drones

MDPI

Article

Design and Analysis of *k*-Connectivity Restoration Algorithms for Fault-Tolerant Drone Swarms in Harsh Civil Environments

Orhan Ceylan [1], Zuleyha Akusta Dagdeviren [2,*], Moharram Challenger [3,4] and Orhan Dagdeviren [2]

[1] International Computer Insitute, Ege University, 35040 Bornova, Turkey; orhanc78@gmail.com
[2] Department of Computer Engineering, Faculty of Computer and Information Sciences, Ege University, 35040 Bornova, Turkey; orhan.dagdeviren@ege.edu.tr
[3] Department of Computer Science, University of Antwerp, 2020 Antwerp, Belgium; moharram.challenger@uantwerpen.be
[4] AnSyMo/Cosys Core Laboratory, Flanders Make Strategic Research Center, 3001 Leuven, Belgium
* Correspondence: zuleyha.akusta.dagdeviren@ege.edu.tr

Highlights

What are the main findings?

- Four novel algorithms are proposed to restore *k*-connectivity in fault-tolerant drone swarms by minimizing drone movements under harsh environmental conditions.
- Extensive performance evaluations are provided under varying swarm sizes, network scales, *k* values, and transmission ranges.

What is the implication of the main finding?

- The proposed algorithms provide practical strategies for maintaining reliable communication and resilience in autonomous multi-drone operations exposed to node failures and environmental disruptions.
- The results provide a solid foundation for developing adaptive *k*-connectivity restoration strategies in large-scale drone networks.

Abstract

Drone swarms are increasingly used in critical civil applications like agriculture, machine maintenance and search-and-rescue, where maintaining network connectivity is essential for effective coordination. However, harsh environmental conditions can lead to drone failures, risking network fragmentation. To improve resilience, designing *k*-connected networks, where up to $k-1$ drone failures can be tolerated without losing connectivity, offers a practical solution by providing multiple independent communication paths between drones. The *k*-connectivity restoration problem is repositioning drones to achieve *k*-connectivity with minimal movement. In this study, we address this NP-Hard problem and propose novel solutions. Unlike existing *k*-connectivity restoration algorithms that constrain drones to predefined points, our model allows free repositioning within the mission area, increasing flexibility but also expanding the solution space and complexity. To address this problem, we propose three center-based algorithms that guide drones toward different central points computed from the network layout: in the first algorithm (ORIGIN), the center point is the geometric origin of the mission area; in the second algorithm (CENTROID), nodes move toward the centroid of all drone positions; and in the third algorithm, the center position is defined as the CENTer of the FARthest nodes (CENTFAR). We also introduce a Minimum Spanning Tree-based (MST) algorithm that moves drones along a minimum spanning tree to achieve and theoretically guarantee *k*-connectivity. Besides checking *k*-connectivity after each individual move, we also develop group-based variants where all drones move simultaneously and *k*-connectivity is checked

Check for updates

Academic Editors: Higinio González Jorge, Fernando Veiga López and Enrique Aldao Pensado

Received: 29 November 2025
Accepted: 23 December 2025
Published: 28 December 2025

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

afterward. We conduct comprehensive simulations under varying drone counts, network sizes, k values, and transmission ranges to evaluate the effectiveness and scalability of the proposed algorithms. CENTFAR provides the best movement efficiency among the center-based algorithms, slightly outperforming CENTROID and ORIGIN and achieving up to 21% lower total and 29% lower maximum movement than MST in smaller areas and higher k values. MST, however, performs best under low k and high transmission ranges, offering up to 57% lower total movement and 20% lower execution time than CENTFAR. Group-based variants accelerate convergence (up to a tenfold speedup) at the cost of a slight increase in movement. Our findings reveal that MST is ideal for low-k settings, while CENTFAR is better suited for high-connectivity deployments.

Keywords: drone swarms; harsh environments; connectivity restoration; drone-to-drone communication; k-connectivity; fault tolerance

1. Introduction

Nowadays, the use of drone swarms is becoming increasingly important for advanced applications that require scanning large areas and making intelligent decisions in critical civil domains such as agriculture, machine maintenance, healthcare and search-and-rescue missions [1–4]. For instance, in time-sensitive search-and-rescue scenarios, swarm drones can rapidly scan vast regions, access rugged and mountainous areas that are difficult for staff to reach, and perform intelligent tracking by combining images of traces captured from various parts of the terrain. To enable effective communication within a drone swarm, it is essential that the network remains connected. In such networks, communication paths must exist between every pair of drones. Since individual drones have limited coverage areas, they establish these communication paths by relaying information through neighboring drones. One of the major challenges in maintaining connectivity in drone swarms is the possibility of drone failures.

Several factors may lead to drones becoming inoperative. Environmental conditions such as precipitation, strong winds, and extreme temperatures can significantly affect drone performance. Additionally, the limited battery life of drones and the risk of battery depletion during a mission pose further challenges. Regardless of the cause, losing a drone from the network can severely impact its overall connectivity. For example, in a connected drone network, the failure of a critical drone can fragment the network into isolated components, thus disrupting its operation. An effective strategy to mitigate such risks is to increase the network's connectivity level. In a k-connected drone network, the system can tolerate up to $k-1$ drone failures without losing overall connectivity. Moreover, there exist k independent communication paths between every pair of drones, which enhances the network's resilience. As k increases, the network becomes more fault-tolerant.

The problem of repositioning drones in a non-k-connected network to achieve k-connectivity with minimal total movement is known as the k-connectivity restoration problem. This problem has been proven to be NP-Hard [5]. In this study, we address this problem and propose new approaches. The contributions of this paper are summarized below.

- Although k-connectivity restoration algorithms exist in the literature, most restrict drone movement to predefined points. While this constraint simplifies the problem, it also limits the flexibility of drone positioning. In contrast, our proposed model introduces a more flexible network approach, allowing drones to reposition freely

within the mission area. This modeling choice significantly increases the problem's complexity by expanding the solution space.

- To address this problem, we first introduce center-based algorithms. In the first algorithm (ORIGIN), drones are directed toward the geometric origin of the mission area; in the second algorithm (CENTROID), they move toward the centroid computed from all drone coordinates; and in the third algorithm (CENTFAR), the target point is chosen as the CENTer of the FARthest nodes in the network.
- We also propose a Minimum Spanning Tree-based k-connectivity restoration algorithm (MST). Unlike the center-based methods, this approach guides drones to move along the edges of a minimum spanning tree until k-connectivity is achieved. We theoretically prove that this algorithm guarantees k-connectivity.
- In the standard versions of ORIGIN, CENTROID, CENTFAR, and MST, k-connectivity is checked after each individual node movement. To explore alternative strategies, we also develop group-based variants, where all drones reposition simultaneously, and k-connectivity is verified only after this collective movement.
- We conduct a series of comprehensive simulations to assess the practical effectiveness and scalability of the proposed algorithms. The experiments consider various settings, including different drone counts, network area sizes, k values, and drone transmission ranges.
- Extensive simulations show that MST achieves the lowest total movement and execution time in scenarios with low k values and high transmission ranges though it results in higher maximum individual movement. CENTFAR performs better in higher k values, offering lower maximum movement. Group-based strategies reduce execution time but slightly increase movement compared to individual updates. Overall, MST suits low-k and high-range scenarios, while CENTFAR is preferable for when k value is high.

In our proposed center-based approaches, unlike previous mobility-based studies that restrict drones to a limited set of grid-defined positions, we restore k-connectivity by directing drones toward strategically defined geometric center points. This continuous and unconstrained movement allows the repositioning process to exploit the natural spatial distribution of the swarm, enabling faster convergence and more balanced topology formation compared to grid-based methods. By relying on geometric centers rather than discrete movement options, our model provides greater flexibility, reduces unnecessary detours, and is better aligned with the mobility characteristics of UAV systems. In contrast, the proposed MST-based algorithms provide a topology-driven repositioning approach that is conceptually different from geometric heuristics and has not been explored in prior k-connectivity restoration studies. By leveraging the structural properties of the minimum spanning tree, it systematically reduces inter-node distances in a hierarchical manner, offering a coordinated and connectivity-aware movement strategy that complements the geometric center–based methods. By combining these complementary strategies, our work offers a structured and comparative evaluation that highlights the novelty of our contributions.

The remainder of this paper is structured as follows: Section 2 provides a review of the related work in the literature, highlighting the main approaches and identifying the research gaps that motivate this study. Section 3 then presents the problem formulation, defining the system model and assumptions. Section 4 introduces the proposed algorithms explaining their design principles and underlying rationale. The performance of the proposed methods is thoroughly analyzed in Section 5. Finally, Section 6 concludes the paper by summarizing the main findings and outlining promising directions for future research.

https://doi.org/10.3390/drones10010016

2. Related Work

Connectivity restoration is a critical problem because the loss of network connectivity in unmanned aerial vehicles (UAV) swarms immediately disrupts coordination, information flow, and mission performance, posing serious risks in time-sensitive operations. In [6], a software-defined networking-based architecture is introduced that centrally manages connectivity restoration in partitioned flying ad hoc networks by collecting global network information through a control UAV. Besides, a greedy UAV scheduling algorithm is proposed that restores connectivity with significantly lower total and maximum movement distances than existing approaches. The study given in [7] introduces a deep reinforcement learning-based algorithm that guides multiple UAVs to restore a damaged collaborative surveillance network by simultaneously improving target coverage and data backhaul efficiency. Using a multihead attention mechanism and a stable learning strategy, the method achieves effective network recovery in both connected and disconnected scenarios. Li et al. proposed a distributed method to rapidly restore connectivity in UAV networks split into multiple clusters after large-scale failures [8]. By combining multi-agent deep deterministic policy gradient-based deep reinforcement learning with an artificial potential field model, their approach achieves faster reconnection with shorter total motion distances. For other reinforcement learning–based relaying solutions, please refer to [9]. Despite their significance, these studies generally aim to address the connectivity problem for the case of $k = 1$. In a 1-connected network, the failure of a single node may be sufficient to break connectivity, whereas in a k-connected network, connectivity is preserved even if up to $k - 1$ nodes fail. Accordingly, our proposed algorithms are specifically designed to meet the more demanding requirements of higher levels of fault tolerance.

Several studies in the literature have focused on restoring k-connectivity through link repair techniques, which generally involve re-routing strategies or adjusting transmission ranges. For instance, Gong et al. [10] proposed a method that utilizes multiple disjoint paths to reestablish connectivity after link failures. Similar strategies are employed in a range of studies [11–24]. Some approaches, such as [25], maintain connectivity by analyzing transmission range intersections, while others, like Zeng et al. [14], propose partial restoration strategies for specific k values. When increasing transmission power is impractical due to energy constraints, infrastructure-based methods involving the deployment of additional nodes are employed [25–28], albeit with increased operational costs.

On the other hand, mobility-based approaches attempt to relocate existing nodes to maintain k-connectivity. The k-Connectivity Maintenance in Heterogeneous networks (CMH) technique [16] leverages 2-hop local graphs but is unsuitable for mobile topologies and does not consider how a node's movement affects global connectivity. Centralized solutions like Minimum Cost k-Connectivity Restoration (MCCR) [12] relocate nodes within shortest path trees but assume static initial node placements. The study given in [29] introduces the fast k-connectivity restoration problem and proposes algorithms that efficiently reposition robots to rapidly regain k-connectivity with minimal movement. These studies are tailored for sensor networks and multi-robot systems; in contrast, our proposed approach in this paper introduces randomly placed drones in a mission area and allows flexible repositioning. This dynamic strategy offers adaptability and robustness, distinguishing it from earlier solutions that rely on fixed topologies or centralized control. Moreover, by providing unconstrained movement, our model can accommodate rapid environmental changes and unexpected disruptions more effectively. This capability allows the network to maintain operational reliability even under harsh or unpredictable mission conditions.

Zhu et al. [30] introduces the connectivity-maintenance UAV formation control algorithm, which addresses the challenge of preserving communication connectivity among multiple UAVs operating in environments with obstacles and interference sources. The

algorithm enables dynamic adaptation of the formation topology to maintain network connectivity and reconstruct disrupted communication links. Additionally, it incorporates a speed-based artificial potential field mechanism to support obstacle avoidance during cooperative formation maneuvers. Simulation-based evaluations demonstrate that the proposed algorithm can successfully establish, sustain, and restore UAV formations under complex operational conditions. This work highlights the importance of adaptive topology control in multi-UAV systems, aligning with the broader goal of robust connectivity maintenance. The most significant distinction between our approach and these studies lies in our explicit focus on achieving and maintaining k-connectivity. In a 1-connected network, the failure of a single critical node is sufficient to cause a complete disconnection. In contrast, a k-connected network (where $k \geq 2$) can withstand the failure of up to $k-1$ nodes without losing overall connectivity. By targeting k-connectivity, our work enhances the resilience of the drone network, ensuring greater fault tolerance and robustness against node failures. This higher level of structural redundancy is essential for real-world deployments in harsh or uncertain environments, where multiple drones may be affected simultaneously. As such, our approach directly addresses the need for reliable and mission-critical connectivity, which is not fully captured by methods limited to 1-connectivity.

Asci et al. [31] addressed the connectivity management problem by proposing four distinct solution approaches: an exact method formulated as an integer programming model, a heuristic that builds upon this integer programming formulation, a node converging heuristic, and a cluster moving heuristic. These approaches were thoroughly evaluated through comparative analyses conducted over a variety of network topologies and configurations. The findings of their study reveal that the heuristic grounded in integer programming is capable of producing solutions that closely approximate the optimal outcome. Meanwhile, the other heuristic methods, although potentially less accurate, exhibit significant computational efficiency, making them more suitable for time-sensitive scenarios. In their study, the authors consider a square geographical area that is partitioned into a grid-like structure, where drones are randomly deployed at the intersection points—referred to as junctions—of the grid lines. However, a key distinction between our approach and that of Asci et al. lies in the way the mobility and placement of drones within the network are modeled more comprehensively. In their formulation, the movement of drones is restricted to a finite and predefined set of grid junctions, which inherently limits the flexibility of drone positioning and network reconfiguration in practice. In contrast, our proposed model introduces a more general and flexible network structure, wherein drones are permitted to move freely and be repositioned to any arbitrary location within the defined mission area, not just to fixed grid points. This fundamental difference in the modeling paradigm significantly enhances the complexity of the problem we address and renders the algorithmic strategies developed by Asci et al. inapplicable or insufficient in our setting. To the best of our knowledge, the study we present here constitutes the first attempt at solving the k-connectivity restoration problem under the assumption that drones can move continuously and without positional constraints throughout the mission area. This unrestricted mobility paradigm introduces new challenges and necessitates the development of novel algorithmic techniques tailored to this more general and realistic operational environment. Moreover, it allows us to capture mobility dynamics that are far closer to real-world UAV operations, where environmental factors and mission constraints often require non-discrete repositioning. As a result, our model provides a more expressive and realistic framework for analyzing fault recovery scenarios and enables the exploration of algorithmic behaviors that cannot emerge under grid-based restrictions. At the end of this section, we present Table 1 to compare the existing studies in the literature and to clearly highlight our contribution.

Table 1. Comparison of Algorithms.

Study	Target Problem	Target Network	Node Positions
Liu et al. [6]	Connectivity Restoration	Flying Ad hoc Network	Not Predefined
Zhang et al. [7]	Connectivity Restoration	Unmanned Aerial Vehicle Network	Not Predefined
Liu et al. [8]	Connectivity Restoration	Unmanned Aerial Vehicle Network	Not Predefined
Gong et al. [10]	Connectivity Restoration	Industrial Wireless Network	Predefined
Senturk et al. [11]	Connectivity Restoration	Mobile Sensor Network	Predefined
Wang et al. [12]	*k*-Connectivity Restoration	Wireless Sensor and Actor Network	Predefined
Uzun et al. [13]	Connectivity Restoration	Underwater Acoustic Sensor Network	Not Predefined
Zeng et al. [14]	2-Connectivity Restoration	Wireless Sensor Network	Predefined
Zhang et al. [15]	Connectivity Restoration	Wireless Sensor and Actor Network	Predefined
Akram et al. [16]	*k*-Connectivity Restoration	Wireless Sensor and Actor Network	Predefined
Yang et al. [17]	*k*-Connectivity Restoration	Mobile Sensor Network	Predefined
Akkaya et al. [18]	Connectivity Restoration	Wireless Sensor and Actor Network	Predefined
Liu et al. [19]	Connectivity Restoration	Wireless Sensor Network	Predefined
Dimarogones and Johansson [20]	Connectivity Restoration	Mobile Network	Predefined
Lu et al. [23]	Connectivity Restoration	Maritime Wireless Sensor Network	Predefined
Zhao et al. [24]	Connectivity Restoration	Underwater Wireless Sensor Network	Not Predefined
Almasaeid and Kamal [25]	*k*-Connectivity Restoration	Wireless Sensor Network	Predefined
Han et al. [26]	*k*-Connectivity Restoration	Wireless Sensor Network	Predefined
Lee et al. [27]	*k*-Connectivity Restoration	Wireless Sensor Network	Predefined
Liu et al. [28]	Connectivity Restoration	Wireless Sensor Network	Predefined
Shi et al. [29]	*k*-Connectivity Restoration	Multi-Robot System	Not Predefined
Zhu et al. [30]	Connectivity Restoration	Unmanned Aerial Vehicle Network	Not Predefined
Asci et al. [31]	*k*-Connectivity Restoration	Drone Network	Predefined
This Paper	*k*-Connectivity Restoration	Drone Network	Not Predefined

3. Problem Formulation

In our model, all drones are assumed to be equipped with uniform hardware components and run identical software architectures, ensuring consistent operational behavior across the fleet and avoiding discrepancies that may arise in heterogeneous systems. Two drones are considered directly connected if the Euclidean distance between them does not exceed the maximum transmission range (T_r), within which reliable bidirectional communication (e.g., IEEE 802.11 [32]) is feasible. Any failure—whether due to hardware malfunction, energy depletion, or loss of control—immediately disrupts the communication links associated with the affected node, potentially degrading overall network connectivity. To manage such disruptions, the execution of the proposed algorithms is delegated to an external ground station rather than the drones themselves. Drones periodically transmit their state information, including position and operational status, to nearby cellular base stations using 5G and beyond (5G/B5G) communication technologies. These base stations forward the telemetry data to the ground station, enabling it to reconstruct the swarm topology at regular intervals. When the ground station detects that the network has lost *k*-connectivity, it computes the necessary repositioning commands and sends individualized movement instructions back to the drones via the same cellular infrastructure. This centralized architecture removes the need for drones to maintain continuous global knowledge of the network, and because communication occurs through terrestrial cellular infrastructure rather than

direct drone-to-drone broadcasting, inconsistencies in the reconstructed topology do not arise during dissemination of control commands.

At the beginning of the problem, we assume that the drones are randomly positioned in the mission area without any prior coordination. Under such a random deployment, it is entirely possible that no communication links exist between certain drones or even that the network is completely disconnected resulting in an initial configuration that is strictly non-k-connected. This assumption is intentional, as our algorithms are designed to operate from arbitrary or highly fragmented initial topologies and restore the required level of k-connectivity.

We model the drone network $G\,(V, E, P)$ as an undirected graph where $V = \{v_1, v_2, \ldots\}$ is the set of drones, $E = \{e_1, e_2, \ldots\}$ is the set of communication links and $P = \{p_1, p_2, \ldots\}$ is the set of drone positions. Each $p_i \in P$ is defined as coordinate pair (X_i, Y_i). We also define $\kappa(G(V, E, P))$ as the k-connectivity value of graph G. As previously stated, $(v_i,v_j) \in E$, if the Euclidean distance between p_i and p_j ($\|p_i - p_j\|_2$) are at most T_r. We define $f : (X, Y) \times (X, Y) \rightarrow \mathbb{R}^+$ as the movement cost function that is the Euclidean distance to move from the initial position (X_i, Y_i) to the final position (X_{new}, Y_{new}). The aim of k-connectivity restoration problem is to minimize $\sum_{p_i \in P} f(P, P_{new})$ such that $G(V, E_{new}, P_{new})$ is k-connected. Formally, the problem is to find a P_{new} set such that

$$minimize \sum_{p_i \in P} f(P, P_{new})$$

subject to the condition that

$$\kappa(G(V, E_{new}, P_{new})) = k$$

where

$$E_{new} = \{e_{ij} : v_i, v_j \in V \; and \; ||p_i - p_j||_2 \leq T_r \; and \; p_i, p_j \in P_{new}\}.$$

Environmental disturbances such as wind gusts, turbulence, and precipitation can induce unintended positional deviations in drones or, in more severe cases, cause partial node failures. These perturbations alter the spatial configuration of the UAV swarm in a non-deterministic manner, potentially breaking one or more of the internally node-disjoint paths required to maintain k-connectivity. In operational terms, even small lateral drifts can increase inter-drone distances beyond the communication threshold, thereby reducing the network's vertex connectivity and fragmenting the communication graph. Within this context, the problem we address explicitly models the consequences of such harsh environmental conditions rather than simulating the physical disturbances themselves. Our proposed algorithms are designed to restore the k-connectivity of the network once environmental factors have degraded it, by computing new positions for the drones and repositioning them to re-establish the required level of redundancy. Formally, when external forces perturb the position set P to a new configuration P', resulting in a connectivity value $\kappa(G(P')) < k$, the goal is to compute a corrected configuration $\widehat{P}$ such that $\kappa(G(\widehat{P})) \geq k$. Because environmental disruptions may persist or occur intermittently throughout the mission, the restoration mechanism must be viewed as a periodic or on-demand control process rather than a one-time correction. Accordingly, the algorithms can be executed repeatedly whenever the system detects a drop in connectivity, ensuring that the network remains robust against continuous or recurrent disturbances in harsh civil environments.

An illustrative example of a drone network consisting of 10 drones is presented in Figure 1. The drones are labeled alphabetically, forming the node set $V = \{A, B, \ldots, J\}$. Each drone has a maximum transmission range (T_r) of 250 m. The communication links

between drones are represented by solid lines, corresponding to the edge set E, where the weight of each edge indicates the Euclidean distance between the connected drones. Initially, the network is not connected and consists of two separate components. The objective is to achieve a 2-connected topology, ensuring that the network remains connected even if any single node fails. To reach this configuration with minimal repositioning effort, drone F is moved 10 m closer to drone E, and drone I is repositioned 20 m toward drone A. As a result of these adjustments, two new links, (E,F) and (A,I), are established. These additional connections effectively merge the previously disconnected components, restoring full network connectivity and resulting in a robust 2-connected topology that can tolerate single-node failures.

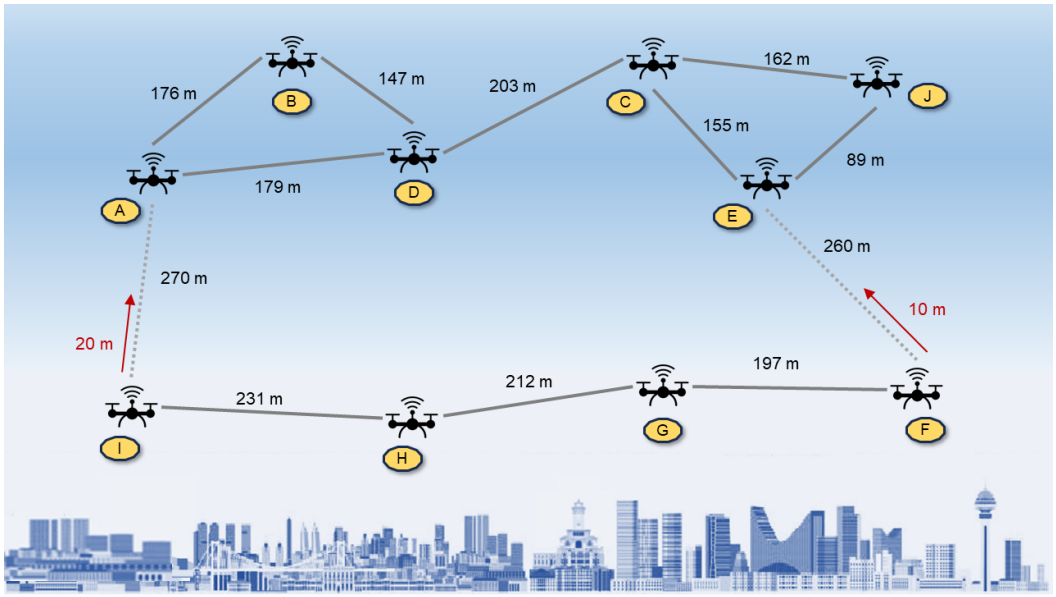


Figure 1. Drone Network Example (Drone identifiers are assigned from A to J).

4. Proposed Algorithms

In this paper, we propose ORIGIN, CENTROID, CENTFAR and MST algorithms which generate deterministic movement patterns to address the problem we are working on. We generalize center-based algorithms (which are ORIGIN, CENTROID and CENTFAR) as CENTER algorithm that is given in Algorithm 1. We denote the drone positions by $P = \{p_1, \ldots, p_n\}$ with $p_i \in \mathbb{R}^2$, the transmission range by T_r, and the target connectivity level by k. The algorithm type is selected from $\{\text{ORIGIN}, \text{CENTROID}, \text{CENTFAR}\}$. In Line 1, the algorithm constructs the communication graph $G = (V, E)$, where $V = \{V_1, \ldots, V_n\}$ and $e_{ij} \in E$ if and only if $\|p_i - p_j\|_2 \leq T_r$. In Line 2, the center point c is computed as follows: for ORIGIN, $c = (0, 0)$; for CENTROID, $c = \frac{1}{n}\sum_{i=1}^{n} p_i$; and for CENTFAR, if $(i^*, j^*) = \arg\max_{i,j} \|p_i - p_j\|_2$, then $c = \frac{p_{i^*} + p_{j^*}}{2}$. The algorithm iterates while $findConnectivity(G) < k$, where connectivity is determined through the vertex-connectivity measure $\kappa(G)$.

In Line 4, each drone moves toward c according to the distance $d_i = \|p_i - c\|_2$. The new position is $p_i' = p_i + \alpha_i \frac{c - p_i}{\|c - p_i\|_2}$, where the step size is

$$\alpha_i = \begin{cases} T_r, & d_i > 1.5T_r, \\ d_i - 0.5T_r, & 0.5T_r < d_i \leq 1.5T_r, \\ 0, & d_i \leq 0.5T_r. \end{cases}$$

After updating all positions, the graph is reconstructed as $G \leftarrow constructGraph(P, T_r)$ (Line 5). The algorithm terminates when $findConnectivity(G) \geq k$ and returns the final position set P.

Algorithm 1 CENTER: The Generalized Algorithm for ORIGIN, CENTROID and CENTFAR Restoration Algorithms

Inputs.
  P: the set of positions
  k: the target k-connectivity value
  T_r: the transmission range
  $type$: the center type

1. $G(V,E) \leftarrow$ constructGraph(P,T_r)
2. $centerPosition \leftarrow$ findCenter(P,$type$)
3. **while** findConnectivity(G) < k
4. $P \leftarrow$ move(P,$centerPosition$)
5. $G(V,E) \leftarrow$ constructGraph(P,T_r)
6. **end while**
7. **return** P

Figure 2 presents an example drone network deployment, where the central positions determined by the ORIGIN, CENTROID, CENTFAR algorithms are represented by green, pink, and orange location markers, respectively. Each algorithm follows a different strategy for defining the center of movement, thereby influencing the final network configuration. ORIGIN uses a predefined coordinate as the target position; (0, 0) in the above scenario. In our tests, initial drone coordinates were randomly generated around the predefined target. In CENTROID, target coordinate is computed as the arithmetic mean of all drone positions. In the example, this value is (239, 216). For CENTFAR, the target coordinate is determined as the midpoint of the farthest X and farthest Y coordinates. In the figure, the farthest X values are 494 (Drone B) and –396 (Drone F), while the farthest Y values are 490 (Drone E) and –234 (Drone F). The resulting midpoints, thus the target coordinate for CENTFAR is (49, 128).

The step-by-step behavior of the ORIGIN algorithm within a network of five drones is illustrated in Figure 3 where T_r = 150 m. After an initial k-connectivity evaluation, the first drone in the node list, Drone A, moves from its initial position (383, 285) to (263, 195), covering a full transmission range since it was more than $1.5 \times T_r = 225$ away from the target. After this move, k-connectivity is checked but since the desired level has not been reached, the algorithm continues with the next drone. Drone B, initially at $(-15, 335)$, is also beyond the $1.5 \times T_r$ boundary and moves one T_r to $(-12, 185)$. Since the k-connectivity is still below the desired level, the algorithm continues with the next drone in the list, Drone C. The above rules are applied to Drone C and to other drones in the list in order. In the first round, all drones move exactly one T_r since all are initially located farther than $1.5 \times T_r$ from the target. In the second round, Drone A repeats a full T_r displacement while the others move just enough to place themselves $0.5 \times T_r$ from the target. After the second round, Drone A is still disconnected from the rest of the network, leaving the graph's k-connectivity at 0. In the third round, Drone A makes a shorter move that brings it to exactly $0.5 \times T_r$ from the target. Following this operation, k-connectivity is checked and it is observed that the desired k-connectivity level is achieved; the algorithm terminates. In Figure 3, the final topology forms a complete graph, achieving the maximum possible k-connectivity. Notably, even a $k = 1$ level could not be achieved earlier; it is only realized when all drones reach their closest permissible distance ($0.5 \times T_r$) from the target.



Figure 2. Exampleof Central Positions in Algorithms (Green Arrow:ORIGIN, Pink Arrow:CENTROID, Orange Arrow:CENTFAR.

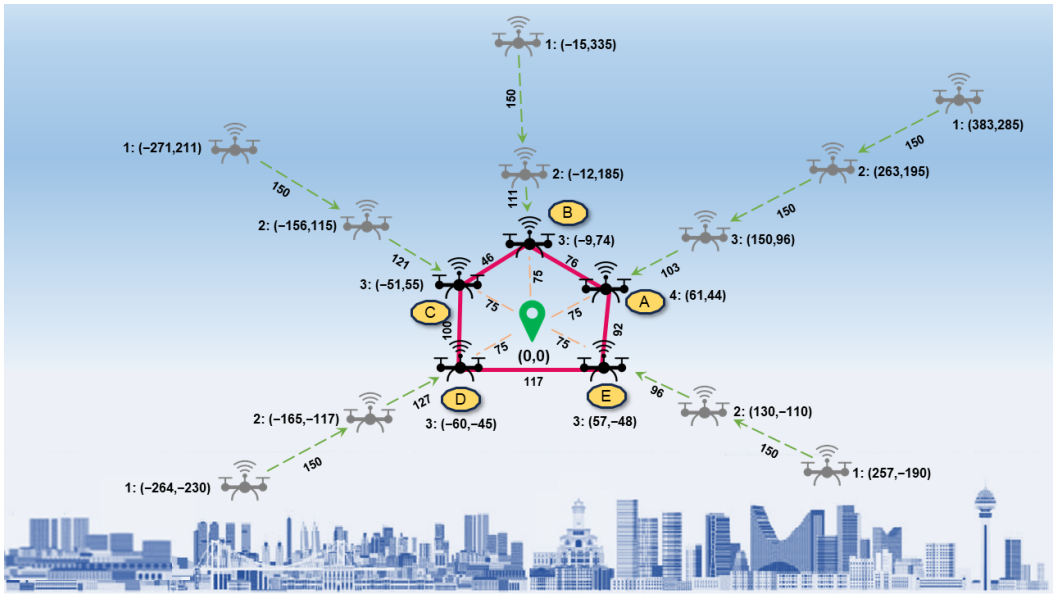


Figure 3. ORIGIN Example.

Another algorithm proposed in this study is the MST algorithm, whose steps are given in Algorithm 2. The input consists of the position set $P = \{p_1, \ldots, p_n\}$, the transmission range T_r, and the target k-connectivity requirement. In Line 1, the communication graph $G = (V, E)$ is generated using $constructGraph(P, T_r)$, where $e_{ij} \in E$ if $\|p_i - p_j\|_2 \leq T_r$. We will use the graph G in Line 5 to evaluate its k-connectivity. In Line 2, the algorithm constructs a weighted complete graph $C = (V, E_C)$ via $constructWeightedCompleteGraph(P)$, where all node pairs are connected and each edge weight is the Euclidean distance, i.e.,

$$w(i, j) = \|p_i - p_j\|_2, \qquad \forall (i, j) \in E_C.$$

Algorithm 2 MST: Minimum Spanning Tree-based Restoration Algorithm

Inputs.
  P: the set of positions
  k: the target k-connectivity value
  T_r: the transmission range

1. $G(V,E) \leftarrow$ constructGraph(P,T_r)
2. $C(V,E,w) \leftarrow$ constructWeightedCompleteGraph(P)
3. $M(V,E,w) \leftarrow$ findMST(C)
4. $centerNode \leftarrow$ findMSTCenter(M)
5. **while** findConnectivity(G) < k
6. $P \leftarrow$ moveMST(P,$centerPosition$)
7. $G(V,E) \leftarrow$ constructGraph(P,T_r)
8. **end while**
9. **return** P

In Line 3, the minimum spanning tree (MST) M of the complete graph C is computed. The *center node* of M is then obtained using $findMSTCenter(M)$, which selects the node minimizing the maximum distance to all other nodes in the MST (i.e., the tree's eccentricity center). Denoting this central node by m, the algorithm proceeds by traversing M in a depth-first manner. For each node i visited during the traversal, MST repositioning is performed using $moveMST$. If $parent(i) = m$, node i is moved toward m until its distance satisfies $\|p_i - p_m\|_2 = T_r/k$. Subsequently, for each child j of node i, the movement rule ensures

$$\|p_j - p_i\|_2 = T_r/k,$$

by moving j toward i along the direction of $(p_i - p_j)$. This hierarchical procedure is applied recursively to all nodes in the MST.

When applied to the full tree, this repositioning ensures that the final topology contains at least k internally node-disjoint paths between any pair of nodes, thereby achieving k-connectivity and providing resilience against up to $k-1$ node or link failures. We further provide a formal proof in Theorem 1, which rigorously demonstrates how the proposed method guarantees k-connectivity under the defined network model and assumptions.

Theorem 1. *MST connectivity restoration algorithm produces k-connected drone network.*

Proof. We use induction to prove this theorem. We start with the basis step (k = 1), proceed with k = 2, and finalize with the inductive step as following.

- Basis step: For k = 1, we assume that the initial network graph is a linear topolog, y as given in Figure 4, since the distance between nodes are farthest in this situation. When node y is attached to the network, k = 1 property holds.
- k = 2: Similar with the previous case, we assume the nodes are arranged as a ring topology, as shown in Figure 5. We connect node y to the initial ring network through node x where the distance between them is $T_r/2$. From the triangle inequality, l_1 and l_2 should be less than or equal to T_r since the distance between node x and n_i is at most $T_r/2$ where $i \in \{1,2\}$. Therefore, node y should be neighbors with both n_1 and n_2. Consequently, the new graph is 2-connected.

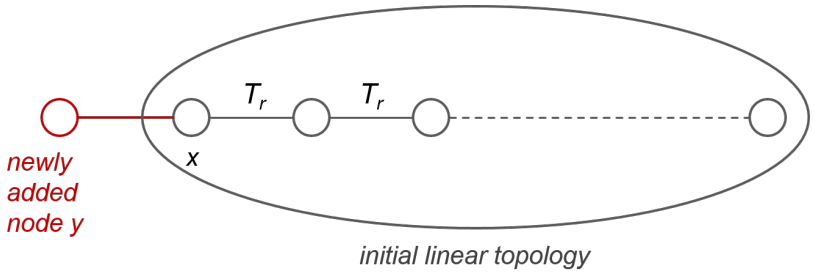


Figure 4. Basis Step (Linear Topology).

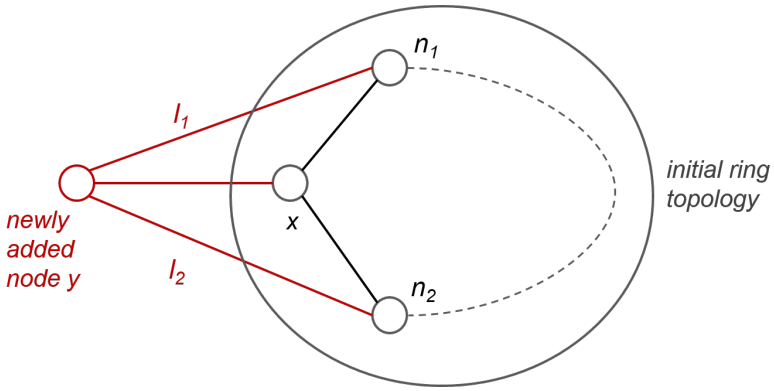


Figure 5. $k = 2$ step.

- Inductive step: We assume that the initial network is k-connected, as given in Figure 6. We attach node y to node x which produces triangles one of which is given in Figure 7. From triangle inequality $l_i \leq 2T_r/k$ and for $k \geq 2$, y should be neighbors with n_i. In a k-connected graph, each node pair should have at least k disjoint paths. So we investigate the disjoint paths from node y to the other nodes. The set of disjoint paths from node y to node x is $S_{yx} = \{(y, n_1, x), (y, n_2, x), \ldots, (y, n_k, k)\}$ as we found that y should be neighbors with n_i. The disjoint path from node x to the other nodes such as node v is $S_{xv} = \{(x, n_1, \ldots, v), (x, n_2, \ldots, v), \ldots, (x, n_k, \ldots, v)\}$ since the initial graph is k-connected. To construct S_{yv} we replace node y instead of node x in S_{xv} and obtain $S_{yv} = \{(y, n_1, \ldots, v), (y, n_2, \ldots, v), \ldots, (y, n_k, \ldots, v)\}$. We have k disjoint paths from node y to all other nodes. This proves the inductive step.

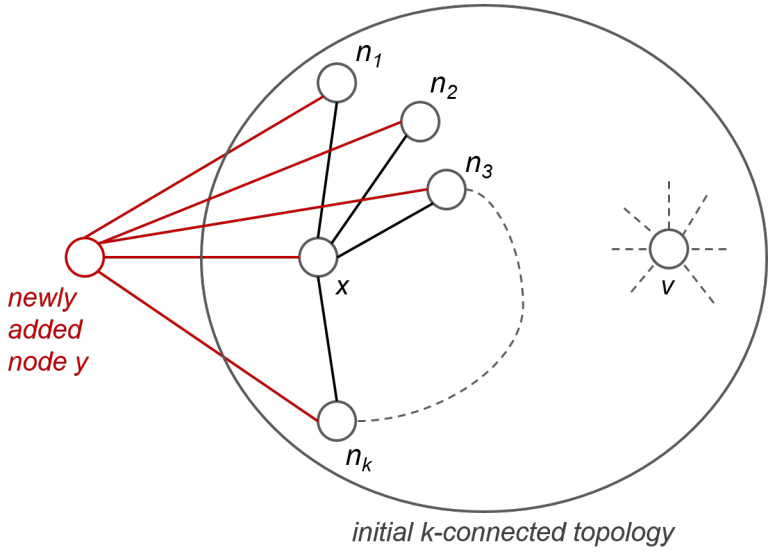


Figure 6. Inductive Step.

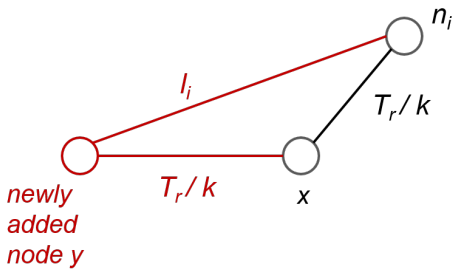


Figure 7. Triangle Graph in Inductive Step.
□

Figure 8 demonstrates a sample application of the MST algorithm. In this scenario, node A is chosen as the central reference point that anchors the network's reconstruction process. Following the edges defined by the minimum spanning tree, drones B, C, H, and I gradually move toward their MST tree parents, ensuring that their final positions create an uninterrupted communication structure, a $k-1$ connected network. The algorithm begins with an initial k-connectivity assessment. If the required level is already met, the algorithm terminates immediately (similar to other target-based algorithms). The first drone in the neighbor list of the central drone, Drone B, moves toward the central drone until the distance between them equals the required threshold T_r/k. After a connectivity check, Drone C, the child of Drone B, moves toward Drone B's updated position until it is within T_r/k as well. Since Drone C has no additional neighbors, algorithm returns to Drone B, and since Drone B has no other neighbors unvisited, it returns back to the central drone, Drone A. Next unvisited drone in the neighbor list of Drone A is Drone H. Drone H is outside the required T_r/k distance and thus moves accordingly. After a connectivity check, its minimum spanning tree child, Drone I, moves to maintain the same T_r/k spacing relative to Drone H's updated location. Since Drone I, and subsequently Drone H has no other unvisited neighbors, algorithm returns to central node again. The central node has still an unvisited neighbor, Drone D, but since it is within the required T_r/k distance, it does not move. The same controls are applied over its neighboring drones and over their neighbors. All those drones do not move since they are already within the requested distance of their parents. When it is returned to the central drone and all nodes have been visited, MST algorithm ends, having produced the required k-connectivity level.

The original proposed approaches perform a k-connectivity evaluation after each individual node movement. Formally, let $p_i^{(t)}$ denote the position of node i at iteration t. The sequential version updates a single node at a time using

$$p_i^{(t+1)} = p_i^{(t)} + \alpha_i^{(t)} \frac{c^{(t)} - p_i^{(t)}}{\|c^{(t)} - p_i^{(t)}\|_2},$$

followed immediately by a connectivity check on the updated graph $G^{(t+1)}$. In contrast, we also evaluate a deterministic simultaneous-movement variant in which all nodes are relocated in parallel during each iteration. In this formulation, the update rule becomes

$$P^{(t+1)} = \left\{ p_i^{(t)} + \alpha_i^{(t)} \frac{c^{(t)} - p_i^{(t)}}{\|c^{(t)} - p_i^{(t)}\|_2} \;\middle|\; i = 1, \ldots, n \right\},$$

meaning that node movements are applied as a group rather than individually. After computing the full updated set $P^{(t+1)}$, the k-connectivity condition is verified once for the entire set:

$$findConnectivity\big(constructGraph(P^{(t+1)}, T_r)\big) \geq k.$$

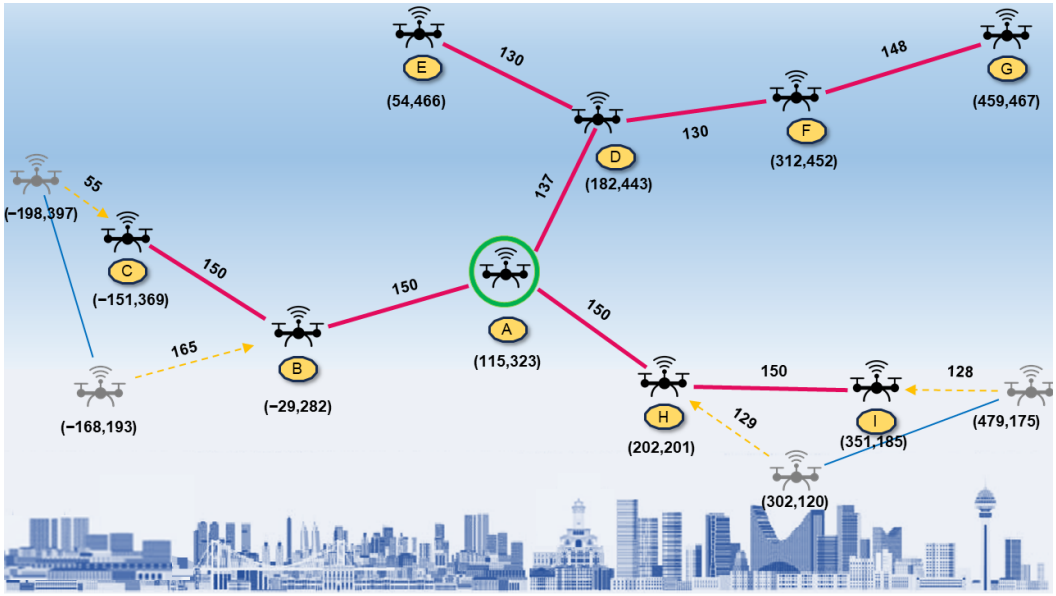


Figure 8. MST Example.

To implement this variation, we modified the internal movement function of our proposed algorithms so that node updates are computed for all $i \in V$ before any connectivity evaluation occurs. This results in a collective movement step at each iteration and allows us to analyze the effect of group-wise repositioning on convergence behavior and connectivity restoration. To distinguish these modified versions, we appended the "-G" (Group) suffix to the original algorithm names.

The time complexities of the algorithms and their group-movement variants are provided in Theorems 2, 3, and 4. In the complexity analysis, the diagonal of the rectangular deployment area is denoted by D and the number of edges in the graph is denoted by m. The proposed algorithms have asymptotically equal time complexities in the best case. In the worst case, the group-based variants achieve better time complexity than their sequential counterparts. Furthermore, the MST-based algorithms exhibit asymptotically better worst-case time complexity compared to the other approaches. A comparative summary of the complexity analysis is presented in Table 2.

Theorem 2. *The time complexities of CENTER-based algorithms and their group-based versions are* $O(\frac{min\{k^3+n,kn\}mnD}{T_r})$ *and* $O(\frac{min\{k^3+n,kn\}mD}{T_r})$*, respectively.*

Proof. The time complexity of the *constructGraph* function in Line 1 is $O(n^2)$. In Line 2, the $findCenter$ function determines the center position depending on whether the algorithm type is ORIGIN, CENTROID, or CENTFAR. Its running time is $O(1)$ for ORIGIN, $O(n)$ for CENTROID, and $O(n^2)$ for CENTFAR. The complexity of the $findConnectivity$ function in Line 3 is $O(\min\{k^3+n, kn\})$ [33]. In Line 4, the *move* function runs in $O(n)$ time for the group-based variants and $O(1)$ time for the sequential (single-movement) versions. As stated earlier, the complexity of Line 5 is again $O(n^2)$.

The algorithm iterates until k-connectivity is achieved. In the worst case, a node positioned on the diagonal of the deployment area must travel from one corner to the opposite corner, and the movement distance per iteration is bounded by T_r (excluding last move). Therefore, the number of iterations of the *while* loop is $O(nD/T_r)$ for the sequential algorithms and $O(D/T_r)$ for the group-based algorithms, where D denotes the diagonal length of the deployment rectangle.

Combining these bounds, the overall time complexities of the CENTER-based algorithms and their group-based versions are $O(\frac{min\{k^3+n,kn\}mnD}{T_r})$ and $O(\frac{min\{k^3+n,kn\}mD}{T_r})$, respectively. □

Theorem 3. *The time complexities of MST-based algorithm and its group-based version are* $O(min\{k^3+n,kn\}mn)$ *and* $O(min\{k^3+n,kn\}m)$*, respectively.*

Proof. As stated in the previous proof, the complexities of the graph construction functions used in Lines 1 and 2 are both $O(n^2)$. In Line 3, the $findMST$ function computes the minimum spanning tree in $O((n+m)\log n)$ time. The $findMSTCenter$ function in Line 4 runs in $O(n^2)$ time. The k-connectivity check in Line 5 requires $O(\min\{k^3+n,kn\})$ time. In Line 6, the *moveMST* function executes in $O(1)$ time for the sequential version, but in $O(n)$ time for the group-based variant.

For the sequential algorithm, the *while* loop may iterate up to $O(n)$ times, whereas for the group-based version the loop executes only once. By combining these bounds, the total time complexities of the MST-based algorithm and its group-based variant are $O(\min\{k^3+n,kn\}mn)$ and $O(\min\{k^3+n,kn\}m)$, respectively. □

Theorem 4. *The lower bound time complexities of all proposed algorithms is* $\Omega(min\{k^3+n,kn\}m)$.

Proof. In the best-case scenario, the *while* loop in all algorithms executes only once. Under this condition, the overall message complexity reduces to $O(min\{k^3+n,\ kn\}m)$. □

Table 2. Time Complexity Comparison of Proposed Algorithms.

Algorithm	Variant	Complexity Bounds	
CENTER-based Algorithms	Individual	Lower: $\Omega(min\{k^3+n,kn\}m)$	Upper: $O(\frac{min\{k^3+n,kn\}mnD}{T_r})$
	Group	Lower: $\Omega(min\{k^3+n,kn\}m)$	Upper: $O(\frac{min\{k^3+n,kn\}mD}{T_r})$
MST-based Algorithms	Individual	Lower: $\Omega(min\{k^3+n,kn\}m)$	Upper: $O(min\{k^3+n,kn\}mn)$
	Group	Tight Bound: $\Theta(min\{k^3+n,kn\}m)$	

5. Experimental Evaluation

To examine the practical applicability and scalability of the proposed algorithms, a series of comprehensive simulations was conducted under various experimental settings. Proposed ORIGIN, CENTFAR, CENTROID and MST algorithms with their group versions are implemented in the experimental process. To compare the performance of our proposed algorithms, we implemented the CENTER-based RANdom-TARget (RANTAR) algorithm. As described, this algorithm begins by selecting a random point in the mission area as the target center position. Similar to the other CENTER algorithms, each node moves toward this point until the network achieves k-connectivity. In the group version, RANTAR-G, all nodes perform one movement step simultaneously, after which the k-connectivity condition is evaluated.

Throughout the evaluation process, three primary metrics were used to quantify performance. First, the Total Movement Distance, which accounts for the sum of distances traveled by all drones to achieve the required connectivity. Second, the Maximum Individual Movement, indicating the largest single movement made by any drone during the process. And third, the WallClock Execution Time, which measures the time taken by the algorithm to compute a valid solution. The experiments aimed to test algorithm performance across different drone counts, area sizes, k-connectivity levels and transmission ranges. Square regions of five different dimensions (1000 m, 1414 m, 1732 m, 2000 m, and

2236 m per side) were used to represent varying spatial constraints. In this way, areas of 1 km^2, 2 km^2, 3 km^2, 4 km^2 and 5 km^2 were used in the simulations. Within each area, drones were randomly positioned to form initially networks with 10, 20, 30, 40 or 50 nodes. In all of our experiments, the initial values of k were equal to 0. While testing the algorithms, k-connectivity values were selected among 1, 2, 3, 4 and 5. The coverage areas were determined as 50 m, 100 m, 150 m, 200 m and 250 m, starting from 50 m and increasing by 50 m.Each measurement was obtained as the average of 30 independent simulation runs. The standard deviations of the measurements were visualized using error bars in the figures. We implemented the algorithms on Google Colab platform by using the Python 3.12.12 programming language. The NetworkX library was used to construct graph representations of drone networks and manage their connectivity dynamically during simulations. The simulation parameters are summarized in Table 3.

Table 3. Simulation Parameters.

Parameter	Values
Implemented Algorithms	ORIGIN, CENTROID, CENTFAR, MST, RANTAR, ORIGIN-G, CENTROID-G, CENTFAR-G, MST-G, RANTAR-G
Drone Counts	10, 20, 30, 40, 50
Network Areas	1000 m × 1000 m, 1414 m × 1414 m, 1732 m × 1732 m, 2000 m × 2000m, 2236 m × 2236 m
Communication Ranges	50 m, 100 m, 150 m, 200 m, 250 m
k values	1, 2, 3, 4, 5

Figures 9 and 10 illustrate the total movement generated by the algorithms under varying node counts, network sizes, k-connectivity requirements, and transmission ranges. As the number of drones increases, the total movement required by all algorithms exhibits linear growth. Notably, the MST algorithm performs increasingly well with higher node counts. CENTER-based algorithms produce similar results. Similarly, as the network area expands, the total movement generated by all algorithms increases almost linearly. This is expected as maintaining connectivity in larger areas requires more repositioning. The MST algorithm shows improved performance in smaller network areas due to naturally higher connectivity. With increasing k values, the movement required by the MST algorithm rises significantly in order to maintain k-connectivity. In contrast, the movement produced by the CENTER-based algorithms increase only slightly as k grows. MST performs especially well when the k requirement is low. As the transmission range of nodes increases, the total movement decreases across all algorithms, since broader ranges make it easier for nodes to establish connections. Under larger transmission conditions, MST shows superior results. However, when the transmission ranges are limited, ORIGIN, CENTFAR, and CENTROID become more effective. RANTAR consistently shows the weakest performance in every evaluation. Finally, it is observed that algorithms that move nodes individually tend to produce slightly lower total movement compared to those that move all nodes as a group.

Figures 11 and 12 present the maximum movement incurred by the algorithms under varying node counts, network sizes, k values, and transmission ranges. As the number of drones increases, the maximum movement produced by the MST algorithm decreases slightly, while the ORIGIN, CENTROID and CENTFAR algorithms show a modest increase in maximum movement. Across all node counts, MST consistently yields the highest maximum movement values, indicating weaker performance in this metric. Similarly, as

the network area expands, the maximum movement required by all algorithms increases almost linearly. Yet again, MST produces the worst results across all area sizes, whereas CENTFAR slightly outperforms the other center-based approaches. When k = 1, MST achieves the best performance among all algorithms. However, as the k value increases, the maximum movement produced by MST rises significantly. In contrast, center-based algorithms maintain relatively stable performance across different k values. An increase in transmission range leads to a reduction in maximum movement across all algorithms, as greater connectivity is more easily achieved. Under varying transmission conditions, CENTFAR demonstrates the best performance, while MST continues to exhibit the highest maximum movement. In all evaluations, RANTAR delivered the poorest maximum-movement performance. Lastly, algorithms that move nodes individually result in slightly lower maximum movement compared to those that move nodes in groups.

Figures 13 and 14 illustrate the wallclock time of the algorithms under varying node counts, network sizes, k values, and transmission ranges. The standard deviation of the RANTAR-G algorithm was relatively high, and including it in the figures significantly reduced readability; therefore, it has been omitted from the Figures 13 and 14. As the node count increases, the time spent by all algorithms increases approximately linearly, indicating that the proposed methods are scalable. Among the algorithms that move nodes individually, MST consistently shows the shortest execution time, making it the most time-efficient in this category. This result aligns well with the complexity analysis. In contrast, the MST-G algorithm, which moves nodes in groups, demonstrates significantly worse performance as the number of drones increases. Execution time also increases with higher k values across all algorithms. In scenarios with varying k, MST remains the fastest among individual-movement algorithms, while MST-G again exhibits the poorest performance among group-movement algorithms. Overall, the ORIGIN, CENTROID, and CENTFAR algorithms show similar execution times. RANTAR performs worst in terms of execution time when compared with the other algorithms. As expected from the complexity analysis, the algorithms that move nodes in groups tend to require less time than those that move nodes individually, due to fewer iterations.

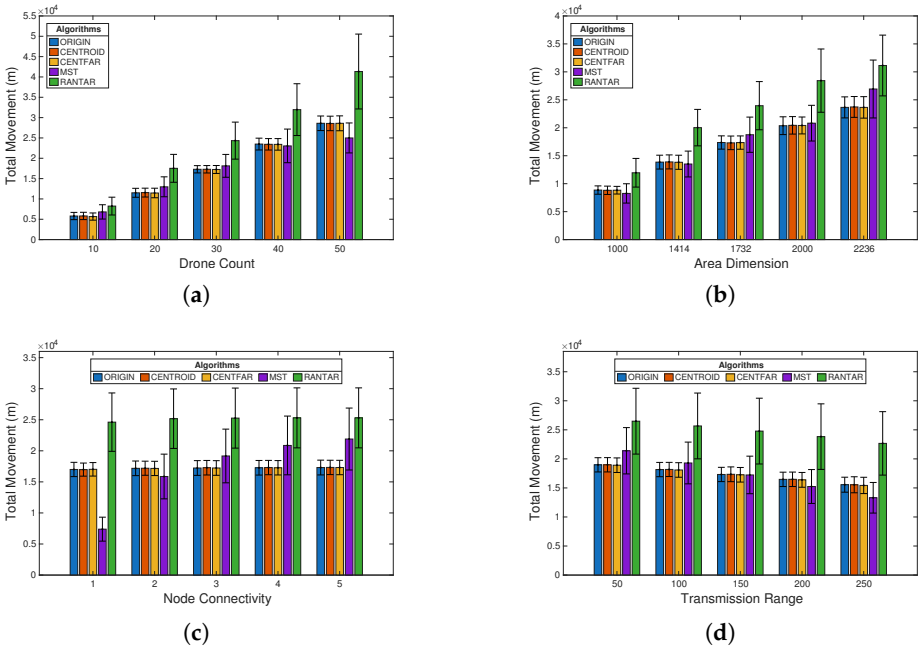


Figure 9. Total movement of algorithms (**a**) against drone count, (**b**) against network area, (**c**) against k, (**d**) against T_r.

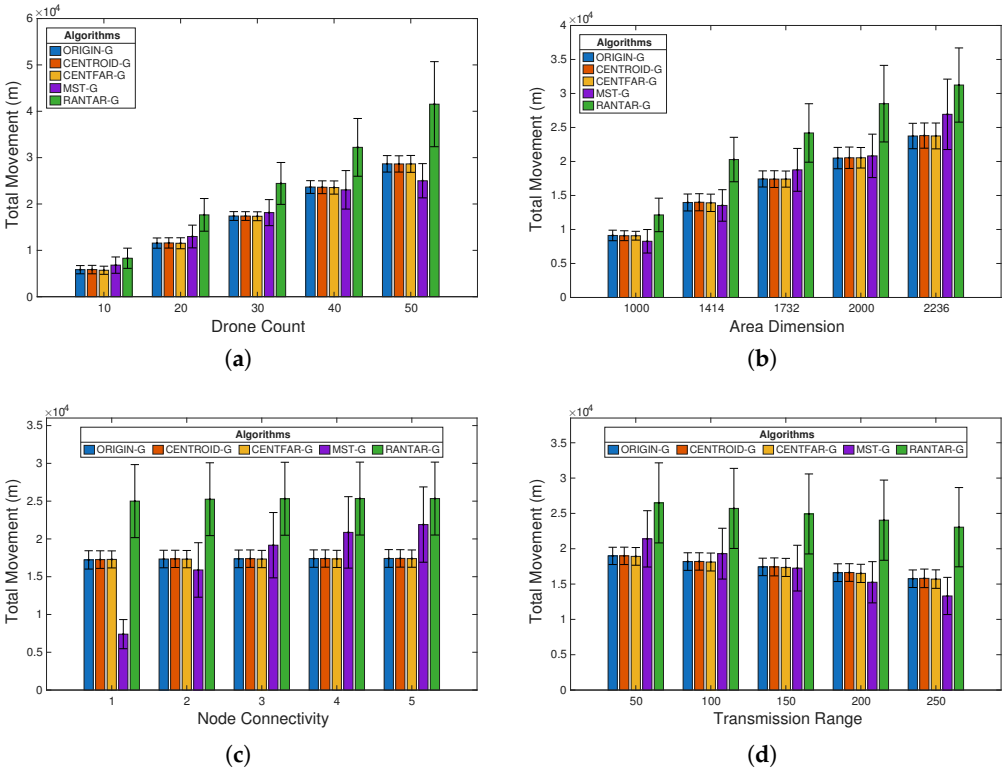


Figure 10. Total movement of group movement-based algorithms (**a**) against drone count, (**b**) against network area, (**c**) against k, (**d**) against T_r.

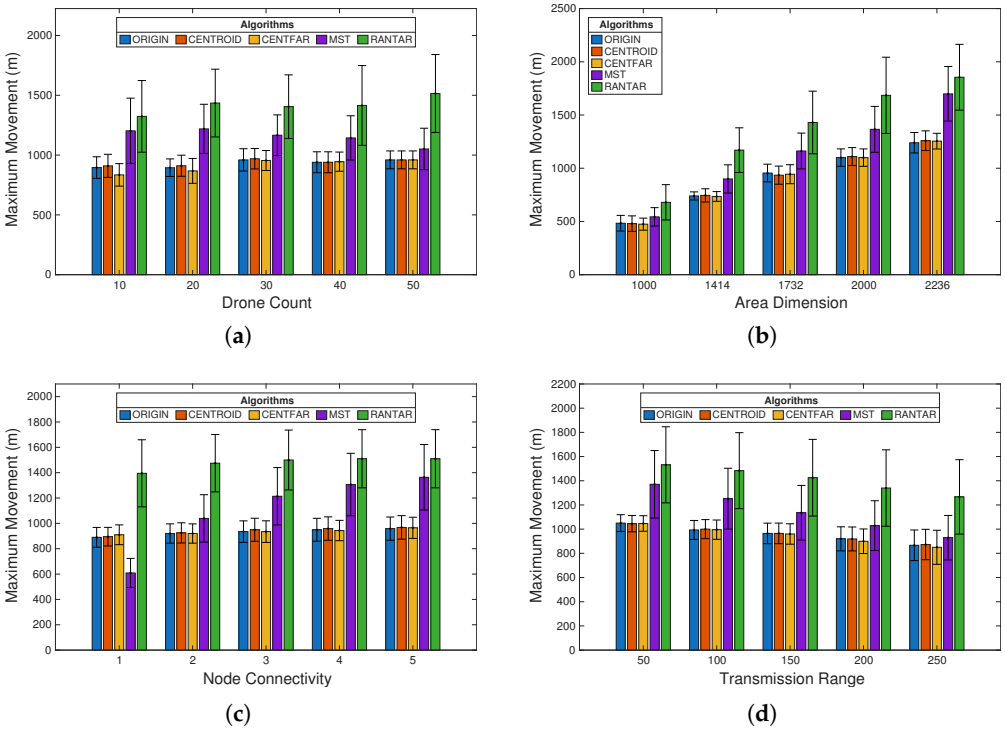


Figure 11. Maximum movement of algorithms (**a**) against drone count, (**b**) against network area, (**c**) against k, (**d**) against T_r.

https://doi.org/10.3390/drones10010016

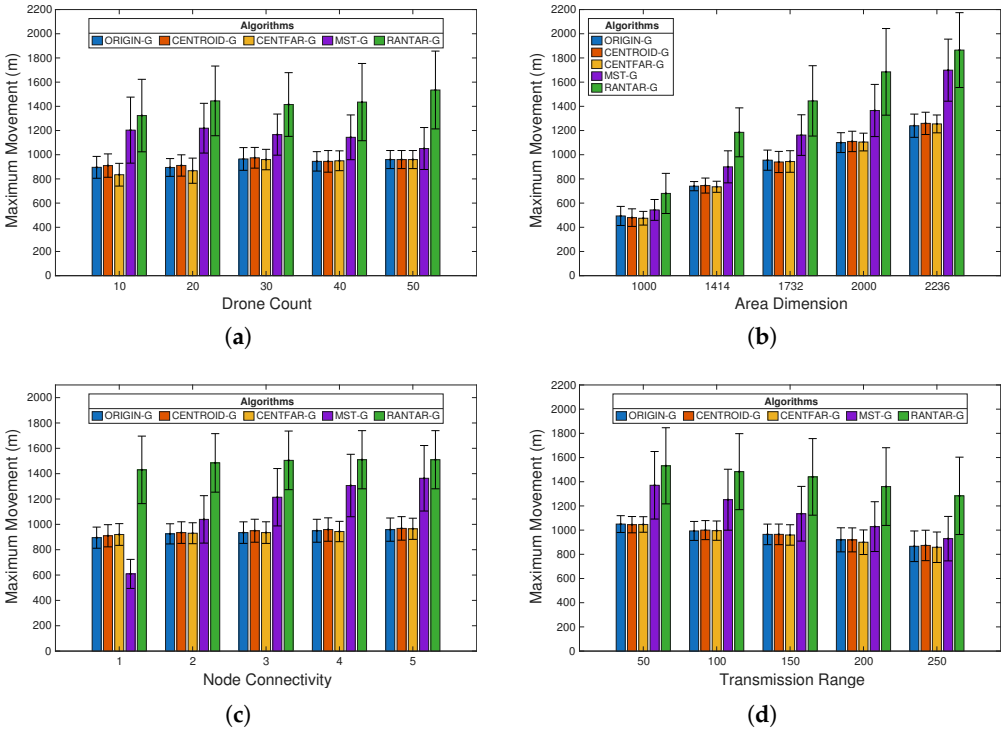


Figure 12. Maximum movement of group movement-based algorithms (**a**) against drone count, (**b**) against network area, (**c**) against k, (**d**) against T_r.

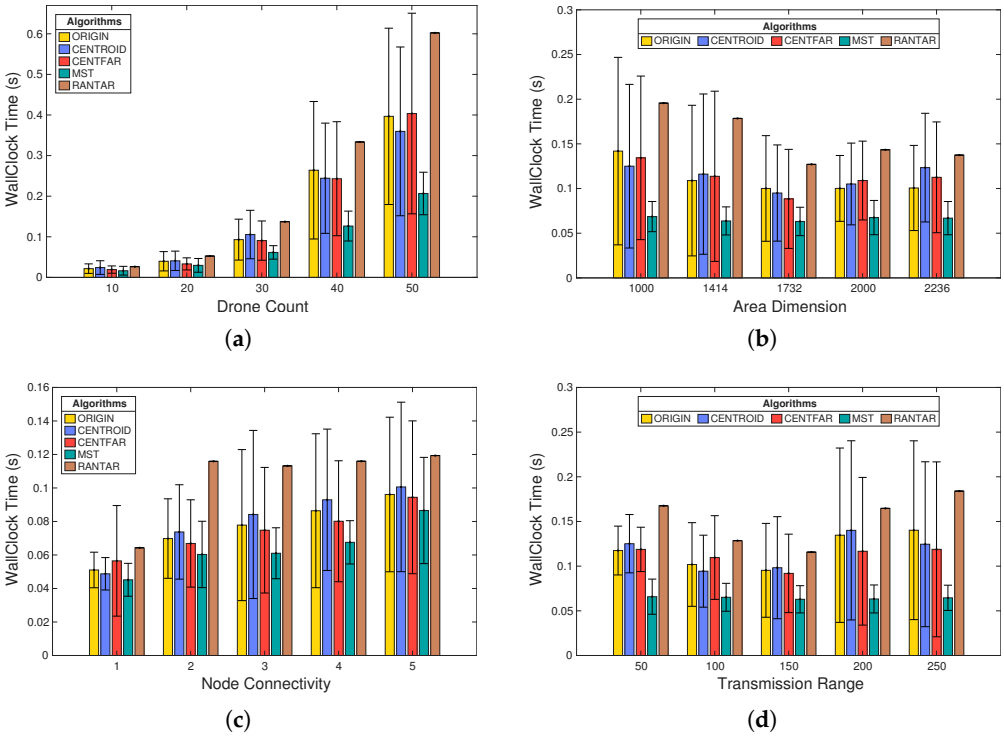


Figure 13. Wallclock time of algorithms (**a**) against drone count, (**b**) against network area, (**c**) against k, (**d**) against T_r.

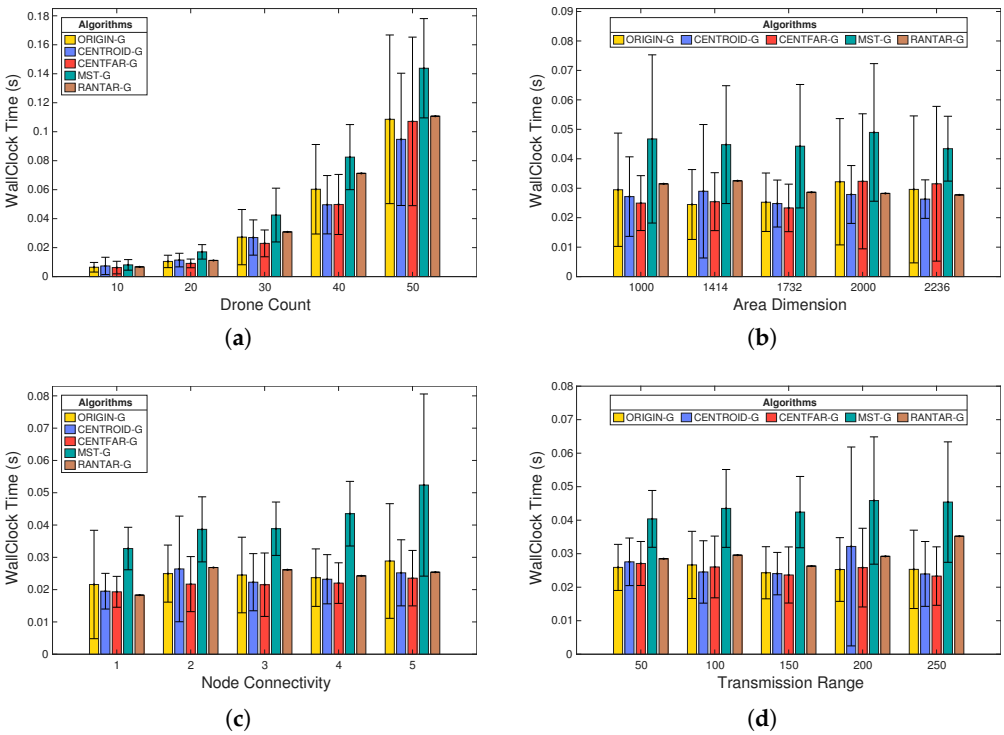


Figure 14. Wallclock time of group movement-based algorithms (**a**) against drone count, (**b**) against network area, (**c**) against k, (**d**) against T_r.

The worst-case time complexity of MST-G is asymptotically better than that of the other algorithms, and unlike the others, its complexity is independent of the network area size. To demonstrate this behavior, we increased the dimensions of the deployment area in Figure 15 and measured the execution times of the group-based movement algorithms. The obtained results are consistent with the theoretical complexity analysis: MST-G operates significantly faster than the other algorithms, particularly as the area size increases. Moreover, this indicates that the test areas provided in Figure 14b are insufficient for MST-G to demonstrate its effective performance.

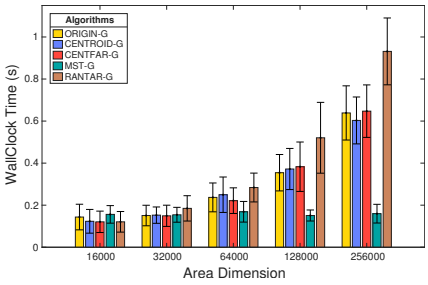


Figure 15. Wallclock time of group movement-based algorithms against large network area.

6. Discussion and Conclusions

Drone swarms are increasingly used in critical civil applications like search-and-rescue, agriculture, machine maintenance, and healthcare where maintaining network connectivity is crucial for effective communication and coordination. However, drones face risks such as harsh weather, battery limitations, and deliberate destruction, which can lead to failures that fragment the network. To address this, building k-connected networks—where each pair of drones is linked by k independent paths—can significantly enhance fault tolerance. As the value of k increases, the swarm becomes more resilient against drone failures, ensuring continued operation even under adverse conditions.

The study introduced the ORIGIN, CENTROID, and CENTFAR algorithms that guided drones to move toward different central points to restore network connectivity. Additionally, it proposed an MST-based algorithm that directed drones along a minimum spanning tree and theoretically guaranteed achieving k-connectivity. Both individual and group-based movement strategies were explored, where k-connectivity was checked either after each drone's move or after all drones moved together.

Extensive simulations evaluated the performance and scalability of these algorithms under varying network sizes, drone counts, k values, and transmission ranges. The MST algorithm performs best in scenarios with low k values and high transmission ranges offering the lowest total movement and execution time, but it performs poorly in terms of maximum movement. This behavior arises from the structural nature of MST: because movement decisions follow a tree hierarchy, nodes may need to traverse long parent–child paths in sparse networks, amplifying individual movement but reducing overall repositioning cost. Although the total movement performance of the CENTER-based algorithms is nearly identical, the CENTFAR algorithm achieves a slightly better maximum movement performance compared to the others. The CENTFAR algorithm outperforms the MST under higher k-connectivity requirements. This advantage stems from its reliance on the farthest-apart drone pair, allowing it to efficiently shrink extreme dispersion and rebalance networks, which becomes increasingly beneficial as area size grows. Group-based movement strategies reduce execution time due to fewer iterations, but they result in slightly higher total and maximum movement compared to individual node movement. Overall, MST is preferable for low-k, large transmission-range scenarios, while CENTFAR is more effective in high-k network deployments, largely due to their distinct structural responses to varying network densities. Table 4 presents a consolidated comparison of the proposed algorithms.

Table 4. Experimental Comparison of Proposed Algorithms.

Algorithm	Variant	Performance Considerations
CENTER-based Algorithms	Individual	The total movement performance of the CENTER-based algorithms are similar. However, the CENTFAR algorithm achieves slightly lower maximum movement (approximately 1%) compared to CENTROID and ORIGIN. The CENTFAR algorithm outperforms the MST under higher k-connectivity requirements, with consistently lower total and maximum movement, achieving up to 21% lower total movement and up to 29% lower maximum movement. In addition, CENTFAR achieves up to 11% better total movement performance and approximately 1% better maximum movement performance compared to CENTFAR-G.
	Group	Group-based CENTER algorithms reduce convergence time due to fewer iterations, but they result in higher total and maximum movement compared to individual node movement-based CENTER algorithms. CENTFAR-G executes up to 10 times faster than CENTFAR.
MST-based Algorithms	Individual	The MST algorithm performs best in scenarios with low k values and high transmission ranges offering the lowest total movement and execution time, achieving up to 57% lower total movement and up to 20% lower execution time than CENTFAR. The MST algorithm exhibits nearly the same maximum movement performance as MST-G while achieving up to 9% better total movement performance.
	Group	The MST-G algorithm converges faster than MST, but this improvement in convergence time comes at the cost of increased total movement relative to individual node movements. Overall, MST-G operates up to 38% faster than MST.

Although the proposed algorithms are effective in restoring k-connectivity in simulated environments, several practical considerations must be addressed when transitioning to real-world UAV deployments. First, mobility-induced energy consumption plays a critical role, as repositioning maneuvers constitute one of the dominant contributors to battery drainage. The feasibility of executing multiple repositioning steps therefore depends on the remaining energy budget, mission duration, and the availability of charging or battery-swapping infrastructure. Second, the algorithms assume accurate positional information; however, real UAVs rely on GPS or onboard localization systems that may experience degradation due to multipath effects, atmospheric disturbances, or urban canyon environments. Such localization errors can influence both the accuracy of connectivity estimation and the precision of movement commands. Third, any movement strategy must operate in parallel with reliable collision avoidance mechanisms (either through onboard sensing or cooperative protocols) to ensure safe maneuvering, especially in dense swarms or constrained airspace. Furthermore, external factors such as wind, turbulence, and temporary communication delays may affect the execution of movement commands and should be considered in practical implementations. While these factors do not invalidate the theoretical design of our algorithms, they highlight the need for additional system-level integration and robustness enhancements, making practical applicability an important direction for future work. Another promising direction for future work is the design and development of population-based metaheuristic algorithms. Such methods could potentially explore the continuous search space more effectively and provide improved repositioning strategies under complex connectivity constraints. Incorporating these advanced optimization techniques may further enhance performance, especially in large-scale or highly dynamic drone network scenarios.

Author Contributions: Conceptualization, O.C. and Z.A.D.; methodology, O.C. and Z.A.D.; software, O.C.; validation, O.C. and O.D.; formal analysis, O.D. and M.C.; investigation, M.C.; resources, O.C. and Z.A.D.; data curation, O.C.; writing—original draft preparation, Z.A.D. and O.D.; writing—review and editing, M.C.; visualization, O.C. and Z.A.D.; supervision, Z.A.D. and O.D.; funding acquisition, Z.A.D. and M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Ege University under Grant No 32182. This work was also funded by Flanders Make.

Data Availability Statement: The dataset used in this study are available in the GitHub repository: https://github.com/netosresearch-ege/DAKRA (accessed on 1 Sepetember 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Fontenla-Carrera, G.; Aldao Pensado, E.; Veiga-López, F.; González-Jorge, H. Efficient offshore wind farm inspections using a support vessel and UAVs. *Ocean. Eng.* **2025**, *332*, 121416. [CrossRef]
2. Guebsi, R.; Mami, S.; Chokmani, K. Drones in Precision Agriculture: A Comprehensive Review of Applications, Technologies, and Challenges. *Drones* **2024**, *8*, 686. [CrossRef]
3. Xiong, T.; Liu, F.; Liu, H.; Ge, J.; Li, H.; Ding, K.; Li, Q. Multi-Drone Optimal Mission Assignment and 3D Path Planning for Disaster Rescue. *Drones* **2023**, *7*, 394. [CrossRef]
4. Zhang, L.; Ma, X.; Zhuang, Z.; Xu, H.; Sharma, V.; Han, Z. Q-Learning Aided Intelligent Routing With Maximum Utility in Cognitive UAV Swarm for Emergency Communications. *IEEE Trans. Veh. Technol.* **2023**, *72*, 3707–3723. [CrossRef]
5. Akram, V.K.; Dagdeviren, O. On Hardness of Connectivity Maintenance Problem in Drone Networks. In Proceedings of the 2018 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Batumi, Georgia, 4–7 June 2018 ; pp. 1–5.

6. Liu, Y.; Xie, J.; Luo, X.; He, Y. SDN-based Connectivity Restoration for Partitioned Flying Ad hoc Networks. In Proceedings of the 2023 9th International Conference on Computer and Communications (ICCC), Chengdu, China, 8–11 December 2023; pp. 1027–1032. [CrossRef]
7. Zhang, J.; Wang, T.; Wang, J.; Du, W.; Zheng, D.; Wang, S.; Li, Y. Multi-UAV Collaborative Surveillance Network Recovery via Deep Reinforcement Learning. *IEEE Internet Things J.* **2024**, *11*, 34528–34540. [CrossRef]
8. Li, J.; Yi, P.; Duan, T.; Zhang, Z.; Li, J.; Wang, Y.; Yu, J. Fast connectivity restoration of UAV communication networks based on distributed hybrid MADDPG and APF algorithm. *Ad Hoc Netw.* **2025**, *171*, 103785. [CrossRef]
9. Rezwan, S.; Choi, W. A Survey on Applications of Reinforcement Learning in Flying Ad-Hoc Networks. *Electronics* **2021**, *10*, 449. [CrossRef]
10. Gong, X.; Plets, D.; Tanghe, E.; De Pessemier, T.; Martens, L.; Joseph, W. An efficient genetic algorithm for large-scale transmit power control of dense and robust wireless networks in harsh industrial environments. *Appl. Soft Comput.* **2018**, *65*, 243–259. [CrossRef]
11. Senturk, I.F.; Akkaya, K.; Janansefat, S. Towards realistic connectivity restoration in partitioned mobile sensor networks. *Int. J. Commun. Sys.* **2016**, *29*, 230–250. [CrossRef]
12. Wang, S.; Mao, X.; Tang, S.; Li, X.; Zhao, J.; Dai, G. On "Movement-Assisted Connectivity Restoration in Wireless Sensor and Actor Networks". *IEEE Trans. Parall. Distr. Sys.* **2011**, *22*, 687–694. [CrossRef]
13. Uzun, E.; Senel, F.; Akkaya, K.; Yazici, A. Distributed connectivity restoration in Underwater Acoustic Sensor Networks via depth adjustment. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015; pp. 6357–6362.
14. Zeng, Y.; Xu, L.; Chen, Z. Fault-Tolerant Algorithms for Connectivity Restoration in Wireless Sensor Networks. *Sensors* **2015**, *16*, 3. [CrossRef]
15. Zhang, Y.; Wang, J.; Hao, G. An autonomous connectivity restoration algorithm based on finite state machine for wireless sensor-actor networks. *Sensors* **2018**, *18*, 153. [CrossRef] [PubMed]
16. Akram, V.K.; Dagdeviren, O.; Tavli, B. Distributed *k*-Connectivity Restoration for Fault Tolerant Wireless Sensor and Actuator Networks: Algorithm Design and Experimental Evaluations. *IEEE Trans. Reliab.* **2020**, *70*, 1112–1125. [CrossRef]
17. Yang, P.; Freeman, R.A.; Gordon, G.J.; Lynch, K.M.; Srinivasa, S.S.; Sukthankar, R. Decentralized estimation and control of graph connectivity in mobile sensor networks. *Automatica* **2008**, *46*, 2678–2683. [CrossRef]
18. Akkaya, K.; Senel, F.; Thimmapuram, A.; Uludag, S. Distributed Recovery from Network Partitioning in Movable Sensor/Actor Networks via Controlled Mobility. *IEEE Trans. Comput.* **2010**, *59*, 258–271. [CrossRef]
19. Liu, X. Survivability-Aware Connectivity Restoration for Partitioned Wireless Sensor Networks. *IEEE Commun. Lett.* **2017**, *21*, 2444–2447. [CrossRef]
20. Dimarogonas, D.V.; Johansson, K.H. Decentralized connectivity maintenance in mobile networks with bounded inputs. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 19–23 May 2008; pp. 1507–1512.
21. Dagdeviren, Z.A.; Akram, V.K.; Dagdeviren, O.; Tavli, B.; Yanikomeroglu, H. k-Connectivity in Wireless Sensor Networks: Overview and Future Research Directions. *IEEE Netw.* **2022**, *37*, 140–145. [CrossRef]
22. Chen, H.; Wu, C.; Wu, M.; Xu, B.; Fan, P. Survey of Connectivity Restoration in 3D Wireless Ad Hoc/Sensor Networks. *Int. J. Distrib. Sens. Netw.* **2024**, *2024*, 1–27. [CrossRef]
23. Lu, X.; Xie, Z.; Yan, P. Connectivity Repair for Maritime Wireless Sensor Networks with Obstacles: Connectivity Repair for Maritime Wireless Sensor Networks. *Wirel. Pers. Commun.* **2024**, *139*, 1477–1497. [CrossRef]
24. Zhao, Z.; Liu, C.; Guang, X.; Zhao, Z.; Qu, W. A Reliability-Driven Topology Restoration Strategy for Underwater Wireless Sensor Networks in Dynamic Ocean Environments. *IEEE Internet Things J.* **2025**, *12*, 10506–10523. [CrossRef]
25. Almasaeid, H.M.; Kamal, A.E. On the Minimum k-Connectivity Repair in Wireless Sensor Networks. In Proceedings of the 2009 IEEE International Conference on Communications, Dresden, Germany, 14–18 June 2009; pp. 1–5.
26. Han, X.; Cao, X.; Lloyd, E.L.; Shen, C.C. Fault-Tolerant Relay Node Placement in Heterogeneous Wireless Sensor Networks. *IEEE Trans. Mob. Comput.* **2009**, *9*, 643–656. [CrossRef]
27. Lee, S.; Younis, M.; Lee, M. Connectivity restoration in a partitioned wireless sensor network with assured fault tolerance. *Ad Hoc Netw.* **2015**, *24*, 1–19. [CrossRef]
28. Liu, X.; Liu, A.; Qiu, T.; Dai, B.; Wang, T.; Yang, L. Restoring Connectivity of Damaged Sensor Networks for Long-Term Survival in Hostile Environments. *IEEE Int. Things J.* **2020**, *7*, 1205–1215. [CrossRef]
29. Shi, G.; Ishat-E-Rabban, M.; Bonner, G.; Tokekar, P. Fast k-connectivity restoration in multi-robot systems for robust communication maintenance: Algorithmic and learning-based solutions. *Auton. Robot.* **2025**, *49*, 34. [CrossRef]
30. Zhu, L.; Ma, C.; Li, J.; Lu, Y.; Yang, Q. Connectivity-Maintenance UAV Formation Control in Complex Environment. *Drones* **2023**, *7*, 229. [CrossRef]

31. Asci, M.; Dagdeviren, Z.A.; Akram, V.K.; Yildiz, H.U.; Dagdeviren, O.; Tavli, B. Enhancing drone network resilience: Investigating strategies for k-connectivity restoration. *Comput. Stand. Interfaces* **2025**, *92*, 103941. [CrossRef]
32. *IEEE Std 802.11-2020*; IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks—Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE: New York, NY, USA, 2021.
33. Henzinger, M.R.; Rao, S.; Gabow, H.N. Computing Vertex Connectivity: New Bounds from Old Techniques. *J. Algorithms* **2000**, *34*, 222–250. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.