A Distributed Method for Detecting Critical Edges and Increasing Edge Connectivity in Undirected Networks

Deepalakshmi Babu Venkateswaran, Zhihua Qu, and Azwirman Gusrialdi

***Abstract*— A critical edge is an edge whose removal results in the associated undirected network becoming disconnected. Identifying these critical edges and enhancing the corresponding edge connectivity is critical for achieving robustness in network connectivity. While existing methodologies are effective, they are centralized and rely on global information, which makes them not scalable with respect to the network size or its implementation. To address these shortcomings, a fully distributed approach is introduced in this paper to identify all the critical edges within an undirected network without requiring a central coordinating authority. Computationally, the proposed method has a complexity of $\mathcal{O}(n)$, where n is the number of nodes, which is more efficient when compared to the centralized approaches. Furthermore, the proposed method can be used to incrementally increase the network's edge connectivity to 2, thus addressing the network's most vulnerable edges.**

I. Introduction

A. Motivation and Literature Review

Network robustness is a critical factor in ensuring the resilience of systems against disruptions, such as link failures or targeted attacks. One key metric that quantifies this robustness is edge connectivity, which measures the minimum number of edges that must be removed to disconnect the network. A network with high edge connectivity can withstand multiple edge failures without losing overall connectivity. Conversely, networks with low edge connectivity, particularly those with critical edges (or bridges), are vulnerable, as the failure of just one such edge can lead to network disconnection [1]. Identifying and reinforcing these critical edges is therefore essential to enhancing network resilience.

Distance-first approaches identify critical edges where each node initiates a search to traverse across the entire graph [2], [3]. Increasing the robustness of the network by identifying and reinforcing critical edges has been addressed through various approaches. [4] explores critical edge identification and robustness improvement in IoT networks through dynamic adjustments and self-competition mechanisms, while [5] uses betweenness scores for critical edge and vulnerability assessment in transportation networks. Some survey papers such as [6], [7] compare methodologies for network vulnerability assessment, emphasizing disruption risks. Some approaches that improve the network robustness without identifying the critical edges include [8], [9] and they use different optimization algorithms. However, all of the above approaches are centralized in nature. Leveraging the scalability, fault tolerance, and adaptability of distributed algorithms [10], we aim to design a fully distributed approach that enhances network connectivity and resilience.

The work of Z. Qu is supported in part by grants by US Department of Energy's awards DE-EE0007998, DE-EE0009028, DE-EE0009152, Award DE-EE0009339, and Award DEAC05-76RL01830. The work of A. Gusrialdi is supported by the Academy of Finland under project no. 330073.

D. Babu Venkateswaran and Z. Qu are with the University of Central Florida, Orlando, FL 32816, USA. Emails: deepalakshmi.babuvenkateswaran@ucf.edu, qu@ucf.edu. A. Gusrialdi is with the Automation and Mechanical Engineering unit at Tampere University, Finland. Email: azwirman.gusrialdi@tuni.fi

B. Statement of Contributions

The proposed distributed method consists of four algorithms. First, the standard maximum/minimum consensus algorithm is judiciously applied to identify not just network connectivity [11], [12], but also the whole neighboring structure, which includes every node's neighbors and their minimum distances within the network. The second algorithm ensures connectivity of the network, by adding edges in a certain order if the original is not connected. Third, a novel distributed algorithm is developed to determine whether a specific edge belongs to a cycle in the network, and it can be used to detect all critical edges of the network. The final algorithm is to perform edge addition strategically so that all critical edges are eliminated.

These four algorithms form a distributed method that fortifies network connectivity, ensuring resilience against link failures and/or Denial of Service (DoS) attacks in networks such as the Internet, multi-agent systems, robotics and mobile communication networks, networked control systems [13], smart grids [14], and cyber-physical systems [15]. Our approach is applicable to real-world scenarios where local communication is the norm, such as in multi-agent systems and smart grids. In these applications, network robustness can be achieved through local communication, and connectivity can be maintained under denial of service attacks.

II. Problem Formulation

Consider an undirected network of n nodes described by $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ is the set of integers 1 to n, and $\mathcal{E}$ is the set of edges between the nodes. Node i has a neighborhood set $\mathcal{N}_i = \{j \in \mathcal{N} : e_{ij} \in \mathcal{E}\}$ indicating direct communication links.

A network $\mathcal{G}$ is said to be *connected* if any pair of nodes is linked by a path. The *neighboring structure* of a node consists of its neighbors and their minimum distances, illustrating how nodes are interconnected within the network. A *cycle* in the network is a closed path that returns to the starting node without retracing any edge. A *critical edge* is an edge whose removal results in the disconnection of the

network. The *edge connectivity* of a network is the minimum number of edges that must be removed to disconnect it, serving as a measure of the network's robustness. A network with *edge connectivity 1* has at least one critical edge, while a network with *edge connectivity 2 or higher* is more robust, as it lacks critical edges and can withstand the removal of multiple edges.

Our objective is to determine the network's neighboring structure as well as all the critical edges, if any. The proposed approach must be distributed and yield the result in finite steps. To facilitate its development, the following assumption is made.

Assumption 1. *Every node in $\mathcal{G}$ is uniquely indexed from 1 to n. Every node knows the indices of its neighbors and the total number of nodes n. Furthermore, when necessary, a node can establish a new edge with another node through communicating with its relevant neighbors and dispatching network addresses.*

In this paper, the global information of the graph is gradually built through the proposed multi-step processes whose implementation is fully distributed. As these processes evolve, each node incrementally accumulates information about the rest of the nodes via its local neighbors. By the end of these distributed, finite-step algorithms, each node has complete information about neighboring structure, critical edges, and potential communication edges to be added. No global information is required beyond the simple indexing requirement in Assumption 1.

III. Distributed Network Discovery

In this section, the following distributed algorithms are presented:

- An n-step algorithm to estimate the connectivity of the network and to determine the neighbor structure, including the distance from one node to every other node in the network.
- An n-step algorithm to ensure the connectivity of the network by adding edges in an ordered fashion.
- A 2-step algorithm to determine the existence of critical edges/nodes by identifying the absence of cycle(s) containing the specific edge or alternate paths between any pair of two neighboring nodes.
- If critical edge(s) is(are) identified, an n-step distributed algorithm to add a minimum number of new edges to eliminate critical edges and increase edge connectivity to 2.

A. Distributed Identification of Node's Neighbor Structure

For each node $i \in \mathcal{N}$, consider the following two states: $\xi_i(k) = [\xi_{i,1}(k) \ \cdots \ \xi_{i,n}(k)]^T \in \mathbb{R}^n$ denoting the node i's estimate of its in-neighbors, and $\omega_i(k) = [\omega_{i,1}(k) \ \cdots \ \omega_{i,n}(k)]^T \in \mathbb{R}^n$ denoting node i's estimate of its neighbor structure.

The distributed maximum and minimum protocols operate over n consecutive iterations, defined as follows for each iteration $k = 0, \cdots, (n-1)$:

$$\xi_{i,j}(k+1) = \max_{l \in \mathcal{N}_i \cup i} \xi_{l,j}(k), \quad j \in \mathcal{N}, \tag{1}$$

$$\omega_{i,j}(k+1) = \begin{cases} \omega_{i,j}(k) & \text{if } \xi_{i,j}(k+1) \\ & = \xi_{i,j}(k), \\ \min_{l \in \mathcal{N}_i}(\omega_{l,j}(k)+1) & \text{if } \xi_{i,j}(k+1) \\ & > \xi_{i,j}(k) \end{cases} \tag{2}$$

where $\mathcal{N}_i$ is the neighbor set of the node i and the initial conditions are given as

$$\xi_{i,j}(0) = \begin{cases} 1, & \text{if } j = i \\ 0, & \text{otherwise} \end{cases} \quad \omega_{i,j}(0) = \begin{cases} 0, & \text{if } j = i \\ \infty, & \text{otherwise} \end{cases} \tag{3}$$

Theorem 1: *Consider an undirected network $\mathcal{G}$ in which each node i executes two concurrent distributed n-step algorithms:* (1) *and* (2)*. Then, the following conclusions can be drawn:*

1) *Node j is one of the p^{th} neighbors of node i: that is, $\{j\} \in \mathcal{N}_i^{(p)}$ if $\omega_{i,j}(n) = p > 0$.*
2) *Each node i knows that the network $\mathcal{G}$ is connected, if and only if all of $\xi_{i,j}(n)$ elements are non zero: that is,*

$$\xi_{i,j}(n) \neq 0, \quad \forall j \in \mathcal{N}. \tag{4}$$

Proof: At $k = 0$, $\xi_{i,j}(0)$ is set to 1 if $j = i$ and 0 otherwise, indicating each node is initially only aware of itself. Correspondingly, $\omega_{i,j}(0)$ is initialized to 0 for itself and to ∞ for all other nodes, indicating that the distance to itself is known to be zero, and the distance to any other nodes are unknown.

For each iteration $(k+1)$, it follows $\xi_{i,j}(k+1)$ updates to 1 if j is reachable from i within $k+1$ steps, determined by neighbor l of i having $\xi_{l,j}(k) = 1$. If $\xi_{i,j}(k+1)$ increases (indicates new reachability), $\omega_{i,j}(k+1)$ updates to capture the shortest path distance from i to j, that is the minimum of $\omega_{l,j}(k)+1$ across all neighbors l of i that can reach j, thereby indicating the path length.

It can be seen that $\xi_{i,j}(k)$ is binary and non-decreasing. Note that $\omega_{i,j}(k)$ will remain zero or ∞ until $\xi_{i,j}(k)$ changes from 0 to 1. In addition, we know by induction that, if $\{j\} \in \mathcal{N}_i^{(p)}$, $\xi_{i,j}(k)$ switches from 0 to 1 precisely at step $k = p$. Afterwards, $\xi_{i,j}(k+1) - \xi_{i,j}(k) \equiv 0$, and invariant, and so is $\omega_{i,j}$. Accordingly, the conclusion is drawn. □

Algorithm 1 operates over n steps ($k = 1, \cdots, n$), by each node i to identify its neighbor structure in terms of its p^{th} neighbor set $\mathcal{N}_i^{(p)}$ by executing update laws (1) and (2).

The following observations are worth noting. First, network $\mathcal{G}$ is connected if $\omega_{i,j}(n) > 0$ for all $i \in \mathcal{N}$ and for all $j \neq i$. Second, if there are alternate paths between nodes i and j, the result $\omega_{i,j}(n)$ from algorithm (2) corresponds to the shortest path.

Example 1: For network $\mathcal{G}_1$ in Fig. 1(a), it follows that, at $k = 8$,

$$\omega_4(8) = [1, 1, 1, 0, 1, 2, 2, 3]^T,$$

which shows $\mathcal{N}_4^{(1)} = \{1,2,3,5\}$, $\mathcal{N}_4^{(2)} = \{6,7\}$, and $\mathcal{N}_4^{(3)} = \{8\}$.

For $\mathcal{G}_2$ in Fig. 1(b), it follows that

$$\omega_4(8) = [1,1,1,0,1,2,2,2]^T,$$

which implies $\mathcal{N}_4^{(1)} = \{1,2,3,5\}$ and $\mathcal{N}_4^{(2)} = \{6,7,8\}$. $\triangle$

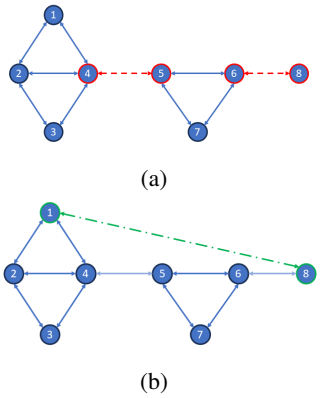


Fig. 1: (a) Network $\mathcal{G}_1$ with *critical edges* (red dotted lines), and (b) Network $\mathcal{G}_2$ without critical edge *by adding an edge* (green dot-dashed line)

B. Distributed Algorithm to Ensure Connectivity

Consider a modified version of update law (2)) for steps $k = n, \cdots, (2n-1)$:

$$\omega_{i,j}(k+1) = \begin{cases} \omega_{i,j}(k) & \text{if } \xi_{i,j}(k+1) \\ & \quad = \xi_{i,j}(k), \\ \omega_{i,i*}(k) + \omega_{j^*,j}(k) & \text{if } \xi_{i,j}(k+1) \\ & \quad > \xi_{i,j}(k). \end{cases} \quad (5)$$

where $\xi_{i,j}(k)$ is obtained using update law (1).

If the original network is not connected, every node in the network becomes aware of this fact at $k = n$. Specifically, if node i does not have node j in its neighboring structure (in the sense that $\xi_{i,j}(n) = 0$), node i knows that the network is not connected. Then, the node i^* with the highest index within the connected neighbors is identified, that is

$$i^* = \max\{j \mid \xi_{i,j}(n) = 1\}.$$

Then, i^* identifies its pair j^*, that is not within its connected neighbors as,

$$j^* = \max\{j \mid \xi_{i,j}(n) = 0\}.$$

A new edge from i^* to j^* is added, which upgrades the network towards being connected. The pseudo-code for this is given in Algorithm A for easy reference.

In the worst case that there is no edge in the original network, Algorithm A can take $n-1$ steps to ensure connectivity. For this reason, even when starting with a connected network, (5) and (1) need to run until $k = 2n$ so that there is no need for central coordination.

Algorithm 2 operates by running the update laws (5) and (1) for n steps. After identifying the nodes i^* and j^* in each iteration of Algorithm A, the update rules (5) and (1) should be immediately applied to ensure that the latest values of $\xi_{i,j}(k)$ are used for accurate distance calculations and network updates. During the first $(n_c - 1)$ steps, where n_c is the number of disjoint connected components, Algorithm A upgrades the network to ensure connectivity. By $k = n + n_c - 1$, the network is connected, and by $k = 2n$, every node in the network recognizes this connectivity within its neighboring structure.

Algorithm A Distributed Connection of Non-Connected Network

Input: $\xi_{i,j}(k)$ at $k = n$

1: **while** $\xi_{i,j}(k) = 0$ for any j **do**
2: identify node $i^* = \max(j)$ with the highest index where $\xi_{i,j}(k) = 1$.
3: **if** i is i^* **then**
4: find node $j^* = \max(j)$ with the highest index where $\xi_{i,j}(k) = 0$.
5: establish a link with node j^*.
6: **end if**
7: $k = k + 1$
8: **end while**

Output: The network connectivity is established.

C. Distributed Determination of Critical Edges

In this section, we present a distributed 2-steps algorithm (a total of $(2n+2)$-steps when combined with the previous algorithms) for the i^{th} node to determine whether its associated edge e_{il} to its first neighbor node l is a critical edge. To facilitate this, we employ a measure, $\Delta_i^{(il)}$, calculated at the $(n+1)^{\text{th}}$ iteration, to evaluate the presence of alternative pathways between any two directly connected nodes, i, and a neighbor $l \in \mathcal{N}i^{(1)}$. This measure is defined as follows:

$$\Delta_i^{(il)} = [\Delta_{i,1}^{(il)} \ \cdots \ \Delta_{i,n}^{(il)}]^T, \quad \Delta_{i,j}^{(il)} = \omega_{i,j}(n) - \omega_{l,j}(n), \quad (6)$$

where $\omega_{i,j}(n)$ and $\omega_{l,j}(n)$ represent the computed shortest distances from node j to nodes i and l, respectively, obtained after n iterations of (2).

In an undirected and connected network, the value of $\Delta_{i,j}^{(il)}$ for each pair of neighboring nodes i and l, with respect to any other node j, can only be -1, 0, or 1. A value of -1 indicates that node j is relatively closer to node i than to node l, and conversely, a value of 1 suggests node j is nearer to node l. A zero value implies that node j is equidistant from both nodes i and l. Through the analysis of these distance increments, we propose a lemma and a theorem that provide criteria for distributively identifying alternate paths and critical edges within the network, thereby enhancing our understanding of the network's robustness to potential disruptions.

Lemma 1: *Given an undirected and connected network $\mathcal{G}$, consider an edge e_{il} connecting nodes i and l. A node, denoted as k, possesses alternate paths to both i and l, bypassing e_{il}, if and only if at least one node j (potentially*

including k itself and residing within the relevant cycle) satisfies one of the two conditions below:

1) Node j is equidistant to i and l

$$\Delta_{i,j}^{(il)} = 0, \tag{7}$$

2) There exist nodes $i' \in \mathcal{N}_i$ and $l' \in \mathcal{N}_l$ such that

$$\Delta_{i,j}^{(il)} \neq 0, \implies \Delta_{i,j}^{(ii')} = \Delta_{l,j}^{(ll')} = 1. \tag{8}$$

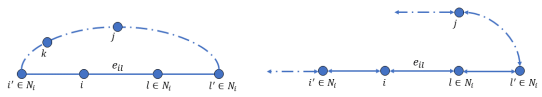
indicating j is on a path that connects i and l through nodes i' and l', thus forming a cycle that includes e_{il}.

Proof: The proof is organized into two main parts: sufficiency and necessity.

Sufficiency: If there exists a node k with alternate paths to i and l that do not pass through e_{il}, then at least one cycle involving e_{il} and other edges is present in the network. This cycle can be detected by examining distances from nodes in the network to i and l:

1) If $\Delta_{i,j}^{(il)} = 0$ for some node j, it indicates that j is equidistant from both i and l. This condition signifies the presence of a cycle that j is a part of, where j is on an alternate path that circumvents e_{il}, showing the cycle's existence without directly counting nodes.
2) Let's consider nodes i' and l', where $i' \in \mathcal{N}_i$ and $l' \in \mathcal{N}_l$ belong to the cycle. There exists a node j that has an equal distance to the cluster consisting of nodes i and l. This implies that departing from either node i or l toward node j results in the distance of the neighbor structure decreasing in both directions, as illustrated in Fig. 2(a). Consequently, both $\Delta_{i,j}^{(ii')}$ and $\Delta_{l,j}^{(ll')}$ are equal to 1, irrespective of the sign of $\Delta_{i,j}^{(il)}$. This condition confirms that j lies on a cycle that includes both i and l, as well as their neighbors, indicating the existence of alternate paths.

Necessity: Assume no alternate paths exist between i and l except through e_{il}. Removing e_{il} disconnects i and l, showing e_{il} is a critical edge. In such a scenario, for any node j not equal to i or l, it's impossible to have $\Delta_{i,j}^{(il)} = 0$ because j cannot be equidistant to i and l without e_{il}. Additionally, you cannot find nodes $i' \in \mathcal{N}_i$ and $l' \in \mathcal{N}_l$ satisfying $\Delta_{i,j}^{(ii')} = \Delta_{l,j}^{(ll')} = 1$ for any j, as there are no cycles including e_{il} and other edges that could provide such alternate paths. Furthermore, we will have $\Delta_{i,j}^{(ii')} * \Delta_{l,j}^{(ll')} = -1$ (i.e., they must have different signs), since leaving the cluster of nodes i and l has opposite effects: closer to node j in one direction, and farther in the other, as shown in Fig. 2(b). □



(a) (b)

Fig. 2: (a) Edge e_{il} is a part of the cycle formed by alternate paths, and (b) Edge e_{il} is not a part of any cycle

Theorem 2: *Given an undirected and connected network $\mathcal{G}$, suppose each node implements a single-step computation* (6)*, in conjunction with the n-step protocols* (1) *and* (2)*. An edge e_{il} is a critical edge within this framework if and only if, for every node $j \in \mathcal{N}$ and for all adjacent nodes $i' \in \mathcal{N}_i$ and $l' \in \mathcal{N}_l$, the following condition is met:*

$$\Delta_{i,j}^{(il)} \neq 0, \quad \text{and} \quad \{\Delta_{i,j}^{(ii')}, \Delta_{l,j}^{(ll')}\} \neq \{1, 1\}. \tag{9}$$

Proof: Condition (9) explicitly negates the scenarios described in (7) and (8) from Lemma 1, indicating e_{il} lacks alternative paths, thus establishing its criticality. Sufficiency arises directly from Lemma 1: if no conditions for non-criticality are met, e_{il} is critical as it is the sole link between i and l. Necessity is self-evident; without e_{il}, connectivity between i and l breaks, signifying its critical.

This proof, by linking the absence of alternate paths to the unique conditions specified, conclusively establishes the criteria for determining the critical nature of an edge within the network. □

Example 2: For $\mathcal{G}_2$ in Fig. 1(b), it follows that, at $k = 9$,

$$\Delta_4^{(45)} = [-1, -1, -1, -1, 1, 1, 1, 0]^T,$$
$$\Delta_4^{(41)} = [1, 0, -1, -1, -1, 0, -1, 1]^T,$$
$$\Delta_5^{(56)} = [0, -1, -1, -1, -1, 1, 0, 1]^T.$$

Note that $\Delta_{4,8}^{(41)} = \Delta_{5,8}^{(56)} = 1$ which implies that the edge $e_{4,5}$ is not a critical edge.

On the other hand, from $\mathcal{G}_1$ in Fig. 1(a),

$$\Delta_4^{(45)} = [-1, -1, -1, -1, 1, 1, 1, 1]^T,$$
$$\Delta_4^{(41)} = [1, 0, -1, -1, -1, -1, -1, -1]^T,$$
$$\Delta_5^{(56)} = [-1, -1, -1, -1, -1, 1, 1, 1]^T,$$

It is straightforward that (9) is satisfied for all i' and l' (e.g., $\Delta_5^{(57)}$) not explicitly shown, thus indicating that edge e_{45} is critical. △

Algorithm 3 runs for 2 steps where each node can identify the set of all its associated critical edges from (9), denoted by $\mathcal{N}_i^c \subset \mathcal{N}_i$ for the i^{th} node in connected graphs.

D. Distributed Edge Addition

This section introduces an algorithm to enhance network robustness by strategically adding new edges to eliminate critical ones, thereby reducing vulnerability to link failures. Our approach focuses on connecting distant nodes within acyclic parts of the network—specifically, those separated by critical edges—to both preserve network connectivity under attack and minimize the additional connections required. By identifying the most remote pairs of nodes adjacent to each critical edge, called *augmentation nodes*, the algorithm ensures these nodes establish new links. Through local propagation, the augmentation nodes receive the information and complete the planned edge addition, which has a maximum of $n + 1$ steps (and, when combined with the previous components, have a total of $3n + 3$ steps).

The first step of the n-step edge addition algorithm, that is **Algorithm 4** is executed only by individually each pair of nodes associated with a critical edge, say $\{i, l\}$ with $i \in \mathcal{N}_l^c$ and $l \in \mathcal{N}_i^c$, where $\mathcal{N}_i^c$ refers to corresponding

neighbor set associated with critical edges. The goal of this step is to identify the corresponding augmentation nodes, $\{i', l'\} \in \mathcal{N}_i^r$, where $\mathcal{N}_i^r$ represents the augmentation nodes (or reconnection nodes) that are strategically chosen as the ones farthest (with respect to the neighbor structure) from the critical edge e_{il}. If there exist multiple nodes at the same distance, the ones with the smallest index are chosen, and it reduces the number of edges added.

$$\{i', l'\} \in \mathcal{N}_i^r \text{ if (11) or (12) or (13) is true,} \quad (10)$$

where

$$\begin{cases} \mu_i = |\mathcal{N}_i| > 1 \\ \mu_l = |\mathcal{N}_l| > 1 \end{cases} \text{ and } \begin{cases} i \in \mathcal{N}_l^c,\ l \in \mathcal{N}_i^c, \\ \Delta_{i,i'}^{(il)} = \Delta_{l,l'}^{(li)} = -1 \\ \omega_{i,i'} = \max_k \omega_{i,k}, \\ \omega_{l,l'} = \max_k \omega_{l,k}, \end{cases}, \quad (11)$$

$$\begin{cases} \mu_i = |\mathcal{N}_i| = 1 \\ \mu_l = |\mathcal{N}_l| > 1 \end{cases} \text{ and } \begin{cases} i \in \mathcal{N}_l^c,\ l \in \mathcal{N}_i^c, \\ \Delta_{l,l'}^{(li)} = -1 \\ i' = i, \\ \omega_{l,l'} = \max_k \omega_{l,k}, \end{cases}, \quad (12)$$

$$\begin{cases} \mu_i = |\mathcal{N}_i| > 1 \\ \mu_l = |\mathcal{N}_l| = 1 \end{cases} \text{ and } \begin{cases} i \in \mathcal{N}_l^c,\ l \in \mathcal{N}_i^c, \\ \Delta_{i,i'}^{(il)} = -1 \\ \omega_{i,i'} = \max_k \omega_{i,k}, \\ l' = l, \end{cases}, \quad (13)$$

and $\mathcal{N}^r = \cup_{i \in \mathcal{N}} \mathcal{N}_i^r$ is the *augmentation action set* to be found distributively. Note that $\mu_i = |\mathcal{N}_i| = \mu_l = |\mathcal{N}_l| = 1$ is the trivial case of a two-node network and hence is excluded from consideration.

The next $(n-1)$ steps of the Algorithm 4 is to propagate $\mathcal{N}_i^r$ to all the nodes so they all have access to $\mathcal{N}^r$ as follows:

$$\mathcal{N}_i^r(k+1) = \cup_{l \in \mathcal{N}_i \cup \{i\}} \mathcal{N}_l^r(k), \quad (14)$$

where $k = (2n+2), \cdots, (3n+1)$, $\cup$ is the union operation of sets containing non-ordering pairs (that is, if $\{i, j\} \subset \mathcal{N}_l^r(k)$, then $\{j, i\} \subset \mathcal{N}_l^r(k)$), and $\mathcal{N}_i^r(3n+2) = \mathcal{N}_i^r$ is given by (10).

And, as the final step, edge addition is accomplished by the pairs of nodes identified in $\mathcal{N}^r$ to complete their connection.

Remark: While the proposed edge addition algorithm focuses on reconnecting farthest nodes, alternative methods such as random edge addition or distributed optimization algorithms could be explored to achieve similar goals while potentially reducing the risk of long communication paths, though the current algorithm prioritizes robust reconnection over minimizing the length or the number of edges added.

Theorem 3: *Consider connected network $\mathcal{G}$ in which each node executes distributed n-step algorithm represented by equations* (10) *and* (14)*, following the distributed $2n+2$-step algorithms, 1,2 and 3. Then, the resulting network has no critical edge or nodes.*

Proof: The algorithm of (10) and (14) ensures there is no critical edge. Hence, the resulting network has no vulnerability under one link failure anywhere in the network. □

Algorithm B Distributed Edge Addition

Input: Node with unique index i: network size n and neighbor set $\mathcal{N}_i$.

1: Run Algorithm 1 and 2 to ensure connectivity **Result:** *For node i, $\omega_i(2n)$ provides its neighbor structure.*

2: For each of its neighbors (i.e., $l \in \mathcal{N}_i$), calculate $\Delta_{i,j}^{(il)}$ using (6) and locally identify critical edges using (9). **Result:** *At the $(2n+2)$nd step local determination of critical edges/nodes.*

3: If i is a critical node, use (10) to determine the augmentation nodes for each pair of its critical node neighbors. **Result:** *At the $(2n+3)$th step, all the augmentation nodes are locally identified as $\mathcal{N}_i^r$.*

4: Use max-consensus protocol (14) to distributively determine set $\mathcal{N}^r$. **Result:** *At the $(3n+2)$nd step, every pair of augmentation nodes knows the need to add an edge between each other.*

5: If i is in $\mathcal{N}^r$, reach out to its pair l to make an edge. **Result:** *At the $(3n+3)$rd step, edge addition is complete.*

Output: Loop the above steps and the resulting network is connected and has no critical node/edge to improve robustness.

E. Complexity and Robustness

All four algorithms proposed in this paper operate in a finite number of steps proportional to n. Specifically, Algorithm B completes its execution in $3n+3$ steps, indicating that the overall time complexity of the method is $\mathcal{O}(n)$. This complexity is derived from the distributed nature of the algorithms, where each node performs computations in parallel across the network. Since each step involves linear updates to n-dimensional vectors according to the previous and current time instants, the computational effort per node is linear, ensuring that the time complexity remains $\mathcal{O}(n)$.

The system's robustness can be measured in terms of algebraic connectivity, which is the second smallest eigenvalue (λ_2) of the graph Laplacian matrix. A higher λ_2 indicates better connectivity. Additionally, edge connectivity provides another critical measure of the system's robustness. This measure denoted as $\kappa'(\mathcal{G})$ for a network $\mathcal{G}$, represents the minimum number of edges that must be removed to disconnect the network [16].

Example 3: For $\mathcal{G}_1$ in Fig. 1 (a) with critical edges, the $\lambda_2 = 0.5083$ and for $\mathcal{G}_2$ in Fig. 1 (b) without critical edges $\lambda_2 = 0.7933$. Similarly $\kappa'(\mathcal{G}_1)$ is 1, while $\kappa'(\mathcal{G}_2)$ is 2. △

IV. Illustrative Example

The following section presents an example and simulation-based demonstration of the proposed algorithms. Consider a network with 15 nodes, connected in a graph, say $\mathcal{G}_e$ as shown in 3, with a total of 20 edges.

The Algorithm 1 executes for 15 steps. By the 15^{th} step, each node i determines its neighbor structure of the network $\mathcal{G}_e$ through equation (2). It also identified that $\mathcal{G}_e$ is not connected. Then within the next 15 steps Algorithm 2 adds a new edge to establish connectivity, as shown in Fig. 4(a).

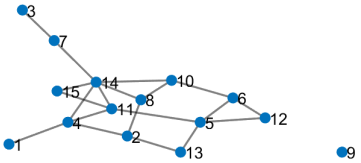
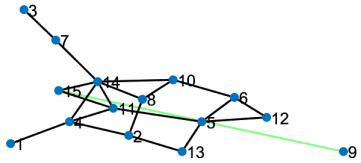
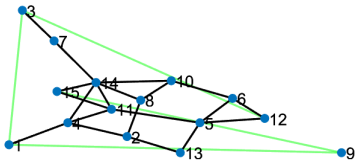


Fig. 3: A random network $\mathcal{G}_e$, of 15 nodes and 20 edges



(a)



(b)

Fig. 4: Augmented network $\mathcal{G}'_e$, as a result of (a) Algorithm 2 (b) Algorithm 4

In the next 2 steps, each node i identifies the critical edges connected to i. The critical edges identified by each node i are as follows:

$$\mathcal{N}_1^c = \{(1,4)\} \quad \mathcal{N}_3^c = \{(3,7)\} \quad \mathcal{N}_4^c = \{(4,1)\}$$
$$\mathcal{N}_7^c = \{(7,3)\} \quad \mathcal{N}_7^c = \{(7,14)\} \quad \mathcal{N}_9^c = \{(9,15)\}$$
$$\mathcal{N}_{14}^c = \{(14,7)\} \quad \mathcal{N}_{15}^c = \{(15,19)\}$$

and for the rest of the nodes $\mathcal{N}_i^c = \{\emptyset\}$

The next 1 step (33^{rd} step) is to identify the corresponding augmentation nodes, which is performed only by the nodes associated with the critical edges. That is, nodes $\{1,3,4,7,9,14,15\}$ and it follows:

$$\mathcal{N}_1^r = \{(1,3)\} \quad \mathcal{N}_3^r = \{(3,12)\} \quad \mathcal{N}_4^r = \{(3,1)\}$$
$$\mathcal{N}_7^r = \{(3,12) \quad \mathcal{N}_9^r = \{(9,1)\} \quad \mathcal{N}_{14}^r = \{(12,3)\}$$
$$(12,3)\} \quad \mathcal{N}_{15}^r = \{(1,9)\}$$

In the following 14 steps, the information is propagated to all the other nodes, and the augmentation node set for each node is $\mathcal{N}_i^r = \{(1,3),(3,12),(1,9)\}$.

In the final 48^{th} step, every node present in $\mathcal{N}^r$ reaches out to its pair and forms the edge. The augmented network is shown in Fig. 4.

Conclusion

In conclusion, this paper introduces a novel distributed method to enhance edge connectivity by identifying/achieving connectivity and reinforcing/removing critical edges. The method includes a four-stage process of neighboring structure discovery, connectivity assurance, critical edge identification, and fortification of critical edges. Each of the four stages involves a distributed algorithm, and the result is that the edge connectivity is improved from 0 or 1 to 2. Hence, the four distributed algorithms together significantly improve the robustness of any undirected network, and the total computational complexity is of $\mathcal{O}(n)$. The only global knowledge required is the network's total number n of nodes, which are uniquely indexed from 1 to n. Future work may consider edge connectivity of more than 2 (i.e., p-order edge connectivity), as this may offer insights into the algorithm's performance under more complex scenarios and potential applications in more robust and fault-tolerant network designs.

References

[1] A.-K. Wu, L. Tian, and Y.-Y. Liu, "Bridges in complex networks," *Phys. Rev. E*, vol. 97, p. 012307, Jan 2018.

[2] L. Dhulipala, G. E. Blelloch, and J. Shun, "Theoretically efficient parallel graph algorithms can be fast and scalable," in *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 393–404.

[3] P. Chaudhuri, "An optimal distributed algorithm for finding articulation points in a network," *Computer Communications*, vol. 21, no. 18, pp. 1707–1715, 1998.

[4] 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, 2022.

[5] M. A. P. Taylor and G. M. D'Este, *Transport Network Vulnerability: a Method for Diagnosis of Critical Locations in Transport Infrastructure Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 9–30.

[6] T. H. Grubesic, T. C. Matisziw, A. T. Murray, and D. Snediker, "Comparative approaches for assessing network vulnerability," *International regional science review*, vol. 31, no. 1, pp. 88–112, 2008.

[7] S. Freitas, D. Yang, S. Kumar, H. Tong, and D. H. Chau, "Graph vulnerability and robustness: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 5915–5934, 2023.

[8] H. Chan and L. Akoglu, "Optimizing network robustness by edge rewiring: a general framework," *Data Min. Knowl. Discov.*, vol. 30, no. 5, p. 1395–1425, sep 2016.

[9] J. S. Baras and P. Hovareshti, "Efficient and robust communication topologies for distributed decision making in networked systems," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, 2009, pp. 3751–3756.

[10] U. Bodkhe, D. Mehta, S. Tanwar, P. Bhattacharya, P. K. Singh, and W.-C. Hong, "A survey on decentralized consensus mechanisms for cyber physical systems," *IEEE Access*, vol. 8, pp. 54 371–54 401, 2020.

[11] 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*, 2009, pp. 1–7.

[12] M. W. S. Atman and A. Gusrialdi, "Finite-time distributed algorithms for verifying and ensuring strong connectivity of directed networks," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 6, pp. 4379–4392, 2022.

[13] A. Gusrialdi and Z. Qu, *Resilient Hierarchical Networked Control Systems: Secure Controls for Critical Locations and at Edge*. Cham: Springer International Publishing, 2022, pp. 95–119.

[14] J. Stoustrup, A. Annaswamy, A. Chakrabortty, and Z. Qu, *Smart Grid Control: Overview and Research Opportunities*, 1st ed. Springer Publishing Company, Incorporated, 2018.

[15] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, S. Sastry *et al.*, "Challenges for securing cyber physical systems," in *Workshop on future directions in cyber-physical systems security*, vol. 5, no. 1, 2009.

[16] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, vol. 92.