Diyala Journal of Engineering Sciences

Journal homepage: https://djes.info/index.php/djes

ISSN: 1999-8716 (Print); 2616-6909 (Online)

An Adaptive-Intelligent Distance-Aware Approach for Dynamic Network Connectivity

Ahmed M. Jasim[1], Haidar N. Al-Anbagi[2*], Saad Al-Azawi[1], Hamed S. Al-Raweshidy[3]

[1] Department of Computer Engineering, University of Diyala, 32001 Diyala, Iraq.
[2] Department of Communications Engineering, University of Diyala, 32001 Diyala, Iraq.
[3] College of Engineering, Design and Physical Sciences, Brunel University of London, Uxbridge, London, UK

ARTICLE INFO

Article History:
Received: 29/04/2025.
Revised: 15/01/2026.
Accepted: 26/01/2026.
Available online: 15/03/2026.

Keywords:
Adaptive Connectivity
Distance-Constrained Networks
Dynamic Network Topologies.
MST
RNCA .

ABSTRACT

Optimizing network connectivity characterizes a major challenge in different applications, e.g. IoT, where the efficiency of connectivity under constraints like distance and adaptability is essential. The key objective of the Minimum Spanning Tree (MST) methods is to construct a subset of a graph in which every vertex is connected with the least amount of edge weight and without any cycles. As a robust framework for constructing distance-constrained network connectivity, this research proposes a novel algorithm named Recursive Node Connectivity Algorithm (RNCA). RNCA forms a tree-like networking structure motivated by minimum spanning tree (MST) principles, while explicitly considering distance constraints to ensure feasible communications. The RNCA utilizes repeated pruning and reweighting procedures to create cost-effective and distance-compliant network structures that can ensure flexible and scalable connectivity in dynamic contexts. The performance of RNCA is evaluated through conducting simulations with different network scales. The results show that RNCA achieves high adaptability, scalability, and reduced redundancy. RNCA outperforms Kruskal's and Prim's MST algorithms by reducing recomputed links during node updates by up to 80%. It also ensures minimum change in network connections when nodes are added or removed from the original network topology. Therefore, it can be considered as a transformative solution for many modern network applications, such as transportation systems, WSNs, and IoT.

1. INTRODUCTION

The geometric representation of the relationships between the links and connecting nodes is known as network topology. Network topologies need to be performance-efficient, scalable, and cost-effective. Therefore, optimizing network connectivity is a real concern in different applications such as communication systems and distributed computing systems. The Minimum Spanning Tree (MST) algorithms, such as Prim's and Kruskal's algorithms, have been developed to establish reliable and yet reduced-cost links between the network's nodes while excluding cycles [1-3]. However, such algorithms may face major difficulties in modern applications with real-world limitations such as diameter, distance between nodes, or resource allocation. Consequently, MST algorithms have been extended to the Diameter-Constrained Minimum Spanning Tree (DCMST) algorithms to address the distance constraints between any two network nodes and thus reduce the delay. However, these methods might encounter serious difficulties when used on large, dynamic networks [4-6].

In addition, recent improvements in network optimization facilitate the management of dynamic networks. Many scenarios highlight that adaptability is essential in enhancing resource allocation and response time, such as optimal placement of servers in dynamic environments. Moreover, these initiatives emphasize the necessity for algorithms to find a balance between scaling and efficiency combined with adaptability in varying network contexts [7-8].

*Corresponding author's.Email: haider_alanbaky2000@yahoo.com.
DOI: 10.24237/djes.2026.19110

Despite the significant development in network systems, the problem of determining distance in dynamic networks remains a real challenge. Most of the existing solutions focus either on static topologies or on cost-efficiency with no distance constraints. As a result, they may not provide effective solutions for real-world scenarios.

Furthermore, the current strategies have not taken into account the distance constraints between nodes, which can lead to impractical, very long links for some applications, such as wireless sensor networks (WSNs) or the Internet of Things (IoT) [9,10]. Therefore, reliable connectivity algorithms should be designed to connect real-world systems and be suitable for modern applications and scenarios that rely on distance as a basis for connecting their nodes. As such, this paper presents a novel algorithm named the Recursive Node Connectivity Algorithm (RNCA), to deal with such constraints. In summary, the contributions of this paper are as follows:

1- The Recursive Node Connectivity Algorithm (RNCA) is introduced as a novel connection framework. RNCA incorporates the distance restrictions into the MST problem to ensure network connectivity. It also strikes a balance between cost-effectiveness, flexibility, and scalability.
2- The modification of the network is explored by maintaining a minimal change in network connections when nodes are added or removed from the original network topology.
3- Simulation-based valuations are designed and performed to assess the depicted algorithm's performance.

Paper Organization: The structure of this paper is organized as follows. In Section 2, the most important related work is reviewed. Section 3 and 4 present the system model and problem formulation respectively. In Section 5, the RNCA algorithm is introduced. The results of the proposed method are presented in Section 6. Finally, Section 7 draws conclusions based on the findings of this study and discusses potential future work.

2. RELATED WORK

The Minimum Spanning Tree Problem (MSTP) represents a usual problem in network connections that is considered by various optimization algorithms. The MSTP determines the spanning tree of a graph with the minimum total edge weight and ensures that all the vertices are connected without creating cycles. The MSTP was solved for the first time by Otakar Borůvka, Joseph Kruskal, and Robert Clay Prim in 1926, 1956, and 1957, respectively [1-3]. Many vital applications, such as telephone network design, optimization of transportation, and clustering analysis, employed MSTP according to [11]. Moreoevr, MSTP was utilized in unsupervised machine learning (ML), e.g., a density-based clustering algorithm [12]. Many researchers have explored the extensions and the variations of the MSTP to deal with practical issues and improve its utility. For example, the study depicted in [13] discussed the MST with the resource allocation as a discrete and continuous optimization problem. In parallel, the Maximum Leaf Spanning Tree Problem (MLSTP) was optimized by using exact methods [14]. Branch-and-Bound method is another extension of the MSTP that can be used to address the generalized MST problem by clustering the vertices and then constructing a minimum-cost tree with one vertex from each cluster only [15].

Furthermore, the incorporation of MSTP with recent optimization algorithms has led to an innovative solution. For example, the transportation problem and MSTP have been solved by an enhancement version of the Ant Colony optimization method [16].

In [17], dynamic programming was combined with genetic algorithms to solve generalized MSTP, whereas in [18], the MST generation to transport crude oil considering the risks of accidents and the costs of construction was explored. Further, in [19] and [20], specific constraints by developing the Least-Cost MST (LC-MST) method and the minimum shortest path tree problem with delay thresholds have been addressed.

Further improvement of the MSTP has been introduced in [21]. The improvement incorporates that the Dhouib-Matrix-MSTP (DM-MSTP) introduced a matrix-based approach to construct minimum spanning trees. DM-MSTP focus on the improvement of the efficiency without addressing the adaptability to the dynamic network conditions or constraints.

In addition, recent research has investigated the integration of evolutionary methods with network optimization. In [22], for example, several hybrid algorithms based on Biogeography-Based Optimization (BBO) were mentioned to solve the Minimum Spanning Tree Problem (MSTP). These include HG-BBO, HBBOG, TD-BBO, Lx-BBO, DE-BBO, and PR-BBO. Each one has its own way of encoding and searching. These methods can give good performance in terms of cost and convergence. However, due to its nature, it may require a lot of parameter tuning and can give different performance from one run to another. While these algorithms are suitable for static MST problems, they lack native support for distance constraints and dynamic node modifications.

RNCA applies strict edge-level distance constraint, which is extremely important in applications such as WSNs and IoT, in contrast to the traditional DCMST

methods that limit the tree's diameter. Furthermore, most existing approaches either require full or partial re-computation of the spanning structure after topology changes, or they assume static topologies. In contrast, RNCA employs a recursive local pruning and merging method that allows node additions or removals with the least amount of structural disruption.

Despite the great improvement offered by these methods in the MST area, most of them are limited by the inability to adapt to dynamic scenarios, combination of distance constraints, or minimize redundant links. Motivated by the abovementioned limitations, this work suggests a new algorithm named Recursive Node Connectivity Algorithm (RNCA). RNCA addresses these issues by ensuring network adaptability and scalability and meanwhile overcoming distance constraints. The primary distinction between RNCA and DM-MSTP is the fact that the latter focuses on adjacent matrix navigation for efficiency, whereas the former emphasizes the adaptability of vast networks. In addition, RNCA assures the connectivity of the networks under practical limitations by reducing the redundant communication links and dynamically adapting to nodes additions or removals. Therefore, the RNCA can be considered as a robust solution suitable for recent applications that require dynamic updates, distance limits and minimal link redundancy. Thus, the RNCA offers a balance between computational efficiency and practical applicability.

3. SYSTEM MODEL

The proposed system model for optimal network connectivity is shown in Figure 1. The network consists of N distributed nodes placed in a two-dimensional area. Each node serves as a network entity, e.g., a sensor or communication device, and is represented by fixed spatial coordinates (xi, yi).

Direct communication between any two nodes can be established only if they are not apart by a distance greater than a threshold, denoted by D_{MAX}. Accordingly, this limitation reflects practical communication constrains such as transmission range and energy efficiency. Based on how the nodes are distributed, a distance matrix D is computed, where the element D_{ij} is the distance between node i and node j.

Out of the elements of the distance matrix D, a binary connectivity matrix is then derived to symbolize feasible links that satisfy the distance constraint. Only edges with $Dij \leq D_{MAX}$ are considered valid candidates for establishing the network structure. Thus, the objective of the proposed system model is to enable the network to include all nodes fulfilling the distance constraints and maintain reduced redundant connections. The RNCA algorithm proceeds in two stages: (1) recursive local connections among nodes using the sorted nearest-neighbour distances, and (2) merging disjoint groups through optimal inter-cluster links. As a result, the network connected structure is a tree-like topology that backs dynamic operation. The suggested system model is flexible for nodes to be added or removed at any time, necessitating the network connectivity structure to readapt without violating the distance constraints. This model forms the foundation upon which the proposed RNCA algorithm functions to construct and preserves an efficient and scalable network.

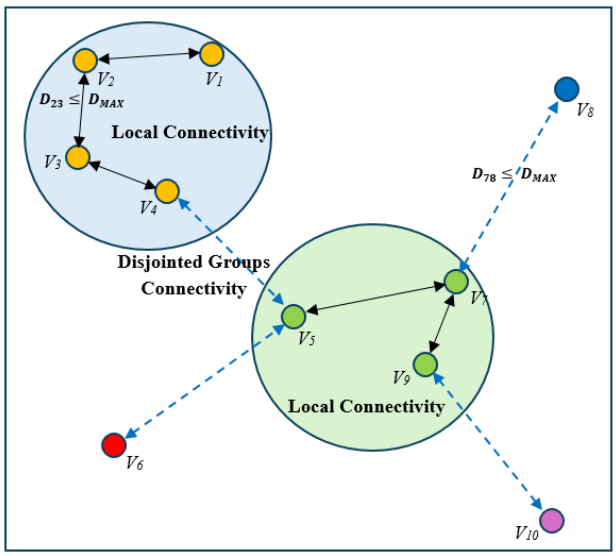

Figure 1. Proposed System Model.

4. PROBLEM FORMULATION

Efficient network connectivity is important for achieving optimal performance, scalability, and cost efficiency in networks applications. Traditional methods, including Kruskal's and Prim's algorithms, aim to minimize the cost of forming an MST. However, in real-world scenarios with distance restrictions, where long-distance connections raise communication costs and impair performance, such methods are not well suited. Additionally, traditional MST methods are characterized by:

1- They do not inherently consider heterogeneous cost metrics (e.g., distance, deployment cost).

2- They Lack adaptability to the dynamic nature of networks with varying topologies.

4.1 Problem Description

Given a network structure demonstrated as a graph $G = (V, E)$:

- V: denotes a set of nodes (e.g., sensors, servers, …..).
- E: indicates a set of edges (e.g., possible links between nodes).
- $d(e)$: refers to distance of edge $e \in E$.

- $c(e)$: designates the cost of edge $e \in E$, and is dependent on distance or any other physical constraints.

Then, the goal is to establish a distance-constrained and cost-efficient network architecture, specifically to build a connected subgraph $G' = (V', E')$ that can be characterized by:

1- Minimizes the total cost of selected edges.
2- Adheres to a maximum distance constraint D_{MAX} on all selected edges.
3- Ensures network connectivity across all nodes.

4.2 Mathematical Formulation

Let:

- x_{ij} be a binary decision variable associated with edge $e_{ij} \in E$, defined as:

$$x_{ij} = \begin{cases} 1 & if\ edge\ e_{ij}\ is\ selected, \\ 0 & otherwise \end{cases}$$

- $c(e_{ij})$: refers to cost of edge e_{ij}.
- $d(e_{ij})$: stands for distance of edge e_{ij}.

Objective: The objective is to structure a distance-constrained network architecture while maintaining as low number of selected edges as possible. Thus, the total connection cost:

$$min \sum_{e_{ij} \in E'} x_{ij} \quad (1)$$

$$min \sum_{e_{ij} \in E'} c(e_{ij}) \cdot x_{ij} \quad (2)$$

Subject to:

1- Distance constraint:

$d(e_{ij}) \leq D_{max} \quad \forall e_{ij} \in E'$ with $x_{ij} = 1$

2- Connectivity constraint:

The selected subgraph $G' = (V', E')$, where $E' = \{e_{ij} \in E \mid x_{ij} = 1\}$, must be connected and span all nodes in V. Connectivity is ensured by constructing a spanning tree that includes all nodes.

3- Minimum edge constraint:

All nodes $v \in V$ must be connected by at least one edge in E':

$$\sum_{e_{ij} \in E'} x_{ij} \geq |V| - 1$$

4- Variable domain constraint:

$$x_{ij} \in \{0,1\} \quad \forall e_{ij} \in E$$

This formulation defines the connectivity problem that RNCA is intended to address. RNCA creates a practical solution that satisfies these constraints without explicitly solving the optimization problem mentioned above.

5. THE RECURSIVE NODE CONNECTIVITY ALGORITHM (RNCA)

RNCA handles the limitations of the existing traditional algorithms by enforcing edge length limits while maintaining reduced costs. It uses the strategy of recursive pruning and reweighting, resulting in distant edges removal and edge weights adjustment to enhance the network connectivity. Thus, RNCA presents a robust framework for optimizing constrained networks by achieving balance, scalable, and cost-effective structures. If compared to the traditional MST and DCMST methods, RNCA is very unique. Since RNCA does not involve the optimization of a single global objective function, it is not a conventional MST. Instead, RNCA gradually builds connectivity according to local feasibility. Furthermore, RNCA maintains network structure and enables localized updates without the need for a full re-computation. Finally, unlike DCMST methods that limit path diameter, RNCA imposes a strict distance constraint, which makes it appropriate for spatial and dynamic network scenarios. The MST algorithms often maintain a global spanning tree by dynamically reweighting edges and pruning or reconnecting links in order to preserve global optimality after topology changes. However, RNCA does not update or maintain a global tree structure. Instead, it enables gradual connectivity construction and repair by making recursive local decisions based on feasibility and distance restrictions. As a result, RNCA focuses on preserving existing, valid connections while addressing connectivity gaps locally, avoiding global reweighting or re-computation.

To provide a comprehensive understanding, the algorithm is explained step by step in the following stages:

5.1 Network Connectivity

Stage 1: Initial Network Connectivity

Step 1: Locate All Coordinates on the Map

At the beginning, RNCA maps and visualizes the coordinates of all nodes in the network. Then, the algorithm produces either random or predefined positions within any region. This initial step ensures a valid and well-structured network map for the algorithm to operate on.

Step 2: Generate Important Matrices

Two vital matrices are generated in this step, the Distance Matrix as well as the Connectivity Matrix. Then, the algorithm triggers the calculation of the spatial proximity between every pair of nodes and accordingly stores it in the Distance Matrix. The generated Distance Matrix forms the baseline for connectivity decisions. Moreover, the Connectivity Matrix shows the direct links between nodes based on D_{MAX}. The elements of the Connectivity Matrix can be either 1 for connected nodes or 0 for unconnected

nodes. Then, all the neighbouring nodes that fulfil the D_{MAX} criteria are connected virtually. These connections represent a base for the subsequent algorithm's steps.

Step 3: Find the Current Connected Nodes

This step generates a matrix known as $Final\_SortedIndicesMatrix$, an organized representation of the network connections. Then, the algorithm ranks the nodes based on their connectivity degree from least to most. Accordingly, each matrix row contains a node number in the first column while the following columns correspond to the connected nodes. This structure helps facilitate the selecting of the best links and improves further optimization procedures in the subsequent stages.

Step 4: Exclude distant nodes

Before proceeding into further steps with the network configuration, RNCA requires identify the nodes that can never connect because of their remote geographical location. These identified nodes, if any, are then stored in a matrix denoted as $NeverConnectedMatrix$. This step is important to ensure the network focuses on viable connections and avoids extra efforts. Moreover, determining these nodes gives important information for developing alternative tactics such as including them into another network if possible, relaxing some of the constraints (on them only), or finding additional connectivity mechanisms.

Step 5: Find Connected Nodes

In this step, RNCA uses $Final\_SortedIndicesMatrix$ to choose the nodes that connect to each other. For each node in the first column, RNCA selects the best node from other matrix columns based on the proximity. Then the selected best node is removed from the other rows to ensure creating a tree structure. The mechanism of dynamic transition guarantees that when a node is connected to another node, the algorithm immediately focuses on the newly connected node to keep forming connections. This step ensures that each node is only connected to one or two nodes. It also enables RNCA to build efficient and scalable network structure which is ready for further optimization in the later stages by giving the priority to the proximity and enforce unique connections.

Step 6: Find Unconnected Nodes

Before finalizing the network connection, RNCA needs to ensure that all nodes are already connected. Otherwise, it must determine which nodes are still not connected and hold them in a new matrix called $UnconnectedNodesMatrix$. The unconnected nodes are then linked to their closest available neighbours as listed in the $SortedIndicesMatrix$. This step is important to achieve full network connectivity.

Step 7: Connect the current nodes in all groups

The final step is network visualization. RNCA connects nodes within every group using $ConnectedNodesMatrix$ which consists of two columns. Each row includes a node and its neighbor. RNCA iterates through each row, drawing connections between the nodes. Since the last row of the $ConnectedNodesMatrix$ may introduce redundancy or cycles, therefore, the algorithm excludes it.

Stage 2: Final Network Connectivity

Step 1: Identify the formed groups.

In this stage, RNCA uses the structure of the initial connected network (from stage 1) to identify each collection of connected nodes as a separate group. Thus, it determines each group, identifies how many nodes are in each cluster, and divides the network into groups for further processing in order to provide the best possible connection. This step contributes to complete the full network connection.

Step 2: Establish Connections between groups

After identifying the groups, RNCA finds the shortest possible link between any two groups using the $DistancesBtAllNodes$ matrix. The selected pair of nodes represent the best connection that minimizes the overall network distance. Once two groups are merged, RNCA updates the number of groups and the connectivity matrix to reflect the new connections. It also visualizes them by drawing the new links. The process of iteration continues until all groups are connected which forms a single fully connected network. This step maintains the tree structure and ensures efficient and scalable approach to global network connectivity by minimizing the inter-cluster connection distance.

5.2 Adding and Removing Nodes

The features that demonstrate the effectiveness of any algorithm are its ability to adapt and scale to different scenarios. Therefore, to prove the efficiency of this algorithm, it must be capable of adding or removing nodes to/from the existing network. The next sections explain how to add/remove nodes without affecting the structure of the existing network.

Adding New Nodes

Step 1: Locate the New Coordinates on the Map

In this step, RNCA places the new nodes on the map. Then, it visually represents them with distinct markers and labels to recognize them.

Step 2: Calculate Distances and Update Connectivity Matrices

RNCA calculates the distance between the new nodes and all other nodes. Accordingly, the distance matrix is updated into a new matrix called $DistancesBtAllNodes\_NewPlus$. Simultaneously, the connectivity matrix is updated and extended to

reflect connections between the new and existing nodes.

Step 3: Connect New Nodes

After updating the required matrices, RNCA selects the best nodes for the new connection. The best one is the closest node, therefore, the algorithm iterates over each new node, identifies the nearest neighbour to the new node by using the updated matrices. These connections are then visualized on the map to ensure seamless integration of the new nodes into the network.

B. *Removing Nodes*

Step 1: Define the Interrupted Nodes and Update Matrices

After identifying nodes that are no longer functioning due to different reasons, RNCA isolates and removes them from the network. Then, it deletes all associated links. Once these nodes are removed, Distance and Connectivity Matrices are updated to reflect the new change. This step ensures the network structure reflects the changes accurately without much affecting the whole network connectivity.

Step 2: Determine the New Groups

After removing nodes, RNCA re-analyses the network to check the effect of changes on the network structure. Therefore, it identifies any new groups that were recently formed.

Step 3: Find the best links to connect the New Groups

This step finalizes the reconstruction process of the network. RNCA iteratively identifies the shortest possible link between the resulted new distinct groups and establishes a connection to restore full connectivity. Therefore, it ensures full connectivity with minimum additional distance by updating the group structure and the matrix of connectivity after each connection. RNCA is presented in Algorithm 1.

Algorithm 1: Recursive Node Connectivity Algorithm (RNCA)

Input: N: Number of initial nodes.
D_{MAX}: Maximum allowable distance for direct connections.
$Coordinates\ (x, y)$: Positions of the nodes in a 2D space.
$NewNodes$: Number of additional nodes to add.
$NodesToRemove$: List of nodes to remove
Output: A fully connected and scalable network.

```
/* Initialization */
Compute distance matrix D where
    D[i][j] ← point i – point j if i ≠ j, else ∞
Construct connectivity matrix C such that
    C[i][j] ← 1 if D[i][j] ≤ D_MAX:, else 0
/* Stage 1: Local Node Connectivity */
For each node i ∈ V do
    Sort neighboring nodes by increasing D[i][·]
end for
Initialize ConnectedNodesMatrix ← ∅
Select nodes in ascending order of connectivity degree
For each selected node i do
    Connect i to its nearest feasible neighbor j
    Avoid cycles and redundant connections
end for
Identify isolated nodes and store in NeverConnectedNodes
/* Stage 2: Cluster Connectivity */
Identify connected components (clusters) in the graph
While number of clusters > 1 do
    For each pair of clusters (G_a, G_b) do
        Find (u, v) with u ∈ G_a, v ∈ G_b minimizing D[u][v]
    end for
    Connect the closest cluster pair
    Merge clusters
end while
/* Add New Nodes */
For each new node k do
    Compute distances to all existing nodes
    Connect k to the nearest feasible node
```

30: end for
31: /* Remove Nodes */
32: Remove all nodes in NodesToRemove and their incident edges
33: Update distance and connectivity matrices
34: Identify new clusters
35: Reconnect clusters using minimum-distance links
36: Return ConnectedNodesMatrix and network metrics

5.3 Complexity and Correctness Analysis

To evaluate the efficiency and reliability of the proposed Recursive Node Connectivity Algorithm (RNCA), both time complexity and structural correctness are analysed.

5.3.1 Time Complexity

RNCA consists of two main phases: local node connection and inter-cluster merging. The dominant operations are:

- Distance matrix computation: $O(n^2)$.
- Sorting neighbors per node: $O(n \log n)$.
- Recursive node linking and pruning: $O(n \log n)$.
- Cluster merging operations: up to $O(n^2)$.

Thus, the overall worst-case time complexity is $O(n^2 \log n)$.

5.3.2 Space Complexity

RNCA needs several matrices such as distance, connectivity, and index matrices, all of which scale quadratically with the number of nodes. Therefore, the space complexity is $O(n^2)$.

5.3.3 Correctness Justification

The following is guaranteed by RNCA:

- Full connectivity: All nodes are connected
- Distance constraint: No link exceeds D_{MAX}.
- Redundancy avoidance.
- Scalability: RNCA offers the possibility to add or remove nodes with minimal disruption.

These features can efficiently and adaptively ensure that the algorithm is capable of building a valid, distance-constrained spanning structure.

6. RESULTS AND DISCUSSION

This section navigates through the collected results of multiple simulations scenarios conducted to evaluate the performance of the suggested RNCA. The presented evaluation results are based on randomly generated node locations, which are represented as points within a defined region. To better visualize the efficiency of the proposed algorithm, a step-by-step process is illustrated using an initial configuration of 16 nodes. Subsequently, the final outcomes are showcased for larger networks with 50 and 100 nodes, as listed in Table 1.

Table 1. RNCA simulation parameters

Parameter	value
Area	100 km × 100 km
Number of nodes	16, 50, 100
Coordinates	Uniform random (2D plane)
Max Distance (D_{MAX})	40 30 km

6.1 Scenario: 1 - 16 nodes

In this scenario, a MATLAB simulation was developed to randomly generate 16 points within a proposed (100 ×100) km geographical area. The initial step is to visualize the nodes spatial distribution within the region by mapping them onto a virtual map to. Figure 2 illustrates the arrangement of the nodes on the map.

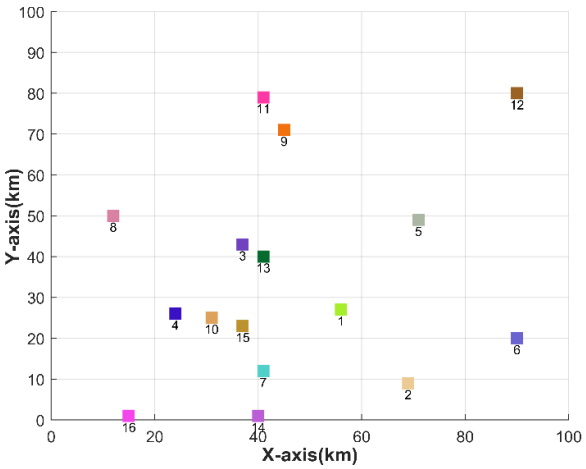

Figure 2. The spatial distribution of 16 initially unconnected nodes.

The next task is to calculate the distance and connectivity matrices for the generated nodes and then establish the virtual connections between them, considering the distance constraint D_{MAX}:. As Figure 3 reveals, the initial configuration showed multiple redundant communication links between certain nodes. On the other hand, the primary objective of the algorithm presented in this work was to get these redundant connections reduced while ensuring full connectivity throughout the whole network.

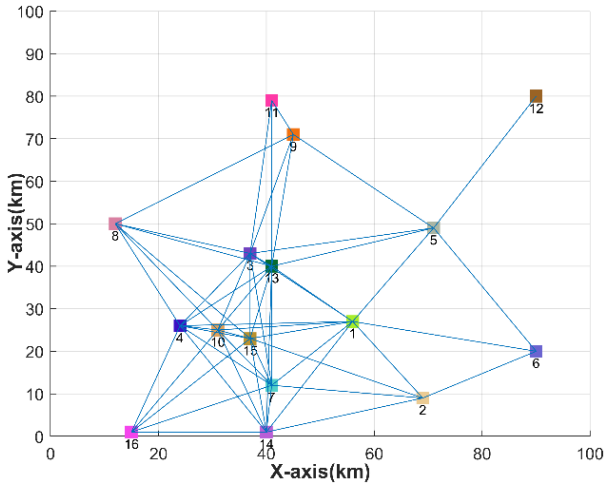

Figure 3. A graph of the initial virtual connectivity of the generated 16 nodes (D_{MAX} = 40 km).

The next step is to formulate the $Final\_SortedIndicesMatrix$, which contains all nodes along with their associated connected nodes, sorted by proximity from closest to farthermost. For example, node 6 is connected to three nodes: node 2 (the closest to node 6), node 5, and node 1 (the farthest to node 6). Equivalently, node 13 is connected to eleven nodes: (3, 15, 10, 1, 4, 7, 8, 9, 5, 11, and 14).

In addition, RNCA sorted the matrix elements from the least connected to the most connected nodes. Doing so guarantees that the algorithm starts up building connections from the far edges of the network, which helps avoid redundant communication links and cycles. In the subsequent steps, the $Final\_SortedIndicesMatrix$ serves as a main tool for establishing the network's connections. Table 2 shows the final sorted indices matrix.

Next is to exclude distant nodes that are too far away and hence cannot be connected to any other nodes. Therefore, the $Final\_SortedIndicesMatrix$, does not include nodes classified as "Never Connected".

RNCA then generates the $ConnectedNodesMatrix$, which forms a representation of the actual connectivity between nodes. The whole process starts with node 5, the first node as identified in Table 2. RNCA selects Node 1 as the closest node to establish a connection to node 5 and marks node 1 as the current node. Later, it moves to node 1 in the table to find out the next optimal connection.

Concurrently, RNCA removes node 1 from all other rows to avoid redundant links. It is worth mentioning that since the algorithm requires one fewer link than the total number of nodes, it eliminates the last connection—node 13 with node 7—to avoid possible overlaps or cycles within the network.

Table 2. $Final\_SortedIndicesMatrix$

Nodes	Correlated nodes										
12	5										
6	2	5	1								
11	9	3	13								
2	1	6	7	14	15						
9	11	3	13	5	8						
16	14	4	7	10	15						
5	1	13	9	3	6	12					
8	3	4	13	10	15	9					
14	7	15	16	10	4	2	1	13			
4	10	15	3	7	13	16	8	14	1		
7	14	15	10	1	4	13	2	16	3		
10	15	4	7	13	3	1	14	16	8		
1	15	13	7	2	3	10	5	14	4	6	
3	13	10	15	4	1	8	9	7	5	11	
15	10	7	4	13	1	3	14	16	2	8	
13	3	15	10	1	4	7	8	9	5	11	14

The next step involves checking all nodes to identify any unconnected nodes. As shown in Table 3, all nodes are individually connected, achieving node-level connectivity. However, the whole network may still not be fully connected, which will be addressed in Stage 2. The current network structure is illustrated in Figure 4, highlighting the updated connections following all preceding steps in Stage 1.

Table 3 $ConnectedNodesMatrix$

Node	selected node
Start 1 → 12	5
5	1
1	15
15	10
10	4
4	3
3	13
13	(ignored – last node in the matrix)
Start 2 → 6	2
2	None (vertex)
Start 3 → 11	9
9	8
8	None (vertex)
Start 4 → 16	14
14	7
4	None (vertex)

In Stage 2, RNCA identifies and classifies the formed groups of nodes. Based on the analysis, four distinct groups are determined: Group 1 (Nodes 11, 9, 8), Group 2 (Nodes 12, 5, 1, 15, 10, 4, 3, 13), Group 3 (Nodes 6, 2), and Group 4 (Nodes 7, 14, 16), as shown in Figure 4. Then, RNCA performs a number of procedures to find the shortest distance between the nodes from different groups and connects them. Then, the number of groups is updated. The forementioned process continues until all other groups are merged into a single one, assuring that the whole network is fully connected. The final fully connected network architecture as well as the demonstration on how well

RNCA integrates various groups into a cohesive network is shown in Figure 5.

It is vital to recall that RNCA created four different groups of nodes with 12 links during Stage 1. Then, tt successfully added three more communication links in Stage 2 to merge the groups into a single one resulting in a fully connected network by reaching the necessary 15 links.

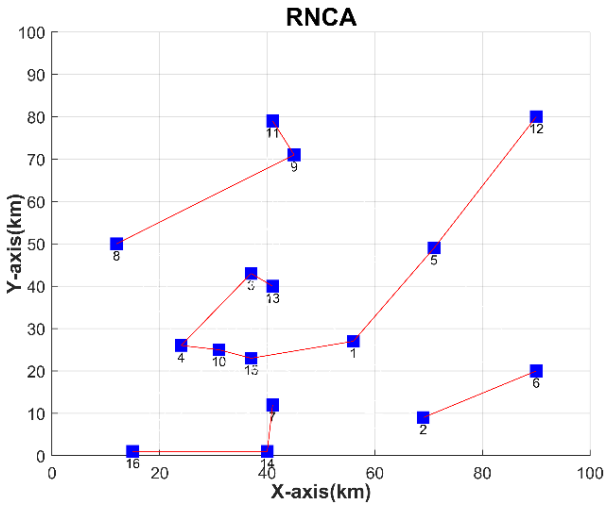

Figure 4. Locally connected network obtained after Stage 1 of RNCA (D_{MAX} = 40 km).

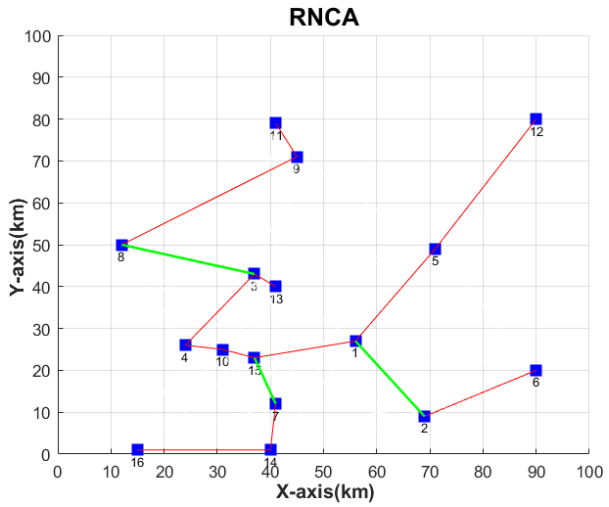

Figure. 5. Final fully connected network after Stage 2 of RNCA (D_{MAX} = 40 km).

This outcome can clearly prove how effective and intelligent the RNCA is in optimizing the network connectivity. RNCA shows its powerfulness to function accurately and professionally by methodically linking the closest nodes between groups. The final structure of the connected network affirms that RNCA retains the requirements of distance-constrained connectivity optimization.

6.1.1 Adding new nodes

In this stage, new nodes are added to the existing network to test and demonstrate RNCA's adaptability and scalability. The new nodes are combined by RNCA in three simple stages apart from the current network structure. First is to place the new nodes on the network map. Second is to determine the distances between the new nodes and every active node in the network. Lastly, RNCA establishes a connection between each node to its closest active node.

This stage can demonstrate the capability of RNCA's to maintain network functionality while adapting to dynamic changes. An example of such test is performed for adding three additional nodes (Nodes 17, 18, and 19) to the network as shown in Figure 6.

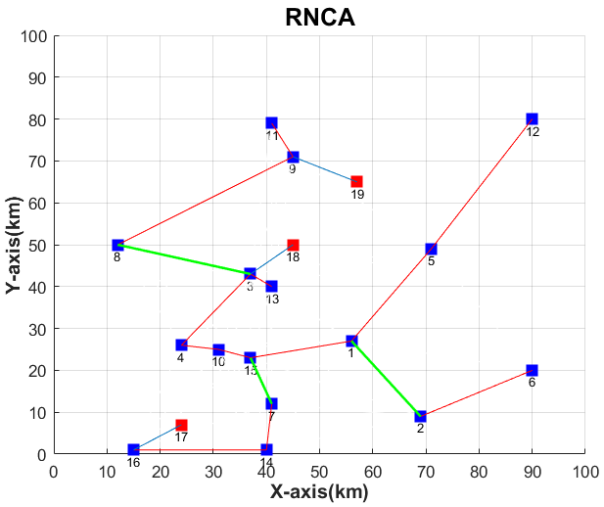

Figure 6. Updated network topology after adding three new nodes (17, 18 and 19).

6.1.2 Removing nodes

This stage confirms RNCA's robustness and adaptability in case of removing nodes from the network for various reasons. This case provides new evidence of the algorithm's ability to keep the network connected under dynamic conditions. Figure 7 provides an example of removing Nodes 15 and 16 from the network. RNCA needs to re-evaluate the network structure after this disruption to perform the required changes. The removal of node 16, which is at the edge of the network, has no appreciable impact on other nodes. However, the removal of Node 15, which held a strategic position in the network, is more challenging. Three nodes which are Nodes 1, 7, and 10 were connected to Node 15. Therefore, the removal of Node 15 splits the network into multi groups. Accordingly, RNCA updates the relevant matrices, such as Distance and Connectivity matrices, to reconnect the whole network. Consequently, it creates new connections between the new groups using the shortest paths. The revised structure with the updated connections is shown in Figure 7. It is important to

notice that the removal of two nodes updated the necessary links only while preserving the majority structure intact.

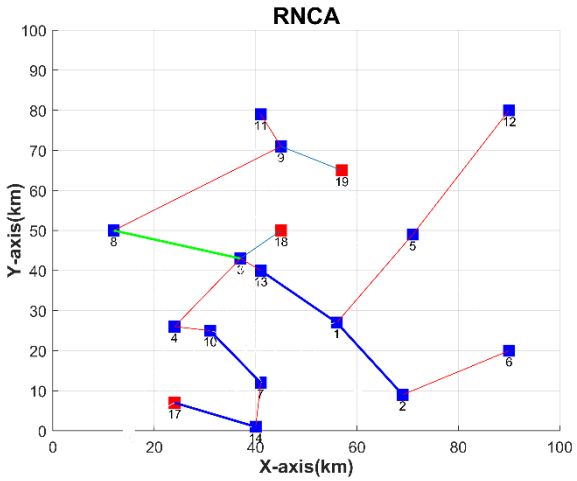

Figure 7. Reconstructed network topology after removing two nodes (15 and 16).

The number of recomputed links needed following node addition or removal is reported in Table 4 to provide quantitative evidence for adaptability and scalability claims. RNCA restricts updates to local neighbourhoods, leading to substantially fewer modified connections than traditional MST-based methods that necessitate full or partial recomputation. This behaviour demonstrates that RNCA is appropriate for dynamic network environments with frequent topology changes. The amount of structural disruption each algorithm causes is directly reflected in the number of recomputed links, which is determined by counting the edges added and removed during topology updates.

Table 4. Quantitative comparison under node addition/removal for 16-nodes network

Algorithm	No. of added/removed nodes	No. of Links Recomputed	Full Rebuild
Kruskal MST	+3 nodes	18	Yes
Prim MST	+3 nodes	18	Yes
DM-MSTP	+3 nodes	~11	Partial
RNCA	+3 nodes	3	No
RNCA	−2 nodes	4	No

6.2 Scenario (2 and 3) – 50 and 100 nodes

This section evaluates two more networks with 50 and 100 nodes in order to confirm the efficiency of RNCA. Figures 8 and 9 illustrate the connectivity results, which demonstrate the RNCA's ability to achieve full connectivity in increasingly complex networks.

Figure 8 shows the final fully connected network of 50 nodes distributed in an area of 200 km × 100 km. The RNCA created five different interconnected groups during the first stage after identifying unconnected nodes (highlighted in green). In order to achieve a fully connected network, the RNCA then successfully combined these groups in the second stage using the shortest links.

The final fully connected network with 100 nodes is shown in Figure 9. Once again, RNCA was able to first identify several groups of interconnected nodes and then connect them using the shortest links despite the increasing complexity. This results further confirm the the scalability and effectiveness of RNCA when working with larger networks.

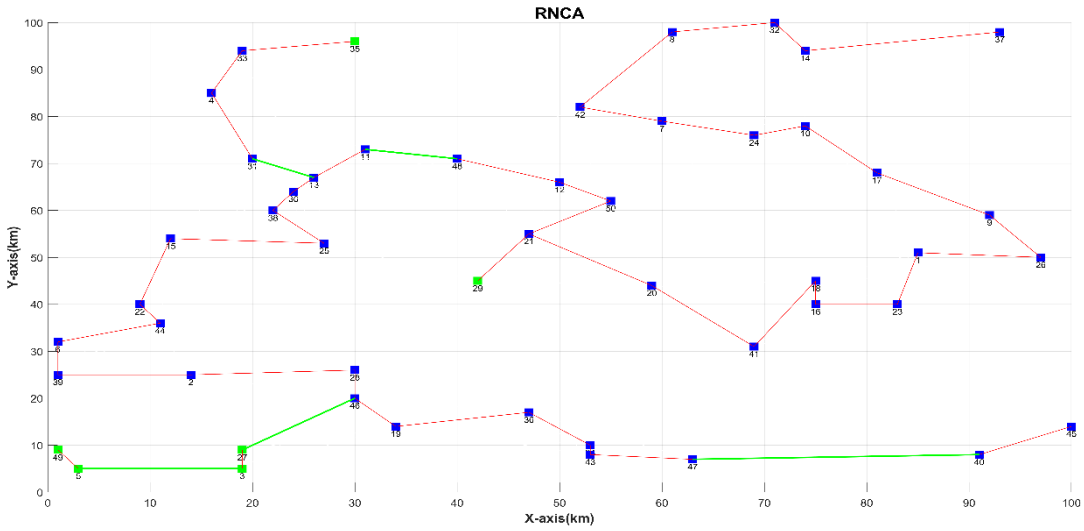

Figure 8. Fully connected network of 50 nodes (D_{MAX} = 30 km).

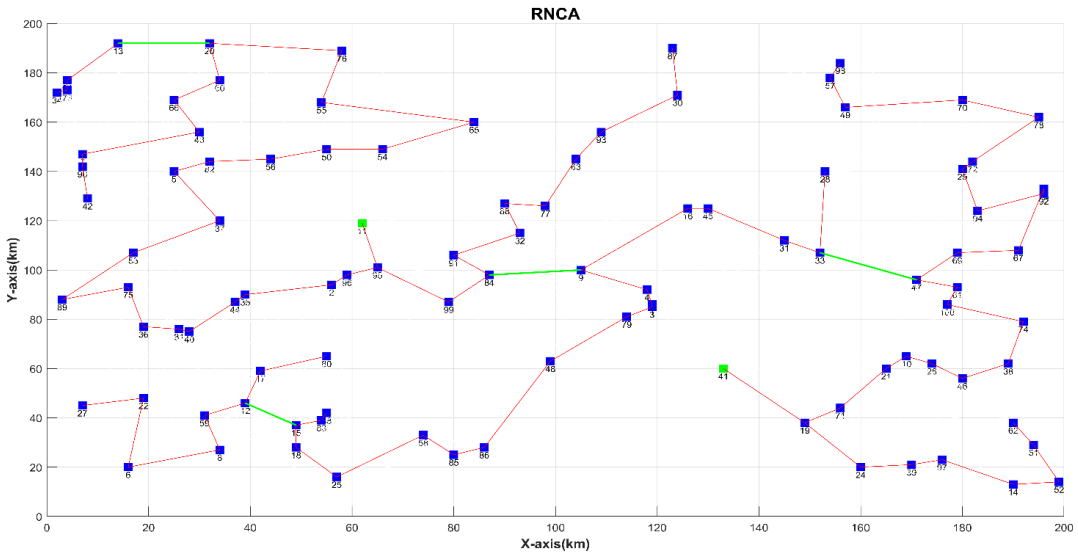

Figure 9. Fully connected network of 100 nodes (D_{MAX} = 30 km).

6.3 Comparison Between RNCA and Existing Algorithms

Table 5 presents a comparison between the proposed Recursive Node Connectivity Algorithm (RNCA) and several well-known network connectivity optimization techniques, including Kruskal's MST, Prim's MST, DM-MSTP, K-Nearest Neighbor (KNN), and Biogeography-Based Optimization (BBO)-based algorithms mentioned in [21].

In terms of redundancy reduction, distance constraint enforcement, and adaptability, RNCA performs noticeably better than the other algorithms. Thanks to its recursive and modular design, RNCA is more suitable for dynamic networks than DM-MSTP, which is based on adjacency matrix navigation in static conditions. Moreover, RNCA offers deterministic, low-cost, and highly adaptive solutions without high computational complexity, unlike BBO-based algorithms [21], which are stochastic and require fine-tuning.

For recent network applications where connectivity must be continuously maintained under shifting topological and spatial conditions, like the Internet of Things, WSNs, and extensive transportation systems, RNCA represent a suitable option.

Table 5. A comparison between RNCA and other algorithms

Criteria	Distance Constraints	Adaptability	Scalability	Connections Required	Redundant Link Avoidance	Average Degree (16 Nodes)	Dynamic Application Suitability
Kruskal's MST	No	Low	High (static networks)	V−1	Moderate	1.93	No
Prim's MST	No	Low	Moderate (static networks)	V−1	Moderate	1.93	No
KNN	Partially	Moderate	Moderate	Variable	Low	2.50	Possible
BBO	No	Variable	Moderate to high	Variable	Variable	Variable	Possible
DM-MSTP	Partially	Moderate	High (static networks)	V−1	High	2.10	No
Proposed RNCA	**Yes**	**High**	**High**	**V−1**	**High**	**1.87**	**Yes**

7. CONCLUSION

This paper presented a novel solution for optimum network connectivity, named Recursive Node Connectivity Algorithm (RNCA). RNCA addresses the drawbacks of the existing algorithms by enforcing edge length limits through the use of a recursive pruning and reweighting strategy. RNCA outperforms the existing conventional algorithms in terms of adaptability and scalability. It successfully reacts to modifications, including adding and

removing nodes to the network architecture. Comprehensive simulations scenarios were conducted to test and evaluate the performance of the suggested RNCA algorithm. The results proved how powerful and effective the algorithm is not only in avoiding redundant links but also in adapting to dynamic changes in network nodes additions or removals. In comparison to the conventional MST-based techniques, the proposed RNCA algorithm demands substantially fewer structural updates, minimizing link re-computation by up to 80% during network topology changes. Thus, RNCA can be deemed as a crucial tool for optimizing the connectivity of current and future modern networks, where resources allocation efficiency and adaptability are critical. As for the future directions to develop the presented algorithm, RNCA can be extended to focus on addressing additional issues such as latency and energy efficiency.

REFERENCES

[1] Nešetřil, J., Milková, E., & Nešetřilová, H. (2001). Otakar Borůvka on minimum spanning tree problem: Translation of both the 1926 papers, comments, history. Discrete Mathematics, 233(1–3), 3–36. https://doi.org/10.1016/S0012-365X(00)00224-7.

[2] Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society, 7(1), 48–50. https://doi.org/10.1090/S0002-9939-1956-0078686-7.

[3] Prim, R. C. (1957). Shortest connection networks and some generalizations. The Bell System Technical Journal, 36(6), 1389–1401. https://doi.org/10.1002/j.1538-7305.1957.tb01515.x.

[4] Dadalto, A. P., Usberti, F. L., & Felice, M. C. S. (2023). Exact approaches for the Minimum Subgraph Diameter Problem. Computers & Operations Research, 150, Article 106050. https://doi.org/10.1016/j.cor.2022.106050.

[5] Gouveia, L., Leitner, M., & Ljubić, I. (2020). A polyhedral study of the diameter constrained minimum spanning tree problem. Discrete Applied Mathematics, 285, 364–379. https://doi.org/10.1016/j.dam.2020.05.020.

[6] Vuppuluri, P. P., & Chellapilla, P. (2021). Serial and parallel memetic algorithms for the bounded diameter minimum spanning tree problem. Expert Systems, 38, Article e12610. https://doi.org/10.1111/exsy.12610.

[7] Jasim, A. M., & Al-Raweshidy, H. (2024). An adaptive SDN-based load balancing method for edge/fog-based real-time healthcare systems. IEEE Systems Journal, 18(2), 1139–1150. https://doi.org/10.1109/JSYST.2024.3402156.

[8] Jasim, A. M., & Al-Raweshidy, H. (2024). Optimal intelligent edge-servers placement in the healthcare field. IET Networks, 13(1), 13–27. https://doi.org/10.1049/ntw2.12097.

[9] Al-Anbagi, H. N., & Vertat, I. (2023). Embracing small satellites into future 6G inclusive IoT coverage: A deployment of diversity-based theoretical framework. IEEE Access, 11, 114602–114612. https://doi.org/10.1109/ACCESS.2023.3324890.

[10] J. S. Abdullah, M. I. Aal-Nouman, and B. Al-Fuhaidi, "Efficient coverage of sensors in a WSN using a modified hybrid PSO and ALO algorithms," Iraqi Journal of Information and Communication Technology, vol. 8, no. 2, pp. 29–41, 2025, doi: 10.31987/ijict.8.2.309.

[11] Mahdi, S. B., Al-Bayati, H. H. J., & Al-Mahdawi, J. A. (2022). Minimum spanning tree application in COVID-19 network structure analysis in the countries of the Middle East. Journal of Discrete Mathematical Sciences and Cryptography, 25(8), 2723–2728. https://doi.org/10.1080/09720529.2022.2060923.

[12] Gao, Q., Gao, Q. Q., Xiong, Z. Y., Zhang, Y. F., & Zhang, M. (2022). A novel minimum spanning tree clustering algorithm based on density core. Computational Intelligence and Neuroscience, 2022, Article 8496265. https://doi.org/10.1155/2022/8496265.

[13] Kataoka, S., & Yamada, T. (2016). Algorithms for the minimum spanning tree problem with resource allocation. Operations Research Perspectives, 3, 5–13. https://doi.org/10.1016/j.orp.2015.12.001.

[14] Fernau, H., Kneis, J., Kratsch, D., Langer, A., Liedloff, M., Raible, D., & Rossmanith, P. (2011). An exact algorithm for the maximum leaf spanning tree problem. Theoretical Computer Science, 412(45), 6290–6302. https://doi.org/10.1016/j.tcs.2011.07.011.

[15] Haouari, M., Chaouachi, J., & Dror, M. (2005). Solving the generalized minimum spanning tree problem by a branch-and-bound algorithm. Journal of the Operational Research Society, 56(4), 382–389. https://doi.org/10.1057/palgrave.jors.2601821.

[16] Ekanayake, U., Daundasekara, W., & Perera, P. (2020). An approach for solving minimum spanning tree problem and transportation problem using modified ant colony algorithm. North American Academic Research, 3(9), 28–45. https://doi.org/10.5281/zenodo.4072472.

[17] Pop, C. P., Matei, O., Sabo, C., & Petrovan, A. (2018). A two-level solution approach for solving the generalized minimum spanning tree problem. European Journal of Operational Research, 265(2), 478–487. https://doi.org/10.1016/j.ejor.2017.08.015.

[18] Belaid, E., Limbourg, S., Mostert, M., Rigo, P., & Cools, M. (2016). Bi-objective road and pipe network design for crude oil transport in the Sfax region in Tunisia. Procedia Engineering, 142, 108–115. https://doi.org/10.1016/j.proeng.2016.02.020.

[19] Hassan, R. M. (2012). An efficient method to solve least-cost minimum spanning tree (LC-MST) problem. Journal of King Saud University – Computer and Information Sciences, 24, 101–105. https://doi.org/10.1016/j.jksuci.2011.12.001.

[20] Lichen, J., Cai, L., Li, J., Liu, S., Pan, P., & Wang, W. (2023). Delay-constrained minimum shortest path trees and related problems. Theoretical Computer Science, 941, 191–201. https://doi.org/10.1016/j.tcs.2022.11.014.

[21] Dhouib, S. (2024). Innovative method to solve the minimum spanning tree problem: The Dhouib-Matrix-MSTP (DM-MSTP). Results in Control and Optimization, 14, Article 100359. https://doi.org/10.1016/j.rico.2023.100359.

[22] Zhang, X., Wang, D., Fu, Z., Liu, S., Mao, W., Liu, G., Jiang, Y., & Li, S. (2020). Novel biogeography-based optimization algorithm with hybrid migration and global-best Gaussian mutation. Applied Mathematical Modelling, 86, 74–91. https://doi.org/10.1016/j.apm.2020.05.016.