Enhancing Drone Network Resilience: Investigating Strategies for k-Connectivity Restoration

Mustafa Asci[a,b], Zuleyha Akusta Dagdeviren[c,*], Vahid Khalilpour Akram[a], Huseyin Ugur Yildiz[d], Orhan Dagdeviren[c], Bulent Tavli[e]

[a]International Computer Institute, Ege University, Izmir, 35100, Turkey
[b]Republic of Turkiye Ministry of Treasury and Finance, Ankara, 06510, Turkey
[c]Department of Computer Engineering, Ege University, Izmir, 35100, Turkey
[d]Department of Electrical and Electronics Engineering, TED University, Ankara, 06420, Turkey
[e]Department of Electrical and Electronics Engineering, TOBB University of Economics and Technology, Ankara, 06560, Turkey

Abstract

Drones have recently become more popular due to technological improvements that have made them useful in many other industries, including agriculture, emergency services, and military operations. Coordination of communication amongst drones is often required for the efficient performance of missions. With an emphasis on building robust k-connected networks and restoration procedures, this paper investigates the relevance of connection in drone swarms. Specifically, we tackle the k-connectivity restoration problem, which aims to create k-connected networks by moving the drones as little as possible. We propose four novel approaches, including an integer programming model, an integer programming-based heuristic approach, a node converging heuristic, and a cluster moving heuristic. Through extensive measurements taken from various drone networking setups, we provide a comparative analysis of the proposed approaches. Our evaluations reveal that the drone movements produced by the integer programming-based heuristics are nearly the same as the original mathematical formulation, whereas the other heuristics are favorable in terms of execution time.

Keywords: drone networks, reliability, k-connectivity, mathematical programming, graph theory

1. Introduction

Advancements in drone technology have seen rapid growth in recent years. Currently, drone technologies are used for many purposes, including large-scale agricultural field operations, product delivery, emergency medical services, firefighting, military surveillance, and search and rescue missions. Completing these duties is more efficient when using a drone fleet that can communicate with each other, rather than relying on a single aerial vehicle [1, 2, 3, 4]. Uninterrupted communication among the drones in the fleet is essential for the mission's overall success. Drone malfunctions in the network might cause communication to cease, resulting in mission failure.

Connectivity is a key aspect of graph theory when used in computer networks. If a route connects every pair of nodes in the network, then the network is considered connected. The network's connectivity may be disrupted if a node experiences a failure or malfunction, contingent upon the specific topology. In such an occurrence, links among other still-functional nodes are severed, resulting in the premature loss of several active resources [5].

A drone network is considered k-connected if a minimum of k independent pathways connect any pair of nodes. In a k-connected drone network, the network remains connected even if up to k-1 nodes fail. A drone network that is k-connected (especially with a high value of k) enhances network resilience against disruptions and improves communication performance due to the availability of alternate communication channels when compared to a 1-connected network. Constructing a k-connected network by applying minimum drone movements is of utmost importance as the problem is in NP-Hard [6, 7].

In this study, we investigate the k-connected network construction problem in which a k-connected drone network is established, and in the event of network degradation, connectivity restoration is achieved to preserve the k value of the network with minimum drone movements. The contributions of this study are listed as follows:

1. Although there are various k-connectivity restoration algorithms in the literature for wireless sensor networks and ad hoc networks with fixed node positions to ensure coverage, to the best of our knowledge, there is no study that divides the drone network area into a grid and achieves k-connectivity for this network. Our paper is the first study in this manner.
2. We first provide the mathematical formulation of the problem through an integer programming model to construct a k-connected network by optimizing the total movement of drones located on an area divided

*Corresponding Author

into grids.

3. To reduce the computational complexity of the mathematical model, we propose an integer programming-based heuristic by creating a spanning tree in which each drone is connected to at least k neighboring drones.
4. Moreover, we propose two heuristic algorithms to construct k-connected drone networks. In the first heuristic, drones move towards the center position to construct a k-connected topology. A cluster-based movement strategy is introduced in the second proposed approach.
5. We provide extensive performance evaluations by varying drone count, area, and k values, which show that the drone movements generated by the heuristic based on integer programming almost exactly match those of the original mathematical formulation, while the other heuristics exhibit advantages in terms of execution speed.

The rest of the paper is organized as follows. In Section 2, related literature is surveyed. Our network and system models are given in Section 3. Proposed approaches are explained in Section 4. Comprehensive performance evaluations are provided in Section 5. Conclusions are drawn in Section 6.

2. Related Work

Link repair techniques primarily aim to promptly fix damaged connections in the network in order to restore k-connectivity. This may be achieved by using routing protocols or modifying the transmission range. In Gong et al.'s study [8], the authors introduced a link repair method that utilizes many separate pathways to efficiently restore k-connectivity after a link failure. Multiple studies in the literature adhere to a similar notion to reestablish k-connectivity, as shown by various sources [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. The paper in [22] discusses how to maintain k-connectivity by analyzing intersections of transmission ranges. The article by Zeng et al. [12] introduces a k-connectivity restoration method and a partial restoration technique that maintain k-connectivity among the original nodes for $k = 2$ and k=3, respectively. In some cases, the transmission range of the nodes may not be increased, or the lifetime of the network may be shortened as the amount of energy consumed will increase as the transmission power is increased. Alternatively, k-connected network construction be accomplished by installing additional infrastructure nodes [23, 22, 24, 25, 26]. However, deploying more nodes will always incur expenses due to the accompanying hardware and operations.

The CMH technique, given in [15], achieves k-connectivity restoration using mobility-based methods, utilizing the 2-hop local graph data of nodes. This algorithm has several drawbacks. The technique was not designed for a topology with mobile nodes. Executing shortest route methods on a completely mobile topology might result in significant message traffic. The CMH method does not take into account the impact of a node's movement on the k-connectivity when selecting a node to move, which is another drawback.

Akram et al. introduced a distributed technique called LINAR for k-connectivity restoration in [17]. The approach utilizes 2-hop local graphs to assess the nodes' state. Imaginary neighbors are inserted between the neighbors that are 2 hops distant in the local graph until it achieves k-connectivity if the 2-hop local graph is not already k-connected. The LINAR algorithm has several specific issues that impact its effectiveness and efficiency. Firstly, it relies on a brute-force technique to identify the imaginary neighbors that need to be included in a node's local network when determining its state. This initial brute-force approach is problematic because it does not offer a solution in polynomial time, making it inefficient for large-scale networks. The anticipated efficiency in terms of polynomial time is not achieved, leading to potentially significant computational delays. Moreover, this brute-force method is not always reliable in reaching a solution. There are instances where it fails to identify all necessary imaginary neighbors, which are crucial for the accurate functioning of the network.

The paper by Wang et al. [10] introduces a centralized technique called MCCR, which involves constructing a bipartite network and using a maximum matching algorithm. The core method was enhanced in terms of time complexity by the TAPU algorithm introduced in [13]. The approach, as described in Akram et al. [27] and Dagdeviren et al. [28], relocates a node along the shortest route tree to the position of a failed node if its removal does not disrupt the k-connectivity. DPkCR algorithm given in [29] has two phases where the states of the nodes are identified in the first phase and the recovery procedure is executed in the latter phase. The algorithms described achieve k-connectivity restoration by moving active nodes to the positions of failed ones, indicating that the initial node positions cannot be expanded. Unlike these methods, our proposed approaches deploy randomly distributed drones within a grid-divided area to create k-connected topologies. This strategy provides greater flexibility and robustness, as it allows for varying initial positions and adapts to different conditions, resulting in a more dynamic and effective solution for maintaining network connectivity.

3. Network and System Modeling

In this section, we outline our network and system models as well as our assumptions.

3.1. Network Model

The assumptions for the network area utilized in the study are listed below:

Table 1: The list of symbols and their definitions used in this paper.

Symbol	Description
ρ	Iteration counter used in NC and CM algorithms.
ρ_{max}	Maximum iteration counter used in NC and CM algorithms.
a_{ij}	Binary variable indicating if a drone is relocated from position-i to position-j after setting up the k-connected network.
b_i	Binary variable equal to 1 if the position-i acts as the root node of the spanning tree and 0 otherwise in the IPBH model.
cl	Clusters used in NC and CM algorithms.
cl_{max}	Maximum element cluster used in NC and CM algorithms.
cl_{min}	Minimum element and further cluster used in NC and CM algorithms.
cp	Central position.
d_{ij}	Euclidean distance between the drone positions-i and j (in meters).
E	Set of all links.
f_{ij}^{mn}	Binary variable representing whether a drone going from position-m to destination position-n passes via link (i, j) in the IP model.
f_{ij}	Integer variable representing the total number of drones traveling via link (i, j) in the IPBH model.
g_{ij}	Parameter indicating whether two drones located at position-i and position-j can communicate or not.
G	Drone network graph.
K_{min}	The minimum k value of the network for establishing the k-connectivity.
N	Set of drones.
P_i	Initial drone placements (parameter), which takes 1 if a drone exists at position-i, and 0 otherwise.
P	Initial drone placement vector (i.e., $P = [P_1\ P_2\ \ldots\ P_{\|V\|}]$).
Q_i	Final drone placements (binary variable), which takes 1 if a drone exists at position-i, and 0 otherwise.
R_{max}	Maximum radio range of each drone (in m).
s_{mn}	Binary variable equal to 1 if two drones are positioned at locations m and n after creating a k-connected network.
t	The total movement of drones for establishing the k-connectivity.
V	Set of drone positions.
x_i	Maximum flow permitted in the spanning tree for the position-i, starting from the root node in the IPBH model.

1. Network area is organized as a 2-D grid ($n \times n$ square) with equal dimensions in each cell point. The Euclidean distance between neighboring points is 100 m.
2. Each cell point can accommodate a single drone, ensuring that no two drones occupy the same cell point simultaneously.
3. Drones can only be positioned at grid points, providing a structured and organized deployment within the designated area.
4. Drones can only move between neighboring points, meaning they are restricted to traveling from one grid point to an adjacent one.

3.2. System Model

The assumptions regarding the system and communication models are outlined as follows:

1. All drones possess identical hardware and software capabilities, promoting uniformity in performance and operational characteristics across the entire fleet.
2. If two nodes are separated by, at most, the maximum transmission range (R_{max}) then they are considered connected (i.e., it is acknowledged that the nodes can communicate and are considered neighbors).
3. Communication between nodes is established bidirectionally, allowing for the exchange of data and commands in both directions.
4. Communication between two nodes halts if one of the nodes experiences a failure.

4. Proposed Approaches

We consider a square area that is divided into grids, where several drones (represented by the set N) are put randomly at the junction points (positions) of the grids (represented by the set V). Our goal is to minimize the drone movement between positions to form a k-connected drone network.

We introduce four methods for the solution of the problem stated above, including an integer programming (IP) model, an integer programming-based heuristic approach (IPBH), a node converging heuristic model (NC), and a cluster moving heuristic model (CM). The symbols and their descriptions used in the paper are shown in Table (1).

4.1. Integer Programming Model (IP)

The integer programming model (IP) is introduced in (1)–(16). The IP model sustains a k-connected network by optimizing drone movement to minimize the total movement.

$$\text{Minimize } t = \sum_{i \in V} \sum_{j \in V} a_{ij} d_{ij} \tag{1}$$

subject to:

$$\sum_{j \in V} a_{ij} = P_i, \ \forall i \in V \tag{2}$$

$$\sum_{j \in V} a_{ji} = Q_i, \ \forall i \in V \tag{3}$$

$$f_{ij}^{mn} \leq Q_i, \ \forall i, j, n \in V \tag{4}$$

$$f_{ij}^{mn} \leq Q_j, \ \forall i, j, n \in V \tag{5}$$

$$f_{ij}^{mn} \leq g_{ij}, \ \forall i, j, n \in V \tag{6}$$

$$s_{mn} \leq (Q_m + Q_n - 1), \ \forall m, n \in V \tag{7}$$

$$s_{mn} \leq Q_m, \ \forall m, n \in V \tag{8}$$

$$s_{mn} \leq Q_n, \ \forall m, n \in V \tag{9}$$

$$\sum_{j \in V - \{i\}} f_{ij}^{mn} - \sum_{j \in V - \{i\}} f_{ji}^{mn} = \begin{cases} s_{mn} K_{min}, & \text{if } i = m \\ -s_{mn} K_{min}, & \text{if } i = n \\ 0 & \text{if } i \neq \{m, n\} \end{cases}, \quad \forall m \in V, n \in V - \{m\} \tag{10}$$

$$\sum_{j \in V - \{i\}} f_{ij}^{mn} \leq 1, \ \forall m \in V, n \in V - \{m\}, i \in V - \{m, n\} \tag{11}$$

$$\sum_{j \in V - \{i\}} f_{ji}^{mn} \leq 1, \ \forall m \in V, n \in V - \{m\}, i \in V - \{m, n\} \tag{12}$$

$$a_{ij} \in \{0, 1\}, \ \forall i, j \in V \tag{13}$$

$$Q_i \in \{0, 1\}, \ \forall i \in V \tag{14}$$

$$f_{ij}^{mn} \in \{0, 1\}, \ \forall i, j, m, n \in V \tag{15}$$

$$s_{mn} \in \{0, 1\}, \ \forall m, n \in V \tag{16}$$

The objective function, shown in (1), aims to minimize the total movement of all drones in the drone network (i.e., $t = \sum_{i \in V} \sum_{j \in V} a_{ij} d_{ij}$). The binary variable a_{ij} represents the movement of a drone from position-i to position-j after establishing the k-connected network. In other words, $a_{ij} = 1$ for movement and 0 otherwise. The parameter d_{ij} denotes the Euclidean distance between drone positions i and j (in meters). Constraint (2) specifies the initial positions for drones to depart, based on the parameter P_i (where $P_i = 1$ if there is a drone at position-i and 0 otherwise). Constraint (3) specifies the destination positions for the arrival of drones and saves this information in the binary variable Q_i (if there is a drone at position-i once the k-connected network is set up, $Q_i = 1$; otherwise, $Q_i = 0$).

The binary variable f_{ij}^{mn} in Constraints (4) to (12) is 1 when a drone, starting at position-m and heading to destination position-n, passes via the link (i, j). We define the set of all links as $E = \{(i, j) \mid i \in V, j \in V, i \neq j\}$. f_{ij}^{mn} variables are linked to the binary variable Q_i in Constraints (4) and (5) to model the movement of drones. When Q_i is 0, then f_{ij}^{mn} is also 0. Constraint (6) represents the maximum range constraint. It specifies that if $g_{ij} = 0$ (i.e., $d_{ij} > R_{max}$), then $f_{ij}^{mn} = 0$. The parameter g_{ij} is computed as,

$$g_{ij} = \begin{cases} 1, & d_{ij} \leq R_{max} \\ 0, & d_{ij} > R_{max} \end{cases}, \forall i, j \in V. \tag{17}$$

In other words, drones are unable to go to positions that are more than R_{max} meters away. Constraints (7)–(9) ensure that the binary variable s_{mn} equals 1 when two drones are located at positions m and n, indicating the presence of a flow between these two drones. If there are no drones at positions m and n, there is no flow. Constraint (10) is the flow balancing constraint that enables drone movement for establishing a k-connected network. In this constraint, the parameter K_{min} is the desired level of k-connectivity (i.e., K_{min}-connectivity). Constraints (11) and (12) dictate that a drone can go from position-i to a single position-j and can only arrive at position-i from another single position-j. Finally, Constraints (13) to (16) show the boundaries of the decision variables used in this model.

4.2. Integer Programming-Based Heuristic Approach (IPBH)

The Integer Programming-Based Heuristic Approach (IPBH) reduces the computational complexity of the IP model by creating a spanning tree in which each drone is connected to at least k neighboring drones. The f_{ij}^{mn} variables in the IP model are simplified to f_{ij} in the IPBH to reduce the search space and minimize the computation time. In this part, f_{ij} is an integer variable representing the total number of drones traveling via link $(i, j) \in E$. The IPBH model is described by (18) through (34).

$$\text{Minimize } t = \sum_{i \in V} \sum_{j \in V} a_{ij} d_{ij} \tag{18}$$

subject to:

$$\sum_{j \in V} a_{ij} = P_i, \ \forall i \in V \tag{19}$$

$$\sum_{j \in V} a_{ji} = Q_i, \ \forall i \in V \tag{20}$$

$$f_{ij} \leq (|N| - 1) Q_i, \ \forall i, j \in V \tag{21}$$

$$f_{ij} \leq (|N| - 1) Q_j, \ \forall i, j \in V \tag{22}$$

$$f_{ij} \leq g_{ij}, \ \forall i, j \in V \tag{23}$$

$$\sum_{j \in V - \{i\}} f_{ij} - \sum_{j \in V - \{i\}} f_{ji} = x_i, \ \forall i \in V \tag{24}$$

$$x_i = |N| b_i - Q_i, \ \forall i \in V \tag{25}$$

$$\sum_{i \in V} b_i = 1 \tag{26}$$

$$b_i \leq Q_i, \ \forall i \in V \tag{27}$$

$$\sum_{j \in V-\{i\}} Q_j g_{ij} \geq Q_i K_{min}, \ \forall i \in V \quad (28)$$

$$\sum_{i \in V, i|=*_i=0} Q_i - \sum_{i \in V, i|=*_i=1} (1 - Q_i) \geq 1 \quad (29)$$

$$a_{ij} \in \{0, 1\}, \ \forall i, j \in V \quad (30)$$

$$Q_i \in \{0, 1\}, \ \forall i \in V \quad (31)$$

$$b_i \in \{0, 1\}, \ \forall i \in V \quad (32)$$

$$-N - 1 \leq f_{ij} \leq |N| - 1, \ \forall i, j \in V \quad (33)$$

$$-1 \leq x_i \leq |N| - 1, \ \forall i \in V \quad (34)$$

The objective function of the IPBH approach, given in (18), aims to minimize the total movement of all drones in the drone network. This objective function is identical to the objective function of the IP model (i.e., Constraint (1)). Constraints (19), (20), and (23) have analogous roles to Constraints (2), (3, and (6) in the IP model. Constraints (21) and (22) are the simplified versions of Constraints (4) and (5) from the IP model. They are used to establish maximum limits for the f_{ij} variable. Note that $|N|$ is the total number of drones. Constraint (24) represents the flow balancing constraint. The variable x_i represents the maximum flow permitted in the spanning tree for the position-i, starting from the root node. x_i is calculated in Constraint (25), where b_i is equal to 1 if the drone at position-i serves as the root node of the spanning tree, and 0 otherwise. Constraint (26) guarantees that the spanning tree has only one root node. Constraint (27) ensures that there is a drone at the location designated to be the root node. Constraint (28) ensures that each drone has at least K_{min} neighbor drones. Constraint (29) is the tabu constraint. This constraint guarantees that the new solution differs from the prior one by at least one position. Constraints (30) to (34) show the boundaries of the decision variables.

The IPBH approach attempts to discover a solution using the objective function (18) and the constraints (19)–(28). The k value of the network is determined by using the Ford-Fulkerson method on the obtained solution. If the computed value of k is lower than the specified K_{min} value, the tabu constraint (i.e., Constraint (29)), is included in the model, and the IPBH is solved once again. The tabu constraint is dynamically included in the model, resulting in a cumulative rise in the number of constraints. The tabu constraint guarantees that the initial solution obtained by the model $(Q_1^*, Q_2^*, \cdots, Q_n^*)$ varies from the solution $(Q_1, Q_2, \cdots, Q_n)$ in at least one position. The first portion of the equation represents the sum of variables with a basic solution of zero or one. The second part represents the sum of variables with a partial solution of one or zero.

4.3. Node Converging Heuristic Algorithm (NC)

The proposed NC aims to establish and restore a k-connected network by moving drones towards the center position calculated through arithmetic mean. The distance of each drone to the center position is calculated, and the drone closest to the center is moved towards the center. After each movement, the connectivity value of the network is detected using the findConnectivityValue algorithm [30]. If the k value of the network is less than the parameter K_{min}, the algorithm continues to run. The pseudo-code for NC is provided in Algorithm 1.

Algorithm 1 Node Converging Heuristic Algorithm (NC)

Require: $G(V, E)$, $P = [P_1 \ P_2 \ \ldots \ P_{|V|}]$, K_{min}
  $\rho \leftarrow 0$
  $t \leftarrow 0$
  $\rho_{max} \leftarrow |V|$
  while $findConnectivityValue(G) < K_{min}$ and $\rho < \rho_{max}$ do
    $cp \leftarrow$ Calculate center position using P.
    Pick the drone which is farthest from the cp, $u \in V$ and $u \in P$
    $P \leftarrow P \backslash u$.
    Select the v position which is nearest to the cp, neighbor with position-u, $v \in u \backslash \{V\}$ and $v \in P$.
    Move the drone from position-u to position-v.
    $P \leftarrow P \cup v$
    $t \leftarrow t + d_{uv}$
    $\rho \leftarrow \rho + 1$
  end while
  return t

Figure 1 provides an example with $|V| = 16$ positions, $|N| = 5$ drones, $K_{min} = 2$, and $R_{max} = 180$ m. Initially, the drones are located at positions 1, 4, 7, 13, and 16. In other words, $P = [1001001000001001]$. In the first round, the center position is calculated as $cp = 7$. The drone at position 13 moves to position 7, arriving at position 10 (Figure 1b). The total movement after the first round is $t = 141$ m. In the second round, the center position is again calculated as 7. The drone at position 1 moves to position 6 (Figure 1c). The total movement becomes $t = 282$ m. In the third round, the center position is recalculated as 7. The drone at position 16 moves to position 11 (Figure 1d). The total movement is now $t = 423$ m. In the fourth and final round, the drone at position 4 moves to position 3 (Figure 1e), creating a 2-connected network resulting a total movement of $t = 523$ m. The resulting topology is shown in Figure 1f.

The findConnectivityValue method finds the connectivity value of the network using the Ford-Fulkerson algorithm having time complexity of $O(n^5)$. The node selection process in the algorithm is performed in $O(n)$ time. Since the complexity of the algorithm is $O(n^5 n) + O(n^5 n)$, the overall time complexity is $O(n^6)$.

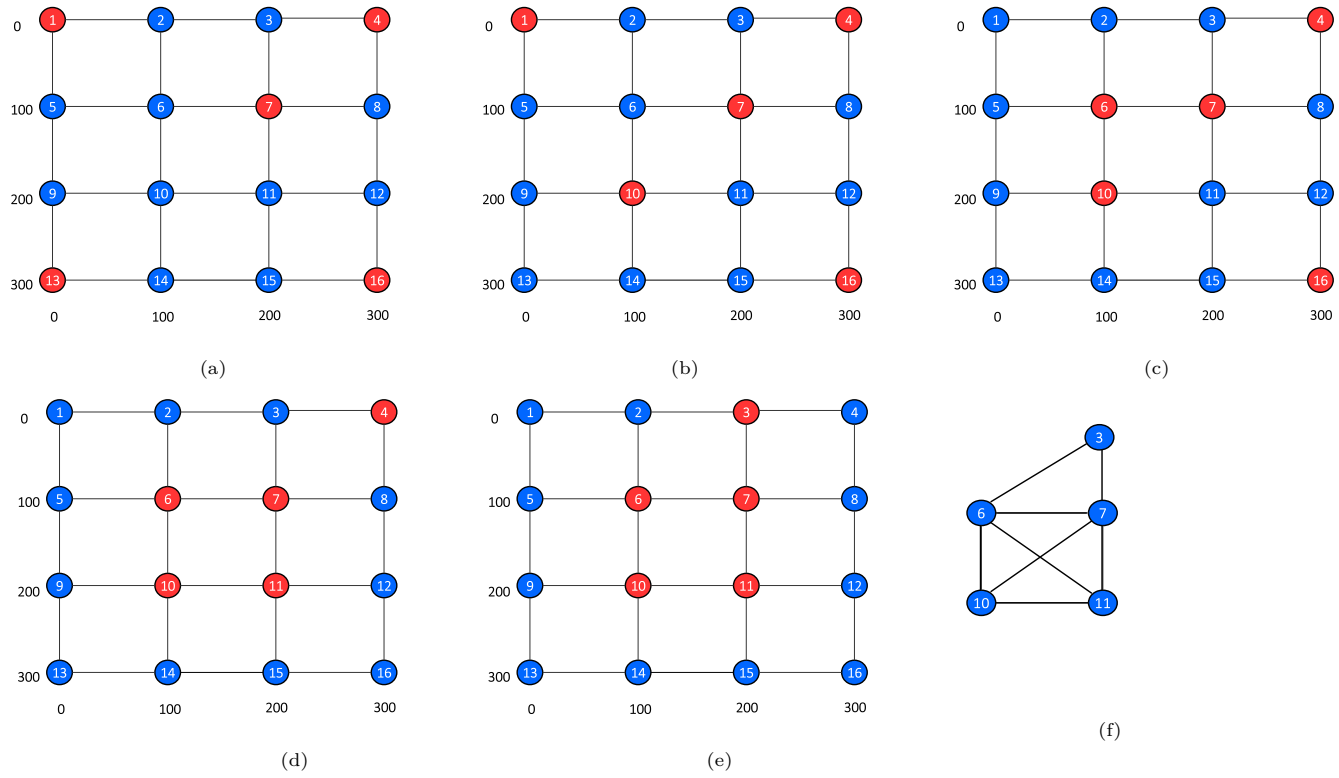


Figure 1: Node converging heuristic example topology (a) initial positions (b) positions at the end of the 1st round (c) positions at the end of the 2nd round (d) positions at the end of the 3rd round (e) final positions (f) 2-connected network.

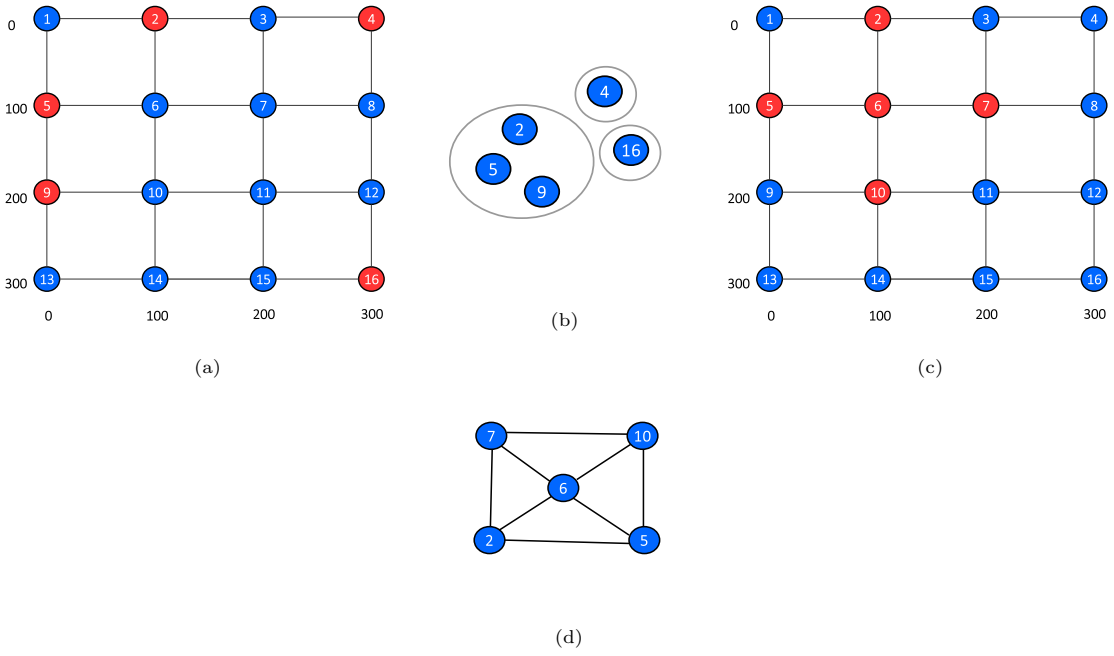


Figure 2: Cluster moving heuristic example topology (a) initial positions, (b) clusters, (c) final positions, (d) 3-connected network.

4.4. Cluster Moving Heuristic Algorithm (CM)

The proposed Cluster Moving Heuristic Algorithm (CM) first creates clusters based on connectivity where connected nodes are included in the same clusters. The drones in the cluster with the maximum number of elements (cl_{max}) move towards the center position of that cluster, while the drones in the cluster with the farthest distance and the minimum number of elements (cl_{min}) move towards their respective center points. The pseudo-code for the CM is provided in Algorithm 2.

Algorithm 2 Cluster Moving Heuristic Algorithm (CM)

Require: $G(V, E)$, $P = [P_1\ P_2\ \dots\ P_{|V|}]$, K_{min}, R_{max}
  $\rho \leftarrow 0$
  $t \leftarrow 0$
  $\rho_{max} \leftarrow |V|$
  while $findConnectivityValue(G) < K_{min}$ and $\rho < \rho_{max}$ do
    $cl \leftarrow$ Create clusters using P and R_{max}.
    if Size of $cl = 1$ then
      Run Node Converging Heuristic Algorithm [1].
    else
      $cl_{max} \leftarrow$ Find the cluster with the most elements in cl.
      $cl_{min} \leftarrow$ Find the cluster with the fewest elements in the cl and the one that is farthest from cl_{max}.
      while until cl_{max} and cl_{min} merge do
        $cp \leftarrow$ Calculate center position using P.
        Pick the drone which is farthest from the cp, $u \in V$ and $u \in P$.
        $P \leftarrow P \setminus u$
        Select the position-v which is nearest to the cp, neighbor with u, $v \in u\setminus\{V\}$ and $v \in P$.
        Move the drone from position-u to position-v.
        $P \leftarrow P \cup v$
        $t \leftarrow t + d_{uv}$
        $\rho \leftarrow \rho + 1$
        $cl_{max} \leftarrow$ Find the cluster with the most elements in cl.
      end while
    end if
  end while
  return t

In Figure 2, an example is given for $K_{min} = 3$ and $R_{max} =$ 180 m with $|V| = 16$ positions and $|N| = 5$ drones. The initial positions of the 5 drones are at positions 2, 4, 5, 9, and 16 (Figure 2a). In other words, $P = [0101000010000001]$. Clusters are formed in Figure 2b. The largest cluster includes the drones at positions 2, 5, and 9, while the other clusters consist of a single drone. The center point of the largest cluster is also $cp = 5$. The drone at position 4 moves to position 7. Drones at positions 9, 5, 2, and 7 form the newest and largest cluster, with position 6 in the center.

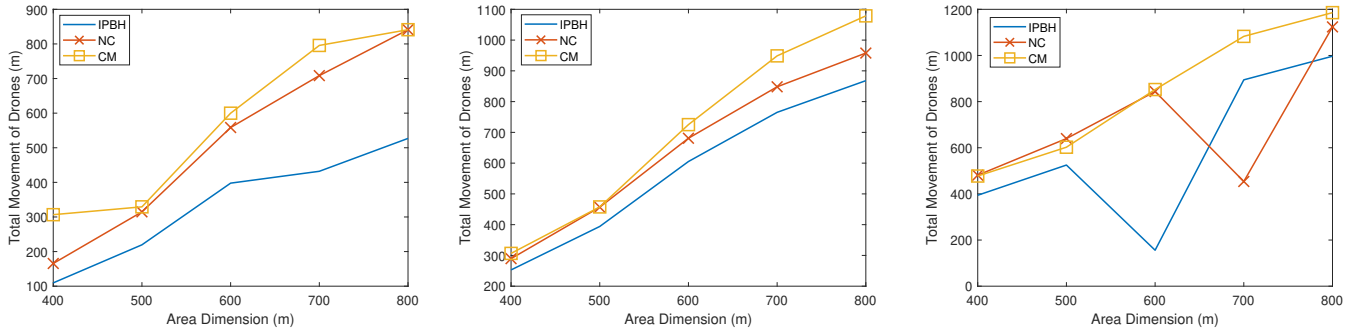
The drone at position 16 is moving to position 11. The largest cluster is formed by drones at positions 9, 5, 2, 7, and 11, with the center position being 6. When the third round begins, the drone at position 1 is moving to position 6. The largest cluster now consists of nodes 6, 10, 11, 7, and 4, forming a single cluster in the network. In this case, the algorithm runs the Algorithm 1. When this algorithm is executed, the drone at position 9 moves to position 6, and the drone at position 11 moves to position 10. Figure 2c shows the final positions of the drones. The total movement is $t = 523$ m. The resulting topology is illustrated in Figure 2d. Similar with the previous proposed heuristic, the time complexity of the algorithm is $O(n^6)$.

5. Performance Evaluation

The algorithms were implemented in Python, leveraging the Pyomo library for mathematical modeling and utilizing CPLEX as the solver for mathematical programs. Additionally, the Networkx library was utilized to create graphs and manage connectivity within the system. All proposed solutions were executed on hardware equipped with a 13th Gen Intel(R) Core(TM) i7-13700H processor running at 2.40 GHz, complemented by 32 GB of RAM and operating on a 64-bit platform. To evaluate the performance of the proposed algorithms, we implemented them in a simulation environment for different drone counts, area sizes, and k values. We utilized square areas of varying sizes: 400 m × 400 m, 500 m × 500 m, 600 m × 600 m, 700 m × 700 m, and 800 m × 800 m. We created random disconnected networks from 5, 10, 15, and 20 drones (i.e., $|N| = \{5, 10, 15, 20\}$) and used the proposed algorithms to establish the networks with $k = 1$, $k = 2$, and $k = 3$. Initially, the drones were placed at random locations inside the square area. The results in this section represent the mean of 100 trials. We measured the total movement of all drones to achieve the desired k, the maximum movement of a single drone, and the wall-clock time of algorithms.

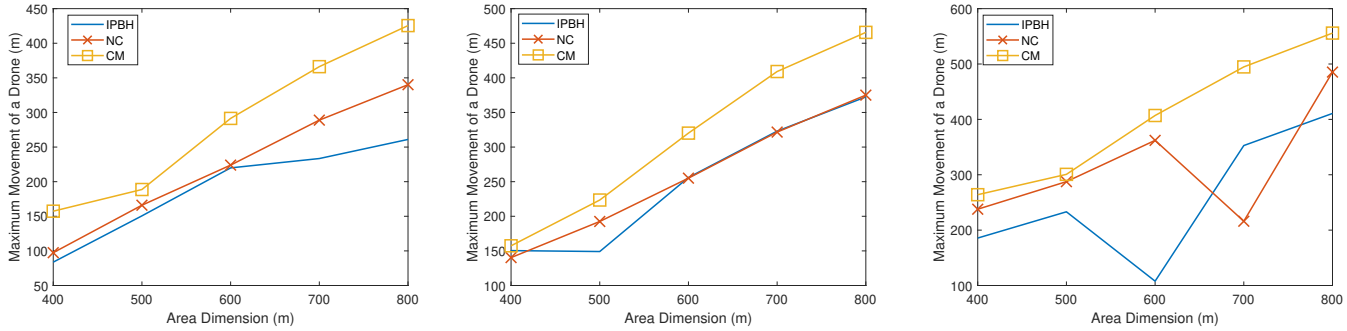
Figure 3 shows the total movement of drones (t) to generate a k-connected network against the area size. Since the IP algorithm could not find a feasible solution for areas larger than 500 m × 500 m, we did not include this algorithm in Figure 3. For $k = 1$ (Figure 3a), IPBH generates much lower movement than other algorithms in all area sizes. In an area of 800 m × 800 m, the IPBH generates 527 m of movement, while the other algorithms generate more than 841 m of movement. The generated total movements of the NC and CM algorithms are close to each other, but generally, the NC algorithm generates shorter movements than CM in all areas. Especially for 400 m × 400 m area, the NC algorithm establishes a 1-connected network by generating 165 m movement, while this value for the CM algorithm is 306 m. This value for IPBH is 109 m. For establishing 2-connected networks, all algorithms generate more movements than 1-connected network (Figure 3b). However, just like the 1-connected network, the IPBH algorithm generates lower movement than the NC and CM algorithms for all area sizes. For $k = 3$, the CM and NC algorithms generate almost the same movements for all areas except 700 m × 700 m (Figure 3c). The IPBH algorithm generates lower movements in all areas, especially in the 600 m × 600 m area, which establishes a 3-connected network by generating less than 200 m in total.

Figure 4 shows the maximum movement of a drone to generate a k-connected network against the area size. Since the IP algorithm could not find a feasible solution for areas larger than 500 m × 500 m, we did not include this algorithm in Figure 4. For $k = 1$ (Figure 4a), the maximum generated movement by IPBH is lower than other



(a) $k = 1$ (b) $k = 2$ (c) $k = 3$

Figure 3: Total movement of drones (t in m) against the area size for various k values.



(a) $k = 1$ (b) $k = 2$ (c) $k = 3$

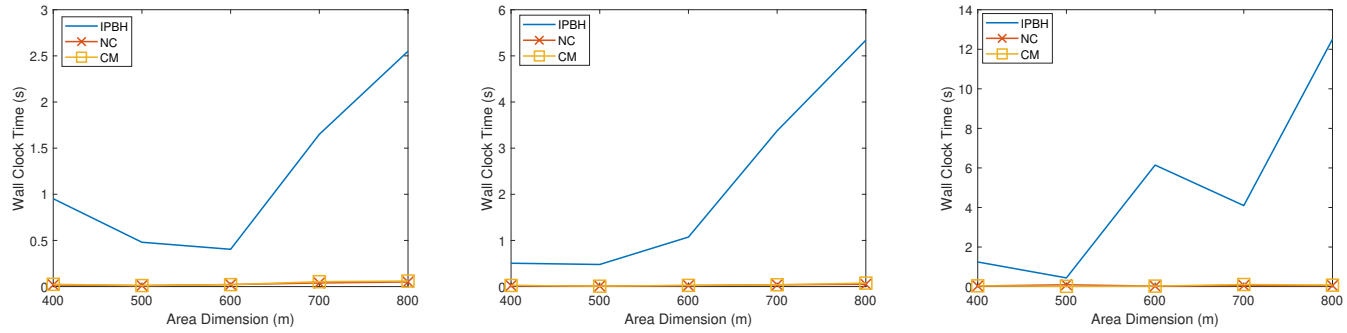
Figure 4: Maximum movement of a drone against the area size for various k values.

algorithms in all area sizes. The difference between IPBH and NC is low up to 600 m × 600 m areas, but after that, the IPBH generates much lower movements. In an 800 m × 800 m area, the maximum movement in IPBH is 261 m, while this value for the NC and CM algorithms is 340 and 425 m, respectively. For $k = 2$, the maximum movement of the IPBH and NC algorithms is almost equal in all areas, while the maximum movement of CM is higher than other algorithms (Figure 4b). Increasing the area size increases the gap between the maximum movement of CM and other algorithms. For $k = 3$, the maximum movement of IPBH is much lower and the maximum movement of CM is higher than other algorithms (Figure 4c).

Figure 5 shows the wall-clock time of algorithms against the area size. For $k = 1$ (Figure 5a), IPBH takes much more time to establish a 1-connected network than the other algorithms, and the gap increases by increasing the area size. For an area of 800 m × 800 m, the IPBH algorithm takes 2.5 s, while the other algorithms take less than 0.06 s. Increasing the k value increases the wall-clock time of all algorithms (Figures 5b and 5c); however, in higher k values, the IPBH takes much more time than the other algorithms. To create a 3-connected network in an area of 800 m × 800 m, the IPBH algorithm takes 12.51 s, while this value for the other algorithms is less than 0.073 s.

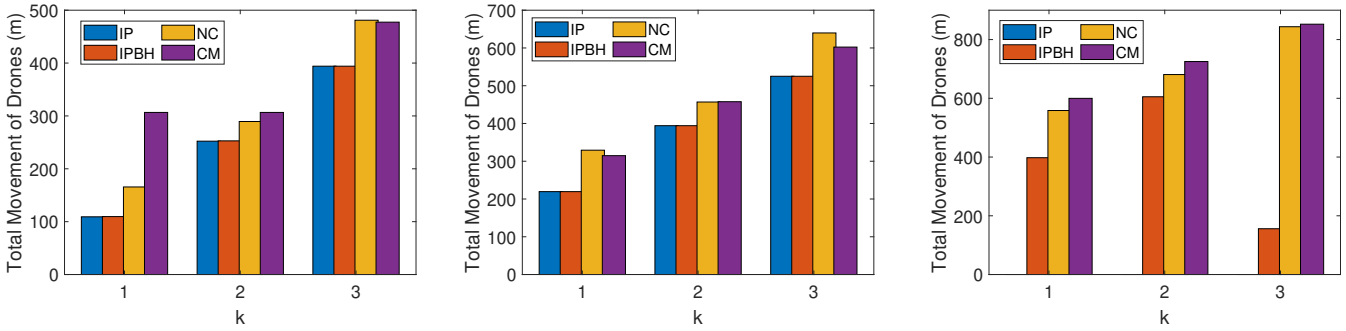
Figure 6 compares the total movement of drones in different algorithms for different k values and area sizes. Generally, increasing the area size increases the total movement in all algorithms. The total movement of IP and IPBH algorithms is equal to and lower than other algorithms for all k values in 400 m × 400 m and 500 m × 500 m areas; however, for 600 m × 600 m area, the IP algorithm could not generate a feasible approach. The total generated movement by the NC and CM algorithms is close to each other in most cases. Figure 7 compares the maximum movement of a drone in different algorithms for different k values and area sizes. The maximum movement of the IP and IPBH algorithms is equal for all k values; however, for 600 m × 600 m area, the IP algorithm could not generate a feasible approach. The maximum movement of the NC and CM algorithms is close to each other and higher than that of the IP and IPBH algorithms in most cases.

Figure 8 compares the wall-clock time of algorithms for different k values and area sizes. The wall-clock time of the IP algorithm is much higher than that of other algorithms in all cases. For 500 m × 500 m area and $k = 3$, the wall-clock time of the IP algorithm is 234.9 s, while the other algorithms finish in less than 2.5 s. For a 500 m × 500 m area, the IP algorithm cannot find a feasible solution, and the IPBH algorithm takes much more time than the other algorithm to establish a k-connected network. For a 600 m × 600 m area and $k = 3$, the wall-clock time of the IPBH algorithm is 6.2 s, while the other algorithms finish in less than 0.22 s. For higher values of k, the IPBH operates



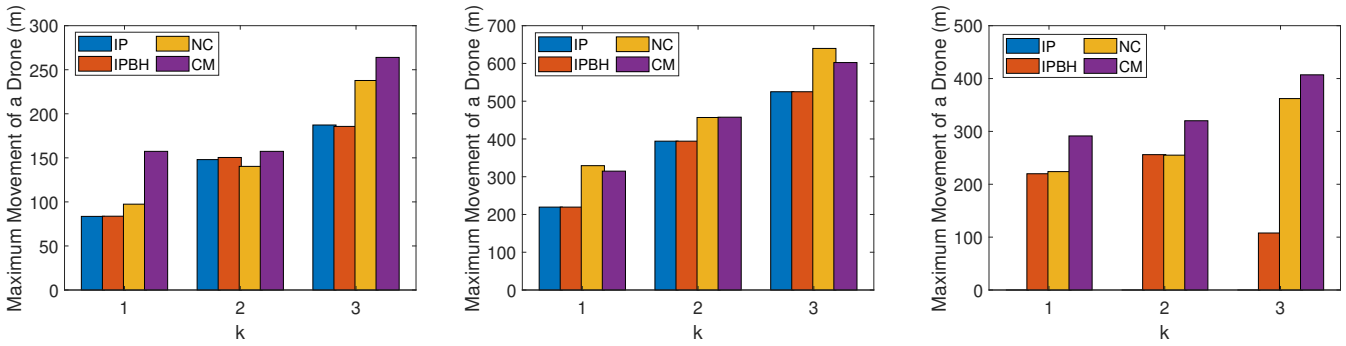
(a) $k = 1$ (b) $k = 2$ (c) $k = 3$

Figure 5: Wall-clock time of algorithms (in s) against the area size for various k values.



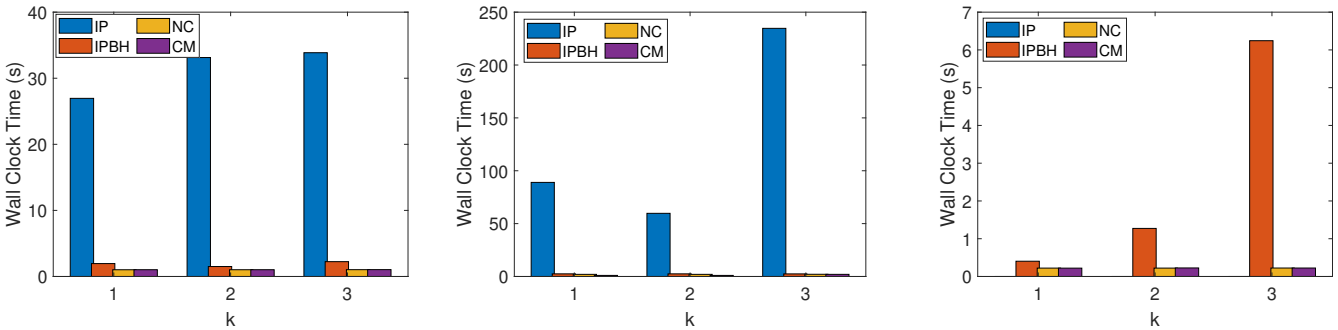
(a) 400 m × 400 m (b) 500 m × 500 m (c) 600 m × 600 m

Figure 6: Total movement of drones in different algorithms for various area sizes.



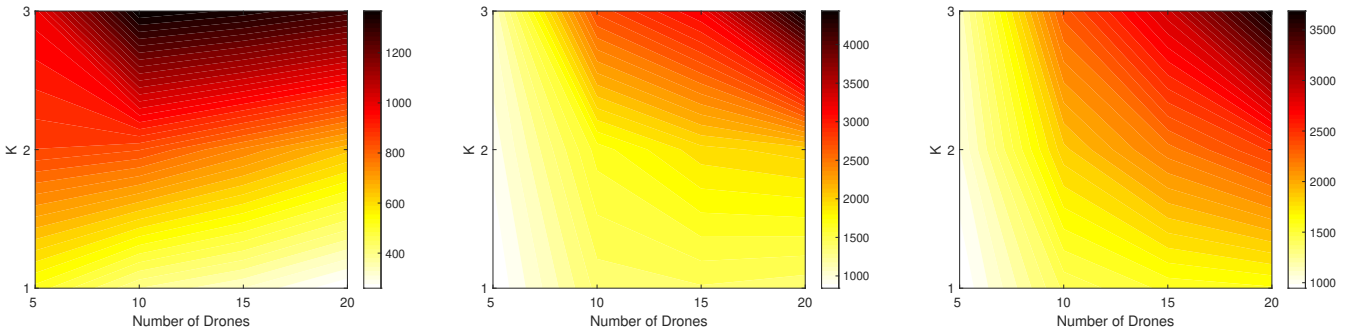
(a) 400 m × 400 m (b) 500 m × 500 m (c) 600 m × 600 m

Figure 7: Maximum movement of a drone in different algorithms for various area sizes.



(a) 400 m × 400 m (b) 500 m × 500 m (c) 600 m × 600 m

Figure 8: Wall-clock time of algorithms for various area sizes.



(a) IPBH (b) NC (c) CM

Figure 9: Total movement of drones against the number of drones and k values.

3.42 times faster than IP, while NC runs approximately 1.169 times faster than CM. Generally, the NC algorithm is about 1.068 times faster than the CM algorithm and 721.706 times faster than the IPBH algorithm.

Figure 9 illustrates the total movement produced by the algorithms for various k values and numbers of drones. Figure 9a shows that the IPBH generates less than 800 m of movement for $k = 1$ and $k = 2$ for all drone counts. For $k = 3$, the total generated movement by IPBH reaches 1200 m when we have more than 10 drones in the network. Figure 9b shows that the NC algorithm generates less than 2500 m of movement for k=1 and k=2 for all drone counts. For $k = 3$, the total generated movement by NC is less than 2000 m when we have less than 10 drones and reaches more than 4000 m for 20 drones. Figure 9c shows that the CM generates less than 2500 m of movement for $k = 1$ and $k = 2$ for all drone counts. For $k = 3$, the total generated movement by CM is less than 1500 m when we have 5 drones and reaches more than 3500 m for 20 drones. For higher values of k, the CM algorithm results in 1.128 times less total movement than the NC algorithm, 1.993 times more than the IPBH algorithm, and 3.509 times more than IP. Generally, the IPBH algorithm generates 1.63 times more total movement than IP. The CM algorithm produces 2.152 times more total movement than IPBH and 1.017 times less than the NC algorithm.

6. Conclusion

Drones have gained increased popularity recently, owing to technological advancements that have extended their utility across various applications. Effective communication coordination among drones is often essential for mission efficiency. In this paper, we focus on establishing resilient k-connected networks and exploring restoration procedures, examining their significance in drone swarms. Specifically, we address the k-connectivity restoration problem, aiming to establish k-connected networks while minimizing drone movement.

We propose four new approaches, including an integer programming model, an integer programming-based heuristic, a node converging heuristic, and a cluster moving heuristic. Through extensive analysis of diverse drone networking scenarios, we offer a comparative assessment of these approaches. Our findings demonstrate that the drone movements generated by the integer programming-based heuristic closely approximate the original mathematical formulation, while the other heuristics offer advantages in terms of execution time. Among the heuristic algorithms, the CM produces the best solution, and the NC algorithm produces the fastest solution.

Acknowledgements

Zuleyha Akusta Dagdeviren acknowledges Ege University, Council of Scientific Research Projects (Project Number: 32182). Vahid Akram, Orhan Dagdeviren and Bulent Tavli express their gratitude to The Scientific and Technological Research Council of Turkey (TUBITAK) for providing grant no 121E500.

References

[1] U. C. Cabuk, M. Tosun, O. Dagdeviren, Y. Ozturk, An architectural design for autonomous and networked drones, in: IEEE Milit. Commun. Conf. (MILCOM), 2022, pp. 962–967.

[2] S. Hussain, K. Mahmood, M. K. Khan, C.-M. Chen, B. A. Alzahrani, S. A. Chaudhry, Designing secure and lightweight user access to drone for smart city surveillance, Computer Standards Interfaces 80 (2022) 103566.

[3] U. C. Cabuk, M. Tosun, O. Dagdeviren, Y. Ozturk, Modeling energy consumption of small drones for swarm missions, IEEE Trans. Intell. Transp. Sys. (2024) 1–14.

[4] Z. Li, Q. Chen, J. Li, J. Huang, W. Mo, D. S. Wong, H. Jiang, A secure and efficient uav network defense strategy: Convergence of blockchain and deep learning, Computer Standards Interfaces 90 (2024) 103844.

[5] Comprehensive survey of uavs communication networks, Computer Standards Interfaces 72 (2020) 103451.

[6] V. K. Akram, O. Dagdeviren, On hardness of connectivity maintenance problem in drone networks, in: IEEE Int. Black Sea Conf. Commun. Netw. (BlackSeaCom), 2018, pp. 1–5.

[7] U. C. Cabuk, V. K. Akram, O. Dagdeviren, Analysis of movement-based connectivity restoration problem in wireless ad-hoc and sensor networks, in: Proc. Tren. Data Eng. Method. Intellig. Sys., 2021, pp. 381–389.

[8] J. Gong, T. Chang, C. Shen, X. Chen, Flight time minimization of UAV for data collection over wireless sensor networks, IEEE J. Sel. Area. Commun. 36 (9) (2018) 1942–1954.
[9] I. F. Senturk, K. Akkaya, S. Janansefat, Towards realistic connectivity restoration in partitioned mobile sensor networks, Int. J. Commun. Sys. 29 (2) (2016) 230–250.
[10] S. Wang, X. Mao, S. Tang, X. Li, J. Zhao, G. Dai, On "movement-assisted connectivity restoration in wireless sensor and actor networks", IEEE Trans. Parall. Distr. Sys. 22 (4) (2011) 687–694.
[11] E. Uzun, F. Senel, K. Akkaya, A. Yazici, Distributed connectivity restoration in underwater acoustic sensor networks via depth adjustment, in: Proc. IEEE Int. Conf. Commun., 2015, pp. 6357–6362.
[12] C. Z. Zeng Y, Xu L, Fault-tolerant algorithms for connectivity restoration in wireless sensor networks, Sensors 16 (1) (2015).
[13] V. K. Akram, O. Dagdeviren, TAPU: Test and pick up-based k-connectivity restoration algorithm for wireless sensor networks, Turk. J. Electr. Eng. Comp. Sci. 27 (2) (2019) 985–997.
[14] Y. Zhang, J. Wang, G. Hao, An autonomous connectivity restoration algorithm based on finite state machine for wireless sensor-actor networks, Sensors 18 (1) (2018) 153.
[15] V. K. Akram, O. Dagdeviren, B. Tavli, Distributed k-connectivity restoration for fault tolerant wireless sensor and actuator networks: Algorithm design and experimental evaluations, IEEE Trans. Reliab. 70 (3) (2020) 1112–1125.
[16] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, R. Sukthankar, Decentralized estimation and control of graph connectivity in mobile sensor networks, in: Proc. Amer. Contr. Conf., 2008, pp. 2678–2683.
[17] V. K. Akram, O. Dagdeviren, B. Tavli, A coverage-aware distributed k-connectivity maintenance algorithm for arbitrarily large k in mobile sensor networks, IEEE/ACM Trans. Netw. 30 (1) (2022) 62–75.
[18] K. Akkaya, F. Senel, A. Thimmapuram, S. Uludag, Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility, IEEE Trans. Comput. 59 (2) (2010) 258–271.
[19] X. Liu, Survivability-aware connectivity restoration for partitioned wireless sensor networks, IEEE Commun. Lett. 21 (11) (2017) 2444–2447.
[20] D. V. Dimarogonas, K. H. Johansson, Decentralized connectivity maintenance in mobile networks with bounded inputs, in: Proc. IEEE Int. Conf. Robot. Autom., 2008, pp. 1507–1512.
[21] Z. A. Dagdeviren, V. K. Akram, O. Dagdeviren, B. Tavli, H. Yanikomeroglu, k-connectivity in wireless sensor networks: Overview and future research directions, IEEE Netw. (2022).
[22] H. M. Almasaeid, A. E. Kamal, On the minimum k-connectivity repair in wireless sensor networks, in: Proc. IEEE Int. Conf. Commun., 2009, pp. 1–5.
[23] X. Han, X. Cao, E. L. Lloyd, C.-C. Shen, Fault-tolerant relay node placement in heterogeneous wireless sensor networks, in: Proc. IEEE Int. Conf. Comput. Commun., 2007, pp. 1667–1675.
[24] S. Lee, M. Younis, M. Lee, Connectivity restoration in a partitioned wireless sensor network with assured fault tolerance, Ad Hoc Netw. 24 (2015) 1–19.
[25] X. Liu, A. Liu, T. Qiu, B. Dai, T. Wang, L. Yang, Restoring connectivity of damaged sensor networks for long-term survival in hostile environments, IEEE Int. Things J. 7 (2) (2020) 1205–1215.
[26] U. C. Cabuk, M. Tosun, V. K. Akram, O. Dagdeviren, Connectivity management in drone networks: Models, algorithms, and methods, in: Proc. Intell. Anal. with Adv. Multi-Ind. Appl., 2021, pp. 128–156.
[27] V. Khalilpour Akram, Z. Akusta Dagdeviren, O. Dagdeviren, M. Challenger, PINC: Pickup non-critical node based k-connectivity restoration in wireless sensor networks, Sensors 21 (19) (2021) 6418.
[28] Z. A. Dagdeviren, V. Akram, Connectivity estimation approaches for internet of things-enabled wireless sensor networks, in: Proc. Emer. Trend. in IoT and Integ. with Data Sci., Cloud Comput., and Big Data Anal., 2022, pp. 104–122.
[29] M. Tosun, U. C. Cabuk, E. Haytaoglu, O. Dagdeviren, Y. Ozturk, DPkCR: Distributed proactive k-connectivity recovery algorithm for UAV-based MANETs, IEEE Trans. Reliab. (2024) 1–15.
[30] V. K. Akram, O. Dagdeviren, On k-connectivity problems in distributed systems, in: Advanced Methods for Complex Network Analysis, IGI Global, 2016, Ch. 2, pp. 30–57.