: execute update law (3) 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 $f_i[n] = 1$.
```

path from any node j to any node i. Hence the graph $\mathcal{G}_0$ is strongly connected, which contradicts the assumption. ■

Finally, to verify whether $\boldsymbol{x_i}[n] = \mathbf{1}_n$ for all $i \in \mathcal{V}$, each node updates its state $f_i[t]$ for n iterations according to

$$f_i[t+1] = \max_{j \in \mathcal{N}_i^{in} \cup \{i\}} f_j[t] \tag{3}$$

whose initial value is chosen as

$$f_i[0] = \begin{cases} 0, & \text{if } \boldsymbol{x_i}[n] = \mathbf{1}_n \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$

Note that $f_i[0] = 0$ also means that node i is reachable to all nodes in $\mathcal{V}$. We then have the following result,

Theorem 2: Given a digraph $\mathcal{G}_0$ and each node executes in sequence update rule (1) and (3) for n iterations each, with each initial values as in (2) and (4). The graph $\mathcal{G}_0$ is strongly connected if and only if $f_i[n] = 0$ for any $i \in \mathcal{V}$.

Proof: 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 $f_i[n] = 0, \ \forall i \in \mathcal{V}$.

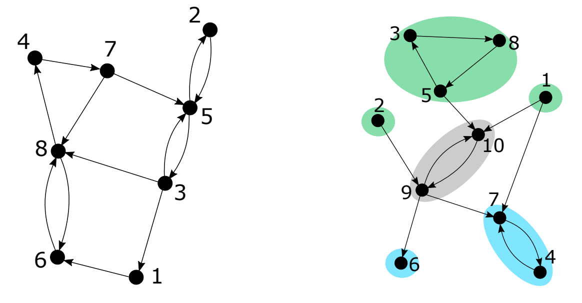
For a non-strongly connected graph $\mathcal{G}_0$, under update law (3), the value of f_i will converge to $\max_i f_i[0] = 1, \ \forall i \in \mathcal{V}$ (weak maximum consensus) if for any node $i \in \mathcal{V}_0$ there exists path ending in i and starting in $j \in \mathcal{V}_1$ [14]. Note that this condition is satisfied as any node $i \in \mathcal{V}_0$ is reachable from all nodes. This ensures that $f_i[n] = f_j[n], \forall i, j \in \mathcal{V}$. ■

The pseudo code of distributed verification algorithm for solving problem 1 is summarized in Algorithm 1, which finishes in $2n$ iterations i.e., its computational complexity is equal to $O(n)$.

The following examples illustrate the proposed distributed verification algorithm.

Example 1: Consider a strongly connected graph in Fig. 1a. All nodes are mutually accessible, hence $\boldsymbol{x_i}[8] = \mathbf{1}_8, \ \forall i \in \mathcal{V}$. All nodes then initialize $f_i[0] = 0$ and it will remain 0 for 8 iterations. Thus, $f_i[8] = 0, \forall i \in \mathcal{V}$ and each node knows that the graph is strongly connected.

Example 2: Consider a weakly connected graph as shown in Fig. 1b with $n = 10$ and let us add edge $(6, 4)$ to the existing graph. This results to an example where there exist nodes, namely nodes 4 and 7, which can retrieve information from all nodes. After 10 iterations in step 2, the state for node $k \in \{4, 7\}$ become $\boldsymbol{x_k}[10] = \mathbf{1}_{10}$, while the state for the rest of node $l \in \{\mathcal{V} \backslash k\}$ is $\boldsymbol{x_l}[10] \neq \mathbf{1}_{10}$. Hence, node k initializes $f_k[0] = 0$ and node l initializes $f_l[0] = 1$. However, during the next 10 iterations in step 4, the node k's state will be changed into $f_k[10] = 1$, for example through the influence of node l, i.e., node 1, 6 or 9. Thus, $f_i[10] = 1, \forall i \in \mathcal{V}$ and all nodes know that the graph is not strongly connected.



(a) (b)

Fig. 1: Examples of (a) a strongly connected graphs and (b) a weakly connected graph with its strongly connected components: source-sccs (green regions), sink-sccs (blue regions), and non-assigned SCC (gray region)

IV. DISTRIBUTED ALGORITHM FOR ENSURING STRONG CONNECTIVITY OF DIRECTED GRAPH

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 a weakly connected digraph. In this section, a distributed algorithm is proposed 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 $\mathcal{G}_0'$ which contains one node for each strongly connected component (SCC) of $\mathcal{G}_0$. The resulting node in $\mathcal{G}_0'$ with no entering edge is called a source, and a node with no exiting edge is called a sink. Finally, the new edges to strongly connect $\mathcal{G}_0'$ can be selected by connecting the existing sink to source following a certain ordering, as shown in [6], [7]. However, the computation for the solution in general is centralized in which information of the overall network topology is required.

In the following, we describe the high-level distributed strategy inspired by [6], [7]. First, each node distributively identifies SCC which corresponds to the sinks and sources in $\mathcal{G}_0'$. New edges are then added by connecting a node in the corresponding sink's SCC to a node in the corresponding source's SCC. These steps are performed iteratively until the resulting graphs become strongly connected. All the computations are performed in a distributed manner and without requiring information of the overall network topology $\mathcal{G}_0$.

Before proceeding, we introduce the following definitions.

Definition 1: source-scc is a SCC with no entering edge and at minimum one exiting edge.

Definition 2: sink-scc is a SCC with at minimum one entering edge and no exiting edge.

An illustration of SCCs is shown in Fig. 1b. Note that a SCC which is neither sink-scc nor source-scc can exist within a weakly connected graph, e.g., the nodes 9 and 10 in Fig. 1b.

A. Distributed Estimation of Strongly Connected Component

In order to add a new edge distributively, each node i needs to distributively estimate the set of SCC it belongs to, namely set $\mathcal{C}_i$, as well as to determine whether its own SCC is a source-scc or sink-scc. Additionally, we define set $\mathcal{P}_i$ as a set of all nodes outside $\mathcal{C}_i$ which can reach any node in $\mathcal{C}_i$. Let us also define the *information number* of node i, denoted as ζ_i, as the number of nodes whose information can be accessed by node i, including his own.

In the following, we propose a distributive approach for the estimation of SCC by utilizing the maximum consensus algorithm. For each node $i \in \mathcal{V}$, let us assign states $\boldsymbol{c_i}[t] \in \mathbb{R}^n$, $\boldsymbol{s_i}[t] \in \mathbb{R}^n$, and $\boldsymbol{o_i}[t] \in \mathbb{R}^n$. State $\boldsymbol{c_i}[t]$ is used to collect the information number from all other accessible nodes to determine $\mathcal{C}_i$ and $\mathcal{P}_i$. Identification of an entering edge to node i's SCC will rely on $\mathcal{P}_i$, and it will be shared through state $\boldsymbol{s_i}[t]$ for source-scc determination. Finally, states $\boldsymbol{o_i}[t]$ is used to identify the exiting edges from node i's SCC to determine sink-scc.

Noting that the existence of a path from node j to i is reflected by $x_{i,j}[n] = 1$, node i's information number can then be calculated as $\zeta_i = \mathbf{1}_n^T \boldsymbol{x_i}[n]$. In order to estimate other nodes' information number, each node updates its state $\boldsymbol{c_i}[t]$ for n iterations according to the following rule

$$c_{i,k}[t+1] = \max_{j \in \mathcal{N}_i^{in} \cup \{i\}} c_{j,k}[t] \tag{5}$$

whose initial condition is chosen as

$$c_{i,j}[0] = \begin{cases} \zeta_i, & \text{if } j = i \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

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

We then have the following results on information number:

Lemma 1: 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 belong to the same SCC (mutually reachable to each other).

Lemma 2: For each node i, the other nodes in the set $\mathcal{P}_i$ have a smaller (positive) information number compared to all nodes in $\mathcal{C}_i$.

Lemma 3: The SCC that comprises of all nodes in the set $\mathcal{C}_i$ has no entering edges if and only if $\mathcal{P}_i = \emptyset$.

As a direct result from Lemma 2, it is clear that the i-th element of $\boldsymbol{c_i}[n]$, i.e. $c_{i,i}[n] = \zeta_i$, always has the highest number. By Lemma 1, each node i can estimate the set $\mathcal{C}_i$ or $\mathcal{P}_i$ by identifying all nodes which have the same or lower information number with itself, respectively. Namely,

$$\mathcal{C}_i := \{\forall j \in \mathcal{V} \mid c_{i,j}[n] = c_{i,i}[n]\}, \tag{7}$$

$$\mathcal{P}_i := \{\forall j \in \mathcal{V} \mid 0 < c_{i,j}[n] < c_{i,i}[n]\}. \tag{8}$$

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

Then, by Lemma 3, each node i can then determine whether its own SCC (set $\mathcal{C}_i$) has no entering edge, namely when $\mathcal{P}_i = \emptyset$. As this information is crucial in identifying the source-scc, each node needs to update its state $\boldsymbol{s_i}[t]$ for n iterations according to the following rule

$$s_{i,k}[t+1] = \max_{j \in \mathcal{N}_i^{in} \cup \{i\}} s_{j,k}[t] \tag{9}$$

whose initial condition is chosen as

$$s_{i,j}[0] = \begin{cases} 1, & \text{if } j = i \text{ and } \mathcal{P}_i = \emptyset \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

The state $\boldsymbol{s_i}[n]$ collects the information from all nodes $k \in \mathcal{P}_i \cup \mathcal{C}_i$ whether node k's SCC has no entering edges.

Finally, to estimate whether exiting edges exist from its own SCC, each node updates its state $\boldsymbol{o_i}[t]$ for n iterations according to the following rule

$$o_{i,k}[t+1] = \max_{j \in \mathcal{N}_i^{in} \cup \{i\}} o_{j,k}[t] \tag{11}$$

whose initial condition is chosen as

$$o_{i,j}[0] = \begin{cases} 1, & \text{if } j = i \text{ and } \exists k \in \mathcal{N}_i^{out} \ (k \notin \mathcal{C}_i) \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

The state $\boldsymbol{o_i}[n]$ collects the information from all nodes $k \in \mathcal{P}_i \cup \mathcal{C}_i$ whether there exist an edge from node k to a node located outside of node k's SCC (set $\mathcal{C}_k$).

We can then establish the following result to distributively determine source-scc and sink-scc.

Proposition 1: Given a digraph $\mathcal{G}_0$ and each node executes in sequence the following update law (1), (5), and (9)-(11) for n iterations each, then each node i can determine the following about its strongly connected component (set $\mathcal{C}_i$):

1) All nodes in the set $\mathcal{C}_i$ is a source-scc if and only if $\mathcal{P}_i = \emptyset$ (equivalent to $s_{i,j}[n] = 1, \ \forall j \in \mathcal{C}_i$) and $\exists j \in \mathcal{C}_i, \ o_{i,j}[n] = 1$.
2) All nodes in the set $\mathcal{C}_i$ is a sink-scc if and only if $\mathcal{P}_i \neq \emptyset$ (equivalent to $s_{i,j}[n] = 0, \ \forall j \in \mathcal{C}_i$) and $o_{i,j}[n] = 0, \ \forall j \in \mathcal{C}_i$.
3) Graph $\mathcal{G}_0$ is strongly connected if and only if $\mathcal{C}_i = \mathcal{V}, \ \forall i \in \mathcal{V}$. Equivalently, $\mathcal{P}_i = \emptyset$, $s_{i,j}[n] = 1$, and $o_{i,j}[n] = 0$ for all $i, j \in \mathcal{V}$.

Proof: The first two statements follows directly from Definition 1-2. The term $\mathcal{P}_i \neq \emptyset$ 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. Contrarily, the absence of entering edge is denoted by $\mathcal{P}_i = \emptyset$. The existence of at least an exiting edge is denoted by $o_{i,j}[n] = 1$ for at least a single node $j \in \mathcal{C}_i$, otherwise $o_{i,j}[n] = 0, \ \forall j \in \mathcal{C}_i$. Finally, the only SCC in a strongly connected graph is the graph itself, hence $\mathcal{C}_i = \mathcal{V}, \ \forall i \in \mathcal{V}$ as stated in the final statement. The later term denotes that there is no entering and exiting edges from $\mathcal{G}_0$. ■

The pseudo code for the proposed distributed estimation of SCC is presented in Algorithm 2, which finishes in $3n$ iterations i.e., its computational complexity is equal to $O(n)$.

Below is an example to illustrate the proposed algorithm.

Algorithm 2 Distributed Estimation of SCC

```
Require: network size $n$, neighbor sets $\mathcal{N}_i^{in}$ and $\mathcal{N}_i^{out}$
1: initialize each element of $\boldsymbol{x_i}[0]$ as in (2)
2: for each $k$-th element of $\boldsymbol{x_i}$ ($k \in \{1,\dots,n\}$), execute update law (1) for $n$ iterations
3: initialize each element of $\boldsymbol{c_i}[0]$ as in (6)
4: for each $k$-th element of $\boldsymbol{c_i}$ ($k \in \{1,\dots,n\}$), execute update law (5) for $n$ iterations
5: estimate $\mathcal{C}_i$ and $\mathcal{P}_i$ by (7) and (8), respectively
6: initialize elements of $\boldsymbol{s_i}[0]$ and $\boldsymbol{o_i}[0]$ as in (10) and (12)
7: for each $k$-th element of $\boldsymbol{s_i}$ and $\boldsymbol{o_i}$ ($k \in \{1,\dots,n\}$), execute update law (9) and (11) for $n$ iterations:
8: node $i$ can determine whether all nodes in $\mathcal{C}_i$ is a source-scc, sink-scc, or neither (Proposition 1).
```

Example 3: Let us consider the weakly connected graph in Fig. 1b and focus on inspecting the states of node 7. At the end of step 4 in Algorithm 2, node 7 can use $\boldsymbol{c_7}[10] = [1\ \ 1\ \ 3\ \ 9\ \ 3\ \ 0\ \ 9\ \ 3\ \ 7\ \ 7]^T$ to determine $\mathcal{C}_7 = \{4,7\}$ and $\mathcal{P}_7 = \{1,2,3,5,8,9,10\}$. The fact that there exist entering edge from outside of the set $\mathcal{C}_7$ (node 1 and 9) is reflected in $\mathcal{P}_7 \neq \emptyset$. With $\mathcal{N}_7^{out} = \{4\}$, node 7 has no exiting edge outside of $\mathcal{C}_7$. Thus, the values of $s_{7,7}[0]$ and $o_{7,7}[0]$ is initialized to 0. Then, at the end of step 7, node 7 will have $\boldsymbol{s_7}[10] = [1\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0]^T$ and $\boldsymbol{o_7}[10] = [1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0]^T$. By inspecting the value of $\boldsymbol{s_7}[10]$ and $\boldsymbol{o_7}[10]$, node 7 can determine that its SCC is a sink-scc as $\mathcal{P}_7 \neq \emptyset$ and $s_{7,k} = o_{7,k} = 0,\ \forall k \in \mathcal{C}_7$.

B. Distributed Link Addition Algorithm

Next, we present distributed algorithm to strongly connect a weakly connected digraph $\mathcal{G}_0$ by iteratively identifying and connecting the sink-sccs to source-sccs.

The decision for new link to be added is computed by each representative node in the sink-scc. For a sink-scc containing multiple nodes, the representative node can be selected by following a predefined rules or by locally communicating among the nodes that belong to the same sink-scc. Consider the new edge to be added as (i^*, j^*), with node i^* denotes the vertex number of the sink-scc's representative node. Node i^* then determines a set $\mathcal{S}_{i^*} \subseteq \mathcal{P}_{i^*}$ containing all nodes in source-sccs that are accessible to i^*.

Lemma 4: For each node i, the set of all nodes in source-sccs that is accessible to node i is denoted by

$$\mathcal{S}_i := \{\forall j \in \mathcal{P}_i \mid s_{i,j}[n] = 1\}. \tag{13}$$

Note that it is not possible to categorize specific source-sccs within $\mathcal{S}_{i^*}$ based on the existing information. Thus, for the candidate of new edge (i^*, j^*), the node j^* is selected randomly from $\mathcal{S}_{i^*}$. Once all the new edges are augmented, the whole procedure is repeated by identifying the new sink-scc and calculate new edge candidate from existing sink-scc to source-scc until the graph becomes strongly connected. The pseudo-code of algorithm for distributed link addition is presented in Algorithm 3.

Algorithm 3 Distributed Algorithm for Solving Problem 2

```
Require: network size $n$, neighbor sets $\mathcal{N}_i^{in}$ and $\mathcal{N}_i^{out}$
1: run Algorithm 2 {for $\mathcal{G}_0 = \{\mathcal{V}, \mathcal{E}_0\}$}
2: while not strongly connected, i.e. $\mathcal{C}_i \neq \mathcal{V}$ do
3:     if $i$ is within sink-scc then
4:         determine representative node within $\mathcal{C}_i$
5:     end if
6:     if $i$ is representative node then
7:         estimate the set $\mathcal{S}_i$ by (13)
8:         randomly select target node candidate $j$ from $\mathcal{S}_i$
9:         add $j$ into $\mathcal{N}_i^{out}$ and establish new link $(i,j)$ {add $i$ into $\mathcal{N}_j^{in}$ and conceptually add $(i,j)$ into $\Delta\mathcal{E}^+$}
10:    end if
11:    run Algorithm 2 {for $\bar{\mathcal{G}}_m = \{\mathcal{V}, \mathcal{E}_0 \cup \Delta\mathcal{E}^+\}$}
12: end while
```

Now, let us consider the case where weakly connected digraph $\mathcal{G}_0$ initially has v_0 number of sink-sccs and w_0 number of source-sccs, and the nodes distributively perform Algorithm 3 which results in $|\Delta\mathcal{E}^+|$ new edges. Furthermore, let us denote Δ^* as the optimality gap between $|\Delta\mathcal{E}^+|$ and the minimum number of links needed to strongly connect $\mathcal{G}_0$. We then have the following main result.

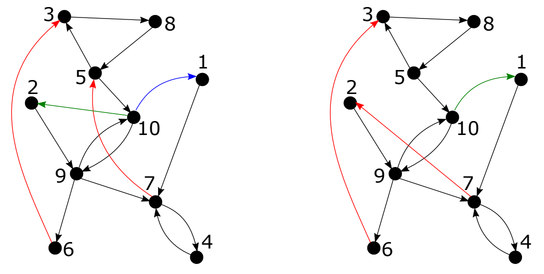
Theorem 3: Given a weakly connected digraph $\mathcal{G}_0 = \{\mathcal{V}, \mathcal{E}_0\}$, then Algorithm 3 results in a strongly connected graph $\bar{\mathcal{G}}_m = \{\mathcal{V}, \mathcal{E}_0 \cup \Delta\mathcal{E}^+\}$ by adding at most $v_0 w_0$ new edges. Furthermore, Algorithm 3 will finish in the worst case of $3n(w_0+1)$ iterations whose optimality gap Δ^* is upper-bounded by $\Delta^* \leq v_0 w_0 - \max\{v_0, w_0\}$.

Proof: Each new link (i^*, j^*) from Algorithm 3 creates a cycle containing all SCCs within the elementary path from j^* (a source-scc) to i^* (a sink-scc), combining them into a single SCC. Hence, each new link reduces the total number of SCCs and ensures that the number of sink-sccs and source-sccs are not increasing at each while-loop. Thus, the while-loop in Algorithm 3 will eventually reach a single SCC, i.e. strongly connected graph $\bar{\mathcal{G}}_m$, in a finite number of loop.

Maximum number of new links: For each while-loop in Algorithm 3 (step 3 to 11), the number of the added links is equal to the number of sinks-sccs, with v_0 as the maximum. Then, the worst case scenario is for all sink-sccs to coincidentally select nodes from the same source-scc at each while-loop. Hence, the maximum number of while-loop is the original number of source-sccs, i.e., w_0, and the number of new links will be upper-bounded by $v_0 w_0$.

Computational complexity: The Algorithm 2 in step 1 and 11 runs in $3n$ iterations, while the rest of the steps (step 3-10) can be done instantaneously. As the maximum number of while-loop is w_0, then by simple calculation, the Algorithm 3 requires $3n(w_0+1)$ iterations in the worst case.

Optimality gap: The minimum number of links to strongly connect a weakly connected digraph is $\max\{v_0, w_0\}$, as shown in [6], [7]. As the maximum number of added link is $v_0 w_0$, the optimality gap Δ^* is upper-bounded by $v_0 w_0 - \max\{v_0, w_0\}$. ∎



(a) Sub-optimal solution (b) Optimal solution

Fig. 2: Example of solutions to problem 2. The red, green, and blue arrows respectively represent the augmented edges for the 1st, 2nd, and 3rd additions.

Corollary 1: Given a weakly connected digraph $\mathcal{G}_0$ with a source-scc ($v_0 = 1$) or a sink-scc ($w_0 = 1$), then Algorithm 3 results in an optimal solution with minimum link addition.

Proof: When $v_0 = 1$ or $w_0 = 1$, we can rewrite the optimality gap in Theorem 3 as $\Delta^* = \max\{v_0, w_0\} - \max\{v_0, w_0\} = 0$, i.e., number of links obtained from Algorithm 3 is minimum. ■

Below is an example to illustrate the proposed algorithm.

Example 4: Consider the weakly connected digraph used in Example 3 (Fig. 1b), which initially has two sink-sccs, namely $\{6\}$ and $\{4, 7\}$. In this example, we consider a predetermined rule to select sink-scc's representative, namely by selecting the node with highest vertex number. Both sink-scc representatives, node 6 and 7, then estimate all accessible nodes that belong to source-sccs by each inspecting $\boldsymbol{s_6}[10] = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T$ and $\boldsymbol{s_7}[10] = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T$. This results in $\mathcal{S}_6 = \{2, 3, 5, 8\}$ and $\mathcal{S}_7 = \{1, 2, 3, 5, 8\}$. New links are then selected towards a random node in $\mathcal{S}_6$ and $\mathcal{S}_7$.

Let us consider a sub-optimal solution as illustrated in Fig. 2a. In the first loop, both node 6 and 7 select nodes from the same source-scc, proceeding with $(6, 3)$ and $(7, 5)$. This merges all nodes except the remaining source sccs $\{1\}$ and $\{2\}$ into a single sink-scc with node 10 as the representative. Then, a single link is added in each subsequent loop to connect the remaining source-sccs. Here, the algorithm finish in 120 iterations and with optimality gap $\Delta^* = 1$ which has upper-bound of 3 from Theorem 3. Theorem 3 also guarantees to provide at maximum $v_0 w_0 = 6$ new edges, much less than the possible new edges that can be selected, which are $n(n-1) - |\mathcal{E}_0| = 77$ edges. As a comparison, an optimal solution is illustrated in Fig. 2b, where the algorithm finishes in 90 iterations.

Remark 1 (Privacy Preservation): All information that each node collects through Algorithm 1, 2 and 3 are the existence of path from other nodes to itself (state $\boldsymbol{x_i}$), the general notion of the strong connectivity (state f_i), the other nodes information number (state $\boldsymbol{c_i}$), and the information of other SCCs' entering and exiting edges (state $\boldsymbol{s_i}$ and $\boldsymbol{o_i}$). This information is not sufficient for each node to reveal the global topology, thus preserving privacy in terms of overall network's topology.

Remark 2 (Implementation): An implementation of Algorithm 1, 2 and 3 in python language is available in `http://github.com/TUNI-IINES/dist-strong-connectivity`.

V. Conclusions and Future Work

This paper proposes two distributed and finite time algorithms to verify strong connectivity of a directed graph and to strongly connect a weakly connected graph. The strategy is inspired by maximum consensus algorithm which is known to have finite computation time. The proposed strategies provide the solutions without requiring knowledge of the overall network topology and further preserve the privacy within the network. 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 includes extending the distributed link addition algorithm for general directed graphs.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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, July 2019.

[14] 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.