DroneNetX: Network Reconstruction Through Connectivity Probing and Relay Deployment by Multiple UAVs in Ad Hoc Networks

So-Yeon Park, Christina Suyong Shin, *Member, IEEE*, Dahee Jeong, and HyungJune Lee, *Member, IEEE*

*Abstract***—In this paper, we consider a network reconstruction problem using unmanned aerial vehicles (UAVs) where stationary ad hoc networks are severely damaged in a post-disaster scenario. The main objective of this paper is to repair the network by supplementing aerial wireless links into the isolated ground network using UAVs. Our scheme performs network probing from the air and finds out crucial spots where both local and global routing performance can significantly be recovered if deployed. First, we propose a novel distributed coverage path planning algorithms with independent and computationally lightweight navigation based on adaptive *zigzag* patterns. Second, we present route topology discovery schemes that capture both local and *non*-local network connectivity by extracting inherent route skeletons via stitching partial local paths obtained from the simple packet probing by UAVs. Finally, we find the optimal UAV relay deployment positions that can improve network-wide data delivery most effectively based on three novel approaches of an optimization technique, an iterative heuristic algorithm, and a topology partitioning of *strongly connected component*. Simulation results demonstrate that our distributed traversing algorithms reduce the complete coverage time, the travel distance, and the duplicate coverage compared to other counterpart algorithms. Our deployment algorithms recover severely impaired routes, incurring reasonable computational overhead.**

*Index Terms***—Network reconstruction, route topology discovery, coverage path planning, network hole detection, relay deployment, self-organizing networks, unmanned aerial vehicles (UAVs).**

I. Introduction

Unmanned Aerial Vehicles (UAVs) have been considered as an emerging disruptive technology to facilitate dynamic in-situ operations such as sensing real-time terrestrial events from the air and unmanned package delivery. These UAVs can form their own aerial networks, while also communicating with terrestrial networks [1], [2]. The recent network has been evolving into forming a two-tier network of aerial and terrestrial ad-hoc networks as a promising self-organizing network thanks to the on-the-fly characteristic of UAVs.

In disaster situations, the terrestrial network can be broken into several isolated sub-networks. The network can be even more critically affected if some crucial relay nodes in the middle of networks become lost or in failure. In this situation, maintaining a reliable communication network through fast network repair is important for effectively sharing in-situ emergency information between victims and first responders.

There have been efforts on utilizing autonomous unmanned terrestrial or aerial vehicles on a Region of Interest (RoI) [3]. These mobile vehicles can be used as effective communication resources to quickly reconnect isolated networks each other through their ad-hoc deployment. Employing UAVs can be a rescue to address the network hole problem by being deployed as temporary relay nodes [4], [5].

An advantage of UAVs compared to terrestrial vehicles is its physically less constrained movement for information gathering. The UAVs can retrieve data from the ground via the ground-to-air communication, relay them to other UAVs in the air-to-air communication, and send back to the ground via the air-to-ground communication. They can gather network collapse status with connectivity probing from the air, and also be deployed as communication relays if necessary.

We consider two major roles of UAVs as exploring network connection status over unknown RoI areas, and being deployed as ground-to-air and air-to-ground relays for autonomous network reconstruction. The challenges are 1) to design a distributed motion planning algorithm for sparse yet efficient connectivity probing over the damaged network, and 2) to locate network holes where the deployment of UAV relays helps to repair the damaged network.

There have been previous works to address the problem of multi-agent exploration in [6]–[8] mostly from robotics community. In [7], [8], researchers propose simple distributed *Ants* algorithms simulating a colony exploration of ants while leaving pheromone traces during the environment traversing. Although [6], [9] present the *Brick&Mortar* algorithm and its variation that reduce duplicate coverage as opposed to the *Ants*, they suffer from computationally intensive loop closure problems. Some other works [10], [11] have extensively investigated the problem of path planning and space exploration. Although most of them aim to mitigate duplicate coverage among agents, they can not directly be applied to the UAV context because there is less consideration of networking capability and lightweight computation.

Manuscript received March 9, 2018; revised July 9, 2018; accepted August 25, 2018. Date of publication September 14, 2018; date of current version November 12, 2018. This work was supported in part by the National Research Foundation of Korea Grant funded by the Ministry of Education (NRF-2015R1D1A1A01057902) and in part by the Ministry of Science and ICT (NRF-2018R1A2B6004006). The review of this paper was coordinated by Prof. C. Assi. *(Corresponding author: HyungJune Lee.)*

The authors are with the Department of Computer Science and Engineering, Ewha Womans University, Seoul 03760, South Korea (e-mail: ps.sally25@gmail.com; suyongshin92@gmail.com; wjdekgml0703@naver.com; hyungjune.lee@ewha.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2018.2870397

The problem of network hole detection and deployment has been studied in [12]–[16] from network community. [12], [14], [15] explore sensor deployment algorithms by finding network holes in from more theoretical perspectives. Regarding the usage of aerial vehicles, aerial communication based on 802.11n performs poorly due to aerial link vulnerability [13], while some antenna extension can enhance the quality of aerial links [16]. Some researchers utilize UAVs to re-establish network connectivity with aerial deployment in [17], [18]. However, network repair improvement with respect to network probing density and the optimal UAV deployment problem based on sparse connectivity information have not been investigated well.

In this paper, we aim to answer two key questions of 1) how to traverse a network efficiently for finding the terrestrial network connection status with multiple UAVs in a distributed way without much duplicate coverage and 2) what the optimal UAV deployment algorithm based on tangible connectivity measurements should be to achieve a practical recovery.

We propose several novel distributed path planning algorithms based on independent and computationally light decisions among several pre-determined *zigzag* patterns. These patterns extend the local coverage as the UAVs are flying forward, while reducing duplicate coverage with other UAVs. Exploring the RoI area, the UAVs periodically probe network connectivity from the air toward stationary networks.

We develop connectivity probing algorithms for UAVs to capture both local and *non*-local network connectivity from the air to accomplish a more suitable UAV deployment in terms of route reconstruction. Based on our newly designed cost-effective path planning algorithms, UAVs drop off probing packets that gather local network connection information within one hop, or keep being relayed within several hops and retrieve them to parse their distinctive partial *non*-local paths. We discover the underlying route topology by constructing the collected partial paths via *path stitching*, where we borrow some general idea and term from the wired network [19].

Once the network traversing procedure based on path planning is completed, we find the optimal UAV relay positions that can repair network-wide data delivery most effectively. First, we aim to find the optimal deployment strategy based on an optimization technique. We formulate the problem into a binary integer program and obtain the optimal deployment positions for UAVs. Second, we seek a computationally more efficient iterative deployment strategy. By leveraging the obtained topology information constructed by UAVs, we locate and prioritize network holes by capturing a more global impact on the overall routing structure. To understand the inherent route skeletons, we perform connectivity-based clustering of stationary nodes and UAV deployment candidate spots. We iteratively find the most effective deployment locations, leading to significant route improvement over the damaged network. Third, we devise a further improved UAV deployment algorithm by capturing more global route skeletons based on the topology partitioning technique of *strongly connected component* from an even higher-level perspective.

Our main contributions can be summarized as follows.

- We design several variants of distributed path planning algorithm dedicated to UAVs that greatly reduces travel time and distance, and duplicate coverage among UAVs.
- We present both local and non-local route topology discovery schemes to extract the inherent route skeletons by stitching partial local paths obtained from simple path probing by UAVs.
- We propose practical network hole replacement algorithms that dispatch a limited number of UAVs to the selected crucial spots, which can achieve both local and global improvement of routing performance.

This paper proposes a network reconstruction framework, called *DroneNetX*, which is a class of network traversing and relay deployment algorithms using multiple UAVs. This work extends our prior work of *DroneNet* [20] and *DroneNet+* [21] as follows.

- We present a further improved network traversing algorithm that reduces the network traversing time down to less than 11% by either adaptively incrementing or decrementing the traversal width of UAVs.
- We propose a new network hole replacement algorithm that captures globally connected sub-network dynamics based on the topology partitioning technique of strongly connected component, improving network recovery performance in terms of end-to-end routing cost and performance stability.
- We add experimental results on how the traversal width should be adjusted within a time windowing manner.
- We add experimental results to compare all the proposed network traversing algorithms including the new further improved algorithm in terms of navigation efficiency.
- We add experimental results to compare all the proposed UAV deployment algorithms including our new algorithm in terms of routing recovery performance.
- We add discussions on the relationship between the number of UAVs and the network collapse degree, the effects of network traversing and deployment decision on overall network recovery performance, the possibility of integrating both traversing and deployment into one stage and employing its progressive optimization, and several practical issues for real-world applications.

The remainder of this paper is organized as follows. After discussing related work in Sec. II, we introduce our problem and system model in Sec. III. We present several algorithms of network traversing consisting of route topology discovery and motion planning in Sec. IV, and Sec. V describes our UAV deployment algorithms. After presenting the evaluation results of our proposed approaches in Sec. VI, we discuss several crucial aspects in Sec. VII, and then finally conclude this paper in Sec. VIII.

II. Related Work

Related work on the problem of network reconstruction using UAVs is classified into two categories: area exploration of multi-agents (or multi-robots) and network hole replacement.

A. Multi-Agent Area Exploration

The area exploration of multi-agents has extensively been investigated in robotics research community with a long history. This problem is also referred to as *coverage path planning*. To minimize the completion time for a certain area, an efficient path or trajectory that can explore all the searching areas is extracted. To achieve a practical yet efficient trajectory, the area space is decomposed into multiple cells with approximate [22], [23], semi-approximate [24], [25], and exact types [25], [26]. The cellular decomposition facilitates practical construction of provable performance guarantee [27].

Some algorithms have been inspired by swarm intelligence for cooperative foraging behaviors of ants or bees. [7], [8] propose decentralized *Ants* algorithms simulating a colony exploration of ants while leaving pheromone traces during the environment traversing. While these *Ants*-based algorithms are not optimized enough in terms of reducing duplicate coverage, [6], [9] present the *Brick&Mortar* algorithm and its variation to tackle this problem. However, they suffer from somewhat computationally intensive loop closure problems. Also, in this type of algorithms, each agent shares its collected information by tagging the environment, e.g., leaving some information (e.g., pheromone) into a certain area, for other agents to use it upon visiting later, limiting applicability to the UAV context.

In the context of UAV usage, there have been some recent studies that design the optimal path-planning using a UAV. A recent work [28] aims to solve a traveling salesman problem for environmental monitoring based on a genetic algorithm that calculates fitness functions for maximizing the coverage. Furthermore, many interesting research has been conducted for deriving effective motion planning for multiple UAVs [24], [29]–[31]. These works can be categorized into three dimensional coverage path planning and task division among multiple UAVs. To obtain coverage methods in 3-dimensional space, some works [24], [31] apply a planar coverage algorithm based on 2-dimension in successive horizontal planes. In [32], [33], 3-dimensional cellular decomposition has been conducted for constructing the 3D coverage path of UAVs. Using multiple UAVs, coverage can be explored in a cooperative fashion through task division by maintaining explore states for multirobot coordination [29] or based on a decentralized gradient-based probabilistic search with cooperative UAVs [30].

Although our work belongs to the cellular decomposition approach with the exact type using zigzag trajectory, we differentiate our work from previous works in the following aspects. Our work adaptively changes the UAV traversal pattern by reflecting the ongoing traversal progress and the encountering events with other UAVs for minimizing the duplicate coverage over the RoI. Further, our scheme performs network topology construction in a more active way via connectivity probing through network probing packets as well as through UAVs' direct visit over terrestrial ad-hoc nodes, concurrently with the above coverage path planning.

B. Network Hole Replacement

The network hole replacement has been explored in ad-hoc sensor networks research community [34], [35]. Lacking wireless coverage in a certain target area, sensing capability over the area or local communication among sensors become undermined. This *coverage hole* problem can be addressed by discovering the existence of coverage holes based on Voronoi diagrams [36], by detecting virtual forces in the potential field among nodes considered as virtual particles [37], by comparing local density with the ideal uniformly distributed density [38], or by maintaining the line of sight communication relationships among nodes [39]. Mobile sensors are dispatched or change their deployment location to the detected coverage hole location.

There have been several efforts to address the network hole problem by utilizing UAVs as relay nodes [3], [17], [18], [40]–[42]. UAVs are dispatched to some critically damaged spots and are served as bridge nodes to connect a terrestrial sub-network with another from the air. They aim to recover the broken connectivity by utilizing UAVs based on Delaunay triangulation [17] or a game theoretic approach [18].

The authors in [3] propose three phase deployment strategy: initial deployment, connectivity measurement on the initially deployed topology, and thereafter deployment position repair. In [40], two types of network connectivity: global message connectivity and worst-case connectivity are quantified, and both connectivity quantification aims to be maximized based on the concept of the minimal Steiner tree from graph theory. Similarly in [41], the 3D locations of the UAVs are determined to maximize the coverage lifetime using circle packing theory. In the similar context, some research work [43] proposes a relay node placement algorithm based on particle swarm optimization that takes considerable computation complexity. The authors in [42] solve the network coverage problem by finding out the optimal deployment positions of UAVs via a fitness function with three metrics of coverage, fault-tolerance, and redundancy, based on a theoretical unit disk model under the known node locations.

Our work presents network hole finding algorithms by understanding both local and non-local network topology in the connectivity space so that it can repair broken packet routes via UAV relays and recover routing performance in a local and non-local manner.

III. System Model

This work considers a network reconstruction problem using UAVs where stationary ad-hoc networks are severely damaged in a post-disaster scenario. Our goal is to repair network coverage by supplementing aerial wireless links into the stationary network to reconnect isolated ground networks each other with a limited number of UAVs.

We assume that UAVs are equipped with the same wireless radio as stationary nodes (e.g., 802.11 or 802.15.4). A UAV can communicate with a part of stationary nodes on the ground or other UAVs in the air as long as they are within radio range. It is also assumed that UAVs are aware of RoI to explore and can keep track of their relative position on RoI compared to their corresponding physical position. Any UAV control issues on moving from one location to another due to external environmental factors such as weather, obstacles, and collisions with other UAVs are out-of-scope in this paper. We consider UAVs to initially be fully charged and keep operating without recharging during a complete mission of network reconstruction.

The problem of network construction using UAVs can be divided into two sub-problems: 1) network connectivity probing from the air based on distributed motion planning of UAVs for the complete RoI coverage, while reducing duplicate coverage (refer to Sec. IV), and 2) optimal UAV relay deployment for

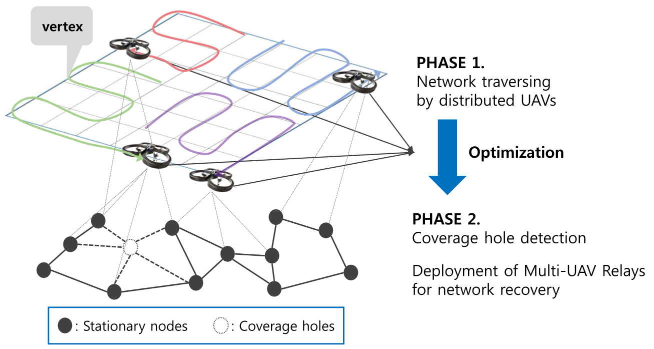


Fig. 1. Overall procedure of network traversing, coverage hole detection, and deployment by exploiting UAVs for autonomous network recovery.

the most effective network recovery given a limited number of UAVs (refer to Sec. V).

After all of UAVs finish network exploration over a given Region of Interest (RoI), they gather at a designated place to share the collected network probing information. Based on the information, their deployment location can be computed at a selected UAV or a group of UAVs. Its deployment computation result is shared with other UAVs, and they are accordingly dispatched at each position as relays, as illustrated in Fig. 1.

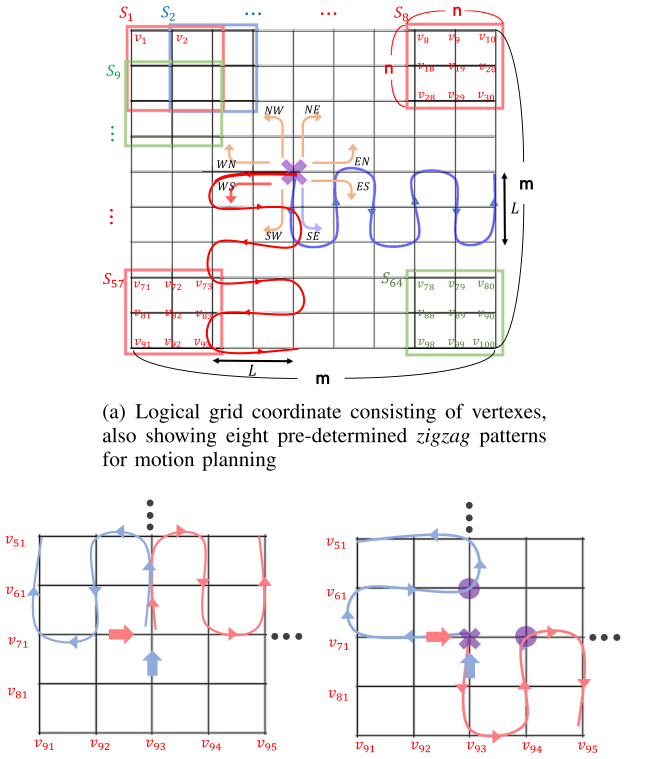
IV. Network Traversing

When a catastrophic disaster occurs, a terrestrial network of stationary ad-hoc nodes may be damaged severely. To maintain reliable routes over stationary ad-hoc networks, it is important to quickly navigate the damaged area for diagnosing network connection status before additional relay nodes are deployed.

We propose network traversing algorithms consisting of distributed motion planning for multiple UAVs and concurrent network probing by them. Multiple UAVs explore the network over RoI according to their own independent navigation decision. For an efficient distributed exploration on the RoI region, we define a frontier map that consists of square grids with $m \times m$ vertexes as in Fig. 2(a). Each UAV initiates its navigation at its currently visiting vertex or a designated vertex, continues its movement decision to the next vertex, and stops if it covers all of the vertexes on the RoI.

A. *DroneNet: Network Traversing*

Each UAV runs an independent motion planning based on one of eight pre-defined *zigzag* patterns, e.g., North-East, North-West, South-East, South-West, East-North, East-South, West-North, and West-South with the orthogonal traversal width L. It generates a *future-vertex-visit-trajectory* with the longest length toward a certain direction up to the boundary of RoI among the above eight pre-determined *zigzag* patterns as shown in Fig. 2(a). Since a UAV is not aware of the total number of UAVs and the location of other UAVs, it is initially supposed to traverse over all the vertexes in RoI. Whenever a UAV visits a vertex at a time, it adds the visited vertex ID to its *vertex-visit-list* and updates its unvisited vertex list. If two or more UAVs are within radio range, they share their own vertex-visit-list with others, and merge them into its original vertex-visit-list and accordingly update their own unvisited vertex list.



(a) Logical grid coordinate consisting of vertexes, also showing eight pre-determined *zigzag* patterns for motion planning

(b) Case 1: The East vertex toward the first quadrant with the longest length up to RoI is taken. Then, the pattern including the North vertex is chosen.

(c) Case 2: Both North and East vertexes toward the first quadrant with the longest length up to RoI are taken. Then, it finds a next longest pattern on another quadrant.

Fig. 2. Logical grid coordinate, *zigzag* movement trajectories, and future vertex visit decision rules in *DroneNet*.

When a UAV decides its next visiting vertex based on the future-vertex-visit-trajectory, it checks whether the anticipating visiting vertex has already been taken by other UAVs by searching it over the vertex-visit-list. In case that the anticipating vertex is already taken, the UAV lists up all available neighboring vertexes to move among (North, East, South, West), except the direction with the taken vertex, and randomly chooses one direction for next move. In this way, a UAV is able to avoid duplicate exploration over the vertexes already visited by other UAVs in a distributed manner. It continues to generate a future-vertex-visit-trajectory with the longest length toward the boundary of RoI and execute its local visit decision afterwards as illustrated in Figs. 2(b) and 2(c).

During each vertex visit, a UAV probes network connectivity with neighboring stationary nodes near the vertex. The UAV broadcasts hello packets with a periodic manner, and any neighboring stationary nodes that have received a hello packet replies back to the UAV with a response packet embedding its own node ID. Based on the collected response packets from connectable nodes for multiple hello packets, the UAV calculates the average Packet Reception Rate (PRR) for each responded node ID at the vertex position. As each UAV traverses over the network on the RoI, it continuously updates its PRR table for the attributes of visited vertex ID and stationary node ID and also exchanges its PRR table together with the vertex-visit-list if other UAVs are within radio range.

Algorithm 1: Distributed Multi-UAV Network Traversing in *DroneNet*.

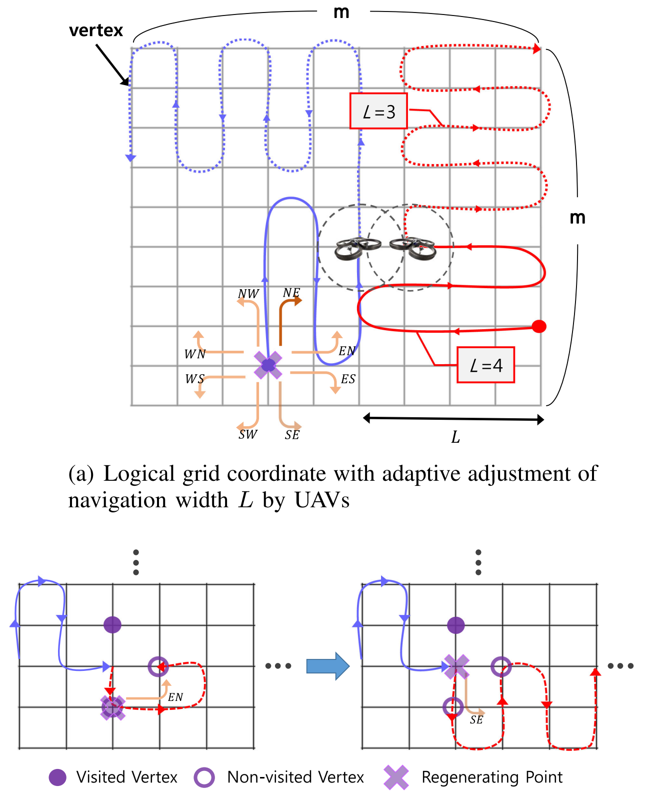
```
Require: CurrentVertexID
Ensure: NextVertexID
// Part I: Motion planning
if (future-vertex-visit-trajectory == ∅) then
    Regenerate the future-vertex-visit-trajectory with the longest length
    that starts from an unvisited vertex;
    NextVertexID = future-vertex-visit-trajectory's first vertex ID;
    Move with one step to the next vertex;
else
    if (future-vertex-visit-trajectory's next vertex is taken or null) then
        future-vertex-visit-trajectory = ∅;
        if (any unvisited neighboring vertex in North, East, South, West)
        then
            NextVertexID = random-pick(unvisited neighboring vertexes);
            Invoke connectivity-probing();
            Move with one step to the next vertex;
        else
            if (there exists any unvisited vertex) then
                NextVertexID = the nearest vertex's ID on the grid coordinate
                from CurrentVertexID;
                Invoke connectivity-probing();
                Fly to the next vertex;
            else
                Terminate;
            end if
        end if
    else
        NextVertexID = future-vertex-visit-trajectory's next vertex ID;
        Invoke connectivity-probing();
        Move with one step to the next vertex;
    end if
end if
// Part II: Connectivity probing
Function connectivity-probing()
    Broadcast hello packets;
    Receive response packets from neighboring stationary nodes;
    Calculate the average PRR for each responded stationary node;
    Update the PRR table for CurrentVertexID and StationaryNodeID;
    if (any UAVs within radio range) then
        Exchange vertex-visit-list and PRR table, and update them;
    end if
EndFunction
```

When all of neighboring vertexes in the north, east, south, and west directions are taken, a UAV compares its vertex-visit-list with the entire vertex list on RoI, and selects an unvisited vertex with the shortest distance on the grid coordinate for its next move. In this case, the UAV directly flies to the selected vertex. If there remains no vertex to visit, it finishes the network traversing procedure.

Our network traversing algorithm guarantees the complete coverage of vertexes with distributed motion planning of multiple UAVs and its successful termination without overlapping loops. The proofs are straightforward and omitted due to space constraints. A more detailed algorithm is described in Algorithm 1.

B. *DroneNet+: Network Traversing*

To maintain reliable route paths over stationary ad-hoc networks, preserving local wireless connectivity to neighboring nodes is essential. Although local connectivity is a good indicator of quantifying local route path stability, it does not necessarily embed the global routing structure within itself. Thus, it is important to diagnose the route status on the damaged network by discovering its global route topology beyond the local one.



(a) Logical grid coordinate with adaptive adjustment of navigation width L by UAVs

(b) More optimized navigation decision, leading to a longer traverse before the next decision

Fig. 3. Adaptive UAV traversing with an effective navigation decision in *DroneNet+*.

In this subsection, we propose a route topology discovery scheme that extracts the inherent route skeletons using simple packet probing by UAVs. Our route topology discovery consists of two phases: network traversing and topology construction via *path stitching*.

1) Adaptive UAV Path Planning: DroneNet as in Sec. IV-A has a drawback of using the fixed L regardless of on-going navigation progress with other UAVs. Now *DroneNet+* adaptively controls the orthogonal width L of the UAV navigation. Initiating its zigzag trajectory with the given L, the UAV can have a more chance to navigate its adjacent vertexes before moving away toward its determined moving direction. As the RoI has been explored further together with other UAVs, it may find any duplicate visited vertex from the exchanged *vertex-visit-list* with another UAV within radio range. In this case, it decrements the navigation width L by one so that it can lessen potential duplicate coverage on its future movement progress as depicted in Fig. 3(a). Also, we devise the future vertex trajectory decision rule of *DroneNet* that just randomly selects a direction with a non-visited vertex as its next move and then generates its future trajectory at the selected vertex after moving to it (shown at the left in Fig. 3(b)). In *DroneNet+*, instead, each UAV regenerates its future trajectory considering nearby non-visited vertexes as soon as it completes the traversal from its previously generated trajectory (shown at the right in Fig. 3(b)).

This adaptive UAV traversing scheme *DroneNet+* greatly advances the previous *DroneNet* in motion planning efficiency

Algorithm 2: Adaptive Multi-UAV Network Traversing in *DroneNet+*.

```
Require: CurrentVertexID
Ensure: NextVertexID
// Part I: Motion planning
if (future-vertex-visit-trajectory == $\emptyset$) or (future-vertex-visit-trajectory's next vertex is taken or null) then
    if (any unvisited neighboring vertex in North, East, South, West) then
        Regenerate the future-vertex-visit-trajectory with the longest length that starts from an unvisited vertex;
        NextVertexID = future-vertex-visit-trajectory's first vertex ID;
        Move with one step to the next vertex;
    else
        if (there exists any unvisited vertex) then
            NextVertexID = the nearest vertex's ID on the grid coordinate from CurrentVertexID;
            Invoke path-probing();
            Fly to the next vertex;
        else
            Terminate;
        end if
    end if
else
    NextVertexID = future-vertex-visit-trajectory's next vertex ID;
    Invoke path-probing();
    Move with one step to the next vertex;
end if
// Part II: Path probing
Function path-probing()
    Broadcast a path-probing packet;
    Any neighboring stationary nodes keep relaying the probing packet up to $n$ hops, while recording a series of relay node IDs in header;
    Receive completed path-probing packets stored at the currently visiting node initiated by itself or other UAVs;
    if (any UAVs within radio range) then
        Exchange vertex-visit-list and update it;
        if (any duplicated visited vertexes) then
            Decrement the navigation width $L$ by 1;
            future-vertex-visit-trajectory = $\emptyset$;
        end if
    end if
EndFunction
```

by lowering duplicate coverage and travel distance until all the vertexes are completely covered by UAVs.

2) Topology Construction via Path Stitching: We let UAVs diagnose the overall network status during traversing by relaying path-probing packets. We construct a global route aggregate by stitching partial local route paths obtained from the path-probing packets via *path stitching*.

a) Probing partial local path via relaying: When a UAV visits a vertex during network traversing, it broadcasts a path-probing packet to its stationary neighbors within radio range. The stationary nodes are designed to relay it up to only n hops by recording their own node ID and the current number of transmission hops in its header. The path-probing packet finishes being relayed to a certain stationary node upon completing n hop transmission, and the recorded path-probing information is stored at the node. This probing information is collected later by a visiting UAV.

b) Topology Discovery via Path Stitching in Off-Line: Once the network traversing procedure is completed, all of the collected local path information by multiple UAVs are used to extract a global route topology in off-line. Based on the path trace information, we construct an undirected graph topology.

Given the local route path information ubiquitously collected by UAVs, we finally construct a global route topology by stitching all the links together as in Fig. 4. A more detailed algorithm is described in Algorithm 2.

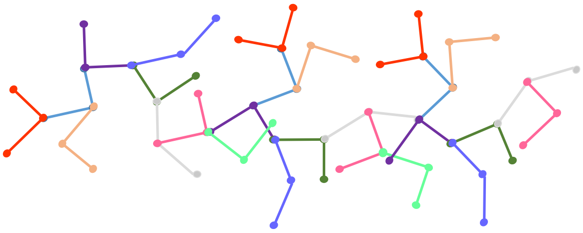


Fig. 4. Topology discovery by stitching partial local route paths (over two hops) via *path stitching*.

C. Adv-DroneNet+: Advanced Network Traversing

In the prior network traversing in *DroneNet+*, each UAV adjusts the orthogonal navigation width L by decrementing by one upon encountered with another UAV, helping to reduce the possibility of duplicate coverage around the nearby area in the near future. However, once the navigation width L is decremented, it stays with the current value or can only be further decremented, never being incremented again. In case that two UAVs are encountered, and their navigation widths are once reduced at the beginning of network traversing stage, even after each UAV has never been encountered with any UAV for quite a long time, their navigation widths still keep the same with the *penalized* value forever.

Our advanced network traversing algorithm addresses this issue by allowing a time window W, which is an expiration time interval for staying with the current navigation width. If a UAV has not been encountered with another UAV during this time window, its navigation width L is incremented by one so that it is allowed to explore the nearby area a little bit more widely before moving away toward the navigation direction.

At the beginning of network traversing, each UAV initializes its own time window W to a constant value, *initialWindow*, and decrements W by one for every vertex visit unless encountered with another UAV. Otherwise, the time window W is re-initialized to *initialWindow*, while the navigation width L is decremented by one. This advanced network traversing algorithm is described in Algorithm 3.

V. UAV Relay Deployment

In this section, we present several UAV relay deployment algorithms that find the best grid positions of multiple UAVs for the optimal network repair. Given the collected local connectivity information over RoI, we find critical network holes that drastically undermine network-wide routing performance. We want to deploy a limited number of available UAVs as relays into the locations where local and non-local connection as well as end-to-end routing can significantly be improved.

A. DroneNet: UAV Relay Deployment

Once the UAVs complete the network traversing procedure in Sec. IV-A, we obtain the connectivity table consisting of Packet Reception Ratio (PRR) from stationary node ID k at vertex ID

Algorithm 3: Fully Adaptive Multi-UAV Network Traversing in *Adv-DroneNet+*.

```
Require: CurrentVertexID
Ensure: NextVertexID
// Part I: Motion planning
if ($W \le 0$) then
    $W$ = initialWindow;
    Increment the navigation width $L$ by 1;
end if
if (future-vertex-visit-trajectory == $\emptyset$) or (future-vertex-visit-trajectory's
  next vertex is taken or null) then
    if (any unvisited neighboring vertex in North, East, South, West) then
        Regenerate the future-vertex-visit-trajectory with the longest length
          that starts from an unvisited vertex;
        NextVertexID = future-vertex-visit-trajectory's first vertex ID;
        Move with one step to the next vertex;
    else
        if (there exists any unvisited vertex) then
            NextVertexID = the nearest vertex's ID on the grid coordinate
              from CurrentVertexID;
            Invoke path-probing();
            Fly to the next vertex;
        else
            Terminate;
        end if
    end if
else
    NextVertexID = future-vertex-visit-trajectory's next vertex ID;
    Invoke path-probing();
    Move with one step to the next vertex;
    Decrement the time window $W$ by 1;
end if
// Part II: Path probing
Function path-probing()
    Broadcast a path-probing packet;
    Any neighboring stationary nodes keep relaying the probing packet
      up to $n$ hops, while recording a series of relay node IDs in header;
    Receive completed path-probing packets stored at the currently
      visiting node initiated by itself or other UAVs;
    if (any UAVs within radio range) then
        Exchange vertex-visit-list and update it;
        if (any duplicated visited vertexes) then
            $W$ = initialWindow;
            Decrement the navigation width $L$ by 1;
            future-vertex-visit-trajectory = $\emptyset$;
        end if
    end if
EndFunction
```

j, i.e., $[\mathtt{PRR}]_{j,k}$ where $1 \le j \le M(= m^2)$ and $1 \le k \le N$. To find the network holes, we observe a set of vertexes that retain the weakest wireless links to the neighboring stationary nodes. Since the number of UAVs is limited, we prioritize the network holes and select some of them as UAV deployment positions. It should be noted that UAVs should not be deployed into the network holes completely isolated by any neighboring stationary nodes because the deployed relay still remains unconnected to any of them.

To benefit the overall network from only few UAV relays for network reconstruction, we aim to minimize duplicate network coverage by prohibiting two or more UAVs from being deployed within communication range. Thus, we want each UAV to contribute to repairing its nearby network connectivity without partial or complete duplicate coverage for the overall network repair enhancement.

We formulate the problem of selecting grid positions of multiple UAVs for network repair into a binary integer program. Our goal is to find a set of vertex regions that have the weakest non-zero PRRs averaged over neighboring stationary nodes, while avoiding duplicate coverage with any of other UAVs.

To formulate this setting, we first define a group of vertexes on a square sub-grid within the average radio range of a wireless interface as $S_i = \{v_{i_1}, v_{i_2}, v_{i_3}, \ldots, v_{i_{n^2}}\}$ where v_{i_l} $(1 \le l \le n^2)$ is a vertex element belonging to the set S_i, and n^2 is the total number of elements in set S_i, and $S_1 \cup S_2 \cup \cdots \cup S_K = \{v_1, v_2, \ldots, v_{m^2}\}$ as in Fig. 2(a). Given the P number of UAVs to deploy, the problem of selecting P grid positions of UAVs is to select the P number of sets with the lowest average PRRs over their corresponding belonging vertexes among $S_1, S_2, \ldots,$ and S_K, while any selected vertex sets should not share any vertex in common. Both S_i and S_j cannot be selected if $S_i \cap S_j \neq \emptyset$. For example, in Fig. 2(a), both S_1 and S_2 cannot be selected as deployment vertexes. This implies that we want to deploy a UAV into a group location of vertexes of which most or all suffer from similarly poor connection.

We introduce indicator functions J_i denoting the vertex group set S_i should be selected, and $I_{i,j}$ denoting whether the vertex group set S_i and its belonging vertex v_j should be selected. Based on these notations, we define the objective function to minimize the summation of the average PRRs of the selected vertexes in the selected vertex set as follows:

$$\text{minimize} \quad \sum_{i,j \in S_i} \overline{PRR}_j \cdot I_{i,j} \tag{1}$$

$$\text{subject to} \quad \sum_i I_{i,j} \le 1 \qquad \forall j \tag{2}$$

$$J_i = I_{i,i_1} = I_{i,i_2} = I_{i,i_3} = \cdots = I_{i,i_{n^2}} \qquad \forall i \tag{3}$$

$$\sum_i J_i = P \tag{4}$$

where $\overline{PRR}_j$ is the average PRR over only the stationary nodes with non-zero PRRs at vertex v_j. In case that vertex v_j has no connection at all, i.e., $\overline{PRR}_j = 0$, we force it to be 1 so that isolated vertexes should never be selected.

Constraint (2) ensures that any selected vertex sets should not share any vertex in common to avoid duplicate coverage. Constraint (3) enforces the condition that once a vertex group set S_i is selected, any belonging vertex $v_j \in S_i$ should be selected. The last constraint (4) requires the total number of selected vertex group sets to be the same number of UAVs.

By using MATLAB bintprog utility or AMPL/CPLEX solver, we can obtain the optimal sets of the most vulnerable vertex groups under critical link outage. Since each UAV ends up with the entire connectivity table for all the vertexes at the end of network traversing procedure, it calculates them for itself. Once each UAV tracks down to these sets, it determines one of sets according to the order of UAV ID, and flies directly to the center position of the selected vertex group for its self-deployment. These positions are exactly where the deployed UAVs can be used as crucial relay resources for starting repairing the locally broken network.

B. DroneNet+: UAV Relay Deployment

We present a more computationally-efficient iterative UAV deployment algorithm that improves routing performance through a heuristic approach from a higher-level routing perspective. We locate and prioritize network holes based on the captured route topology in Sec. IV-B and deploy UAVs as relays to connect with terrestrial networks.

To understand the inherent global routing structure over the networks, it is necessary to find out crucial *skeleton* nodes that connect not only local neighboring nodes but rather other neighboring sub-networks. To extract those skeleton nodes in the networks, we incorporate a connectivity-based clustering algorithm and use the selected cluster heads to efficiently connect via inter-cluster networking.

Once the skeleton nodes are obtained, we associate two-tier networks: a network of real nodes (i.e., cluster heads) and the other network of virtual nodes (i.e., vertexes, which are candidate spots for UAV deployment). To gain the knowledge of parts of vertexes with high connectivity toward real skeleton nodes, we precompute a measure of how much the overall routing performance can be improved when comparing between *before* and *after* UAVs are deployed at the vertexes.

After prioritizing those vertexes, we dispatch UAVs to the most effective vertexes that lead to the most influential network repair in terms of routing performance. We iteratively find vertexes, deploy UAVs to their locations, and perform this iteration continuously until all the UAVs are dispatched.

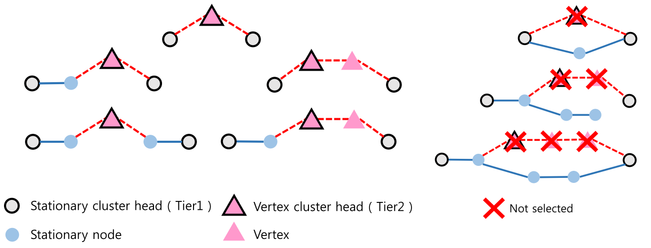
1) Connectivity-Based k-hop Clustering: We present a simple yet efficient connectivity-based k-hop clustering method performed in a centralized manner similar to [44], [45], which does not require any node location and better reflects empirical wireless connectivity behaviors. Using a constructed route topology, we count the number of connected neighboring nodes for each node within k hops and then prioritize the node list in the descending order.

We initially elect a node with the highest connectivity as the first cluster head. Given this cluster head, all of the nodes within k hops from the cluster head join this cluster as cluster members. Once a cluster head and its belonging members are determined, we exclude these nodes from the above node list. We continue this procedure for the remaining nodes in the list until all the nodes are traversed. If there are a few nodes with the same number of neighbors in the cluster head selection, we randomly pick up one node among them as the next cluster head. It should be noted that a cluster head without any member is prohibited, and thus, there can exist some single nodes that do not belong to any cluster.

Our connectivity-based clustering approach enables to understand high-level route establishment over the entire networks through several cluster heads used as *skeleton* nodes.

2) Network Hole Replacement With UAV Relays: Our network hole replacement algorithm consists of two phases: multi-level clustering and deployment. First, we perform a connectivity-based k-hop clustering for all of stationary nodes based on the obtained route topology. This captures high-level skeleton nodes that serve an important role to connect with even farther nodes.

Once cluster heads are elected, we perform additional clustering only for cluster heads (that are real nodes), and vertexes (that are virtual nodes considered as candidate places where UAVs can be deployed). Then, we obtain vertex cluster heads and their belonging members. This second-tier clustering offers an informative high-level connectivity structure of how virtual nodes located at vertex positions can deeply be associated with cluster heads, *skeleton nodes* in real networks. We select the most influential vertex positions with the highest impact on route connectivity with skeleton nodes as the deployment positions of UAVs.


Stationary cluster head (Tier1)
Vertex cluster head (Tier2)
Not selected
Stationary node
Vertex

(a) Possible vertex cases to connect two cluster heads at each end within two hops (2nd Case)

(b) Validation check with no better route

Fig. 5. Iterative clustering and deployment decision procedure in *DroneNet+*.

We consider three *deployment cases* for UAVs: 1) a vertex that can connect one cluster head at the one end with another at the other end within one hop, 2) a vertex to connect two cluster heads at each end within two hops, and 3) a vertex to connect two cluster heads at each end within three hops, as the second case is depicted in Fig. 5(a). If two cluster heads have any existing paths with the lower number of hops away *not through* the vertexes within the designated number of hops for each case, we no longer consider these vertexes as deployment candidates as illustrated in Fig. 5(b). This is due to the fact that the deployment of UAVs at those positions are definitely not a desirable choice compared to otherwise scenarios with two cluster heads connectable only through the vertexes.

Our network hole replacement algorithm iteratively tries to deploy all possible UAVs to the most effective vertexes. To evaluate the *route effectiveness* of UAV deployment at certain vertex locations, we probe routing performance improvement in case of deploying UAVs at vertex candidates. We calculate the percentage of source-to-destination pairs with no existing path among all possible within λ hops from the vertex for both *before*-deployment and *after*-deployment scenarios. As the percentage difference between *before* and *after* increases, it is reasonable to say that the effectiveness of UAV deployment increases. We prioritize all possible vertex candidates for UAVs to be deployed in the descending order of this effectiveness measure. UAVs are consequently deployed to the vertexes with the highest route effectiveness.

In case that all of UAVs are not deployed at this stage yet, we continue the above multi-level clustering and deployment procedures by extending to the second deployment case, and doing so up to the third deployment case. It should be noted that once some UAVs are deployed at the selected vertexes, we treat the UAVs as normal stationary nodes at the remaining clustering and deployment procedures.

Even after executing over all three deployment cases, there can still be remaining UAVs to be deployed yet. We prioritize the remaining vertexes in the descending order according to the number of neighbors within k hops after deploying all possible UAVs at the prior steps, and eventually deploy all the remaining UAVs to them.

Algorithm 4: Deployment of UAV Relays at Network Holes in *DroneNet+*.

```
Require: Route topology & # of available UAVs for relay deployment
Ensure: selectedVertexesForUAVs
selectedVertexesForUAVs = $\emptyset$;
// Deployment case 1 to 3: multi-level clustering & deployment
deploymentCase = 1;
while(deploymentCase <= 3)
    // 1st-tier clustering for all stationary nodes using $k$ hops
    // return stationary cluster heads
    sClusterHeads = connectivity-clustering(stationaryNodes);
    // 2nd-tier clustering for stationary cluster heads and all vertexes
    // using deploymentCase hops
    // return vertex cluster heads
    vClusterHeads = connectivity-clustering(sClusterHeads $\cup$ vertexes,
                                            deploymentCase hops);
    Find parts of vClusterHeads connecting two sClusterHeads through;
    Exclude ones with any existing route within deploymentCase hops;
    Prioritize all possible vertex candidates in terms of route effectiveness;
    Deploy all possible UAVs to the prioritized vertexes;
    Update selectedVertexsForUAVs with them;
    vertexes $\leftarrow$ vertexes – selectedVertexesForUAVs;
    stationaryNodes $\leftarrow$ stationaryNodes $\cup$ selectedVertexesForUAVs;
    deploymentCase++;
endwhile
if (any UAVs still left) then
    Deploy remaining UAVs to vertexes with the highest # of neighbors
    within $k$ hops;
    Update selectedVertexesForUAVs with them;
end if
```

This iterative algorithm provides a lightweight yet effective deployment decision for multiple UAVs, contributing to significant improvement in routing performance. A more detailed algorithm is described in Algorithm 4.

C. *Adv-DroneNet+: Advanced UAV Relay Deployment*

Although *DroneNet+* captures the inherent route skeletons among stationary cluster heads and focuses on a new route establishment between two cluster heads for network recovery, it may neglect some more effective route establishment between stationary nodes that are not classified as stationary cluster heads, near the boundary of interfacing clusters.

In this subsection, we propose a new network hole replacement algorithm that captures globally connected sub-network dynamics based on the concept of *strongly connected component* [46], [47] in graph theory. A strongly connected component is a graph structure where every node is reachable from every other node through a valid route.

We first construct a directed graph based on the constructed route topology from Sec. IV-B. Then, we partition the directed graph into *strongly connected components*. Our goal is to connect these isolated strongly connected components each other by deploying available UAV relays. Given the limited number of UAVs for deployment, we prioritize all possible combination pairs of two components among the extracted strongly connected components, which can directly be connectable if one UAV relay is deployed at a certain vertex.

To prioritize multiple candidate pairs, we count the total number of nodes that belong to two components if they are reachable via a newly deployed UAV . After finding out one pair of two reachable components, there may exist several vertexes where a deployed UAV can relay between these two components. To select the most effective vertex among them, we calculate the percentage of source-to-destination pairs with no valid route among all the stationary nodes for *before*-deployment and *after*-deployment at each vertex candidate. A vertex with the largest improvement is selected as the deployment position for the first UAV. We iteratively continue to perform the above deployment. After all possible component-to-component-wise connections are repaired, if there still exist deployable UAVs, one additional UAV is allowed to be deployed between already-connected two components in the order of component pair priority for fault-tolerance. A more detailed procedure is described in Algorithm 5.

Algorithm 5: Deployment of UAV Relays at Network Holes in *Adv-DroneNet+*.

```
Require: Route topology & # of available UAVs for relay deployment
Ensure: selectedVertexesForUAVs
Construct a directed graph from Route topology;
Partition the directed graph into strongly connected component;
Calculate a priority metric of each component, $p$, as the number of nodes
belonging to the component;
pairList = all possible pairs of two components that are reachable via one
vertex;
Sort pairList by the descending order of the summation of each component
priority, i.e., the total number of nodes within them;
index $i$ = 1;    flag $break$ = false;
while (# of selectedVertexesForUAVs < # of available UAVs)
    if ($i >$ the last index of $pairCandidate$) then
        if ($break$) then
            Break from the loop;
        else
            $i$ = 1;
            break = true;
    end if
    pairCandidate = the $i$th priority pair from pairList;
    vertexCandidate = all possible vertexes that can connect
    pairCandidate each other;
    Calculate the percentage of source-to-destination pairs with no valid
    route among all stationary nodes of before-deployment and
    after-deployment cases for all vertexCandidate;
    selectedVertexForUAVs $\leftarrow$ selectedVertexForUAVs $\cup$ Vertex that has
    the largest improvement between before and after;
    $i$++;
endwhile
if (any UAVs still left) then
    Deploy remaining UAVs to vertexes with the highest # of neighbors
    within $k$ hops;
    Update selectedVertexesForUAVs with them;
end if
```

VI. Evaluation

We evaluate three versions of our route recovery algorithm, *DroneNet*, *DroneNet+*, and *Adv-DroneNet+* in a network of 64 stationary nodes over the RoI of 144 × 144 m^2 as in Fig. 6. We simulate a damaged network consisting of almost half ($\simeq 53.8\%$) broken source-to-destination pairs with no route out of all possible pairs in TinyOS 2.1.2 TOSSIM environment. To model the radio propagation, a combined path-loss shadowing model with a path-loss exponent of 3.3, a shadowing standard deviation of 5.5 dB, a reference distance of 1 m, a power decay of 52.1 dB, a radio noise floor of -104 dBm, a high asymmetric link model, and a white Gaussian noise of 4 dB in TOSSIM `LinkLayerModel` are used. To reflect a more realistic interference environment, we incorporate the CPM interference model [48] with `meyer-light` noise traces.

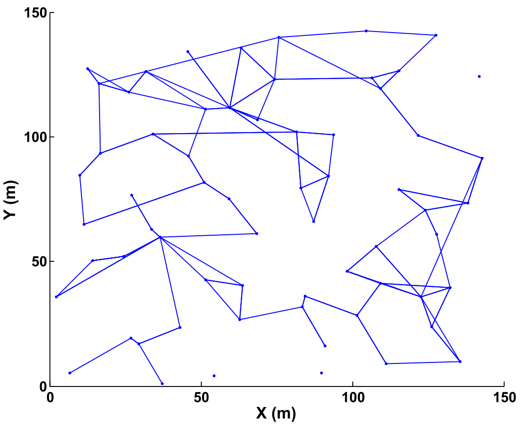
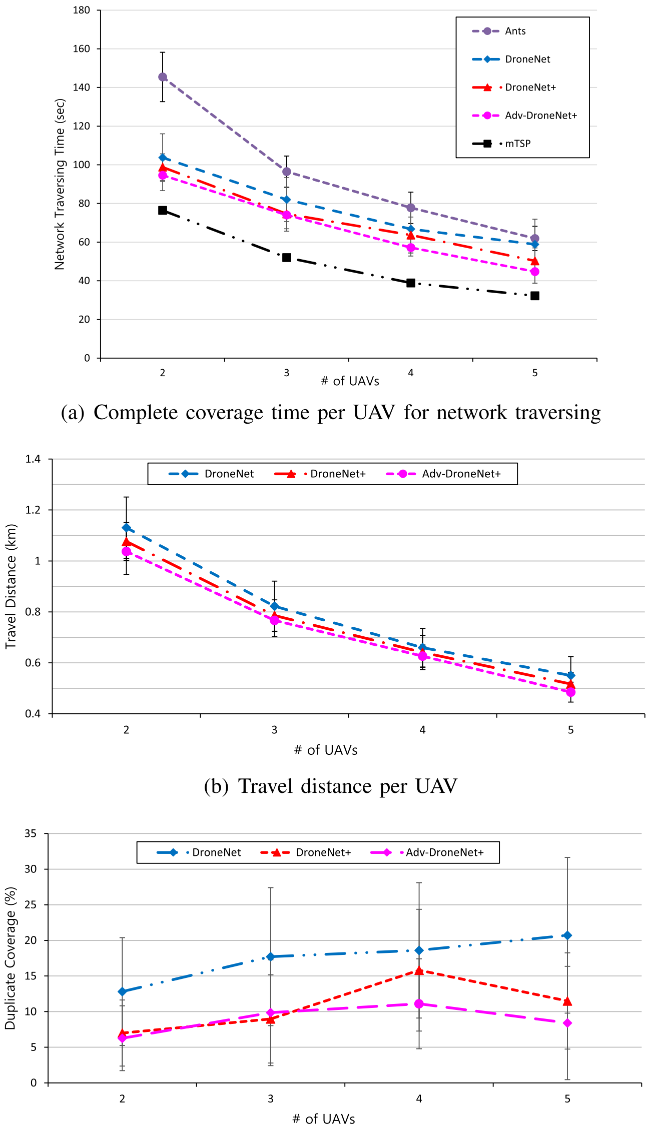


Fig. 6. Network topology of 64 sensor nodes over RoI in a simulated network, having almost half source-to-destination route pairs with no existing path (where good communication links are shown for PRR $\geq 75\%$).



(a) Complete coverage time per UAV for network traversing

(b) Travel distance per UAV

(c) Duplicate coverage rate per UAV

Fig. 7. Network exploration performance comparison with respect to the number of UAVs with the error bars of standard deviation.

We focus on more in-depth network performance improvements in a relatively small but critically damaged network even using a small number of UAVs. The simulation results are still valid for a large scale network under critical damage, with a fairly larger number of UAVs. We believe that our simulation setting does not provide qualitatively different results, serving as a reasonable representative to effectively show the inherent performance.

In our experiments, the total number of vertexes is 100 where $m = 10$, and UAVs fly at the height of 3 m. For relaying probing packets over stationary nodes, we use three maximum number of retransmissions. The parameters of $n = 1$ on the number of relaying hops, $k = 1$ on connectivity-based clustering, and $\lambda = 1$ on network hole replacement are tuned to be used for *DroneNet+*. The parameter of *initialWindow* is set to one unit time where one unit time is the traveling time from a vertex to its adjacent vertex, i.e., 7.2 seconds, in *Adv-DroneNet+*. We show the average performance over 10 independent simulations, unless otherwise noted.

Our validation is divided into two parts: network traversing based on motion planning and network hole replacement algorithms. First, we evaluate network traversing performance of *DroneNet*, *DroneNet+*, and *Adv-DroneNet+* in terms of complete coverage time, travel distance, and duplicate coverage rate by varying the number of UAVs compared to *Ants* [8] and a centralized optimal solution that solves the Multiple Traveling Salesman Problem, *mTSP* [49] serving as a theoretical bound. Second, we investigate network repair performance of *DroneNet*, *DroneNet+*, and *Adv-DroneNet+* in terms of end-to-end routing cost and source-to-destination pairs with no route, as opposed to an upper-bound counterpart algorithm. We quantify the computation complexity of our deployment algorithms in terms of the number of iterations and running time. Also, we evaluate dynamic network recovery performance as stationary nodes become dying out over time.

A. Network Traversing

We explore the efficiency of our UAV traversing algorithms. The flying speed of UAVs is assumed to be 11.1 m/s (as per Parrot AR.Drone 2.0 specification). The initial position of each UAV is placed at a randomly selected vertex. We measure the complete coverage time and the average travel distance of each individual UAV until UAVs finish the network exploration. The initial navigation width $L = 4$ is used. We quantify the duplicate coverage of how much the vertexes visited by a UAV are overlapped with those by other UAVs. We run 50 simulations and show the average performance in results.

Regarding the complete coverage time, all of our traversing algorithms outperform *Ants* that does not share visited vertex information with other agents. *DroneNet*, *DroneNet+*, and *Adv-DroneNet+* reduce the network traversing time spent for the complete coverage over RoI in Fig. 7(a). This implies that sharing previous trajectory information with other UAVs is essential to reduce duplicated exploration. Furthermore, *DroneNet+* lessens the network traversing time with up to 14.5% compared to *DroneNet*, while *Adv-DroneNet+* further reduces with up to 11.1% compared to *DroneNet+*. This means that the adaptive control of navigation width L by both incrementing and decrementing depending on the coverage progress of other UAVs plays an important role on navigation efficiency by reducing the duplicate coverage. Moreover, to deeply understand the traversing performance achievable by our proposed practical algorithms, we find a theoretical limit of traversing time by solving an *mTSP*, which offers an optimal solution. It should be noted that *mTSP* finds out the optimal traversal in a centralized manner under global knowledge, whereas our traversing algorithms perform in a distributed manner under only partially self-collected information. Although our algorithm keeps optimized from *DroneNet*, *DroneNet+* toward *Adv-DroneNet+*, which is a heuristic distributed one, its resulting performance is approaching to a theoretical bound of *mTSP* and has a relatively similar tendency with it, as the number of UAVs increases.

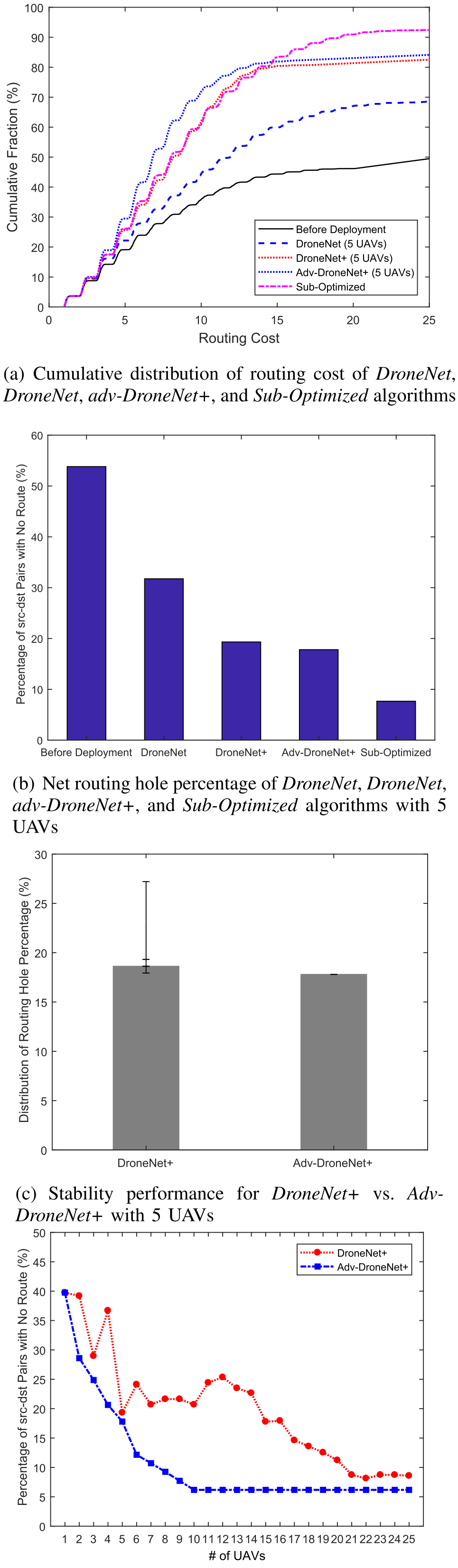
We measure travel distance and duplicate coverage in Figs. 7(b) and 7(c), respectively. Although *Adv-DroneNet+* has a slightly lower travel distance than *DroneNet+*, and *DroneNet+* has the better performance than *DroneNet*, all three motion planning algorithms reduce travel distance as the number of UAVs increases and show the similar performance on navigation efficiency in the space domain as in Fig. 7(b). As for duplicate coverage in Fig. 7(c), *DroneNet+* reduces the duplicate coverage to 11.5% compared to *DroneNet* with 20.7% thanks to the adaptive navigation control by decrementing the navigation width L. *Adv-DroneNet+* lessens the duplicate coverage down to 8.4%, even lower than *DroneNet+*, showing that the adaptive navigation control by widening the width is an important factor for improving navigation efficiency.

We discuss the scalability of our network traversing algorithm. Since our network traversing performs on the basis of vertex-based grid coordinate, its performance is not dependent of the number of stationary nodes, but rather dependent of the number of vertexes (which is fixed). Thus, the network traversing performance is scalable with the number of nodes.

B. Network Recovery

We investigate network recovery performance of our algorithms to measure how the selection of UAV deployment positions reduces the number of network holes and improves routing cost. We compare *DroneNet*, *DroneNet+*, and *Adv-DroneNet+* with an upper bound algorithm. We devise an upper bound algorithm, *Sub-Optimized* scheme that makes a recursive attempt to check all possible subsequent deployment positions given the past UAV deployments in a brute-force manner. This scheme makes a series of UAV deployment decisions that can lead to the lowest network hole fraction and the most effective route repair.

We measure routing cost in Fig. 8 when each algorithm of *DroneNet*, *DroneNet+*, *Adv-DroneNet+*, and a *Sub-Optimized* upper bound algorithm is applied to the same damaged network where 53.8% of source-to-destination routing pairs have no existing path as in Fig. 6. Routing cost is measured as the sum of the expected number of transmissions over routing hops. As the cumulative distributions of routing cost are shown in Fig. 8(a), *Adv-DroneNet+* outperforms *DroneNet* and *DroneNet+*, showing the lowest routing cost over the overall routing cost range. Also, one interesting observation is that our *Adv-DroneNet+*



(a) Cumulative distribution of routing cost of *DroneNet*, *DroneNet*, *adv-DroneNet+*, and *Sub-Optimized* algorithms

(b) Net routing hole percentage of *DroneNet*, *DroneNet*, *adv-DroneNet+*, and *Sub-Optimized* algorithms with 5 UAVs

(c) Stability performance for *DroneNet+* vs. *Adv-DroneNet+* with 5 UAVs

(d) Routing performance improvement with respect to the number of UAVs

Fig. 8. Network recovery performance comparison with a counterpart algorithm and one another in terms of the end-to-end routing cost.

based on a relatively lightweight iterative approach works better than *Sub-Optimized* upper bound algorithm that requires high computation complexity, under the routing scenarios with the low and the middle range of routing cost.

As measured in Fig. 8(b) for the net routing hole percentage by taking the average value over 10 simulations, *DroneNet*, *DroneNet+*, and *Adv-DroneNet+* lead to the network hole percentage of 31.7%, 19.3%, and 17.8%, respectively, while *Sub-Optimized* shows the lowest percentage of 7.7%. In particular for a more intricate performance comparison between *DroneNet+* and *Adv-DroneNet+*, we show the distribution of the percentiles (0, 25, 50, 75, and 100%) of the routing hole percentage as in Fig. 8(c). *DroneNet+* has performance variations due to some randomness from clustering, whereas *Adv-DroneNet+* has a very stable performance in its network recovery evaluation.

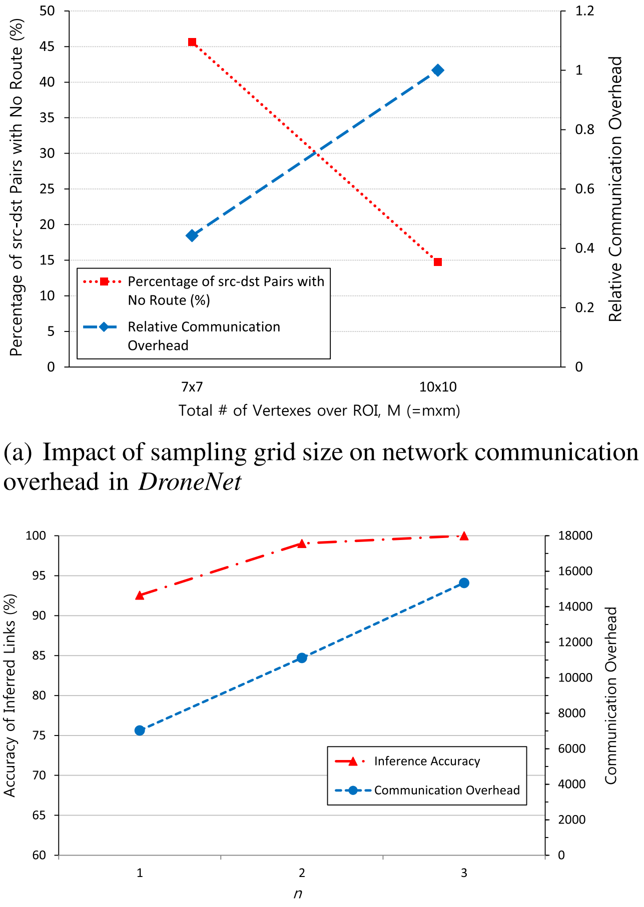
We investigate how each additional UAV deployment can enhance routing performance in Fig. 8(d). *DroneNet+* improves routing performance eventually with some fluctuation in the middle of UAV usage, while *Adv-DroneNet+* shows a more effective and stable performance. *DroneNet+* achieves the invalid route ratio of 8.7% via 21 UAVs, whereas *Adv-DroneNet+* requires only 10 UAVs reaching at an even lower ratio of 6.2%. More UAV deployment beyond this necessary number of UAVs may become redundant, but at the same time starts having fault-tolerance in return.

C. Effect of Design Parameters

We discuss how different design parameters in our algorithms affect system performance in Figs. 9 and 10. First, we investigate how the probing density affects routing performance and communication overhead in *DroneNet*. We measure communication overhead as the accumulated packet transmissions for sending hello packets and response packets from each UAV, and exchanging the vertex-visit-list and the PRR table among UAVs. As shown in Fig. 9(a), as the probing density increases from 7 × 7 to 10 × 10, approximately by two, we can achieve routing performance improvement with a factor of 3.1, while consuming more communication overhead with a factor of 2.3. This demonstrates that *DroneNet* can achieve a higher benefit of network-wide data delivery with a relatively smaller network overhead increase, showing an interesting trade-off relationship.

We explore the effect of the maximum hop distance of relaying the path-probing packet, n in *DroneNet+* on the accuracy of correctly inferred links among ground-truth links as in Fig. 9(b). As the the number of relaying hops increases, the inference accuracy also increases. The required communication overhead of path-probing broadcast, on the other hand, becomes accordingly larger. As a reasonable trade-off point, $n = 1$ is selected in our experiments where 92.6% of links are correctly inferred.

In *Adv-DroneNet+*, we have introduced the time window such that after passing the window without encountering other UAVs, the navigation width L is incremented by one. We explore the effect of the window size on network traversing efficiency, showing the average and the standard deviation over 50 simulations as in Fig. 10. Since UAVs spend 7.2 seconds to move from one grid point to another adjacent one, we use this value as one unit of the time window. As varying the time window from 0 to 2 (i.e., from 0 second to 14.4 seconds), all of traversing metrics, network traversing time, travel distance, and duplicate coverage lead to the lowest upon selecting one unit of time window (i.e., 7.2 seconds). This means that the navigation width would rather be reverted back when the UAV reaches a next grid point after encountering a UAV and leaving the current grid point while moving with its reduced navigation width.



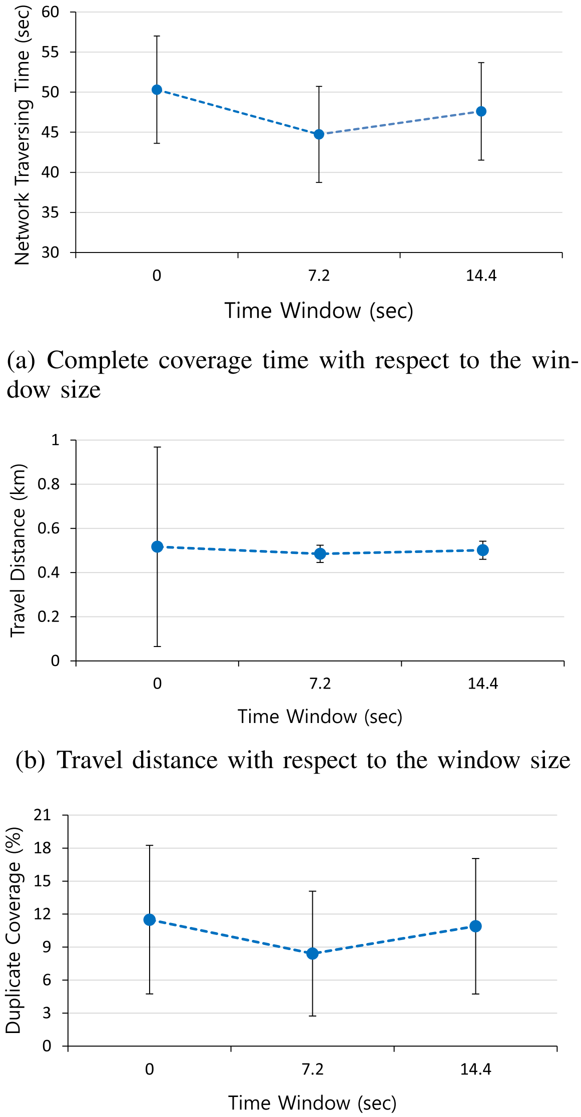
(a) Impact of sampling grid size on network communication overhead in *DroneNet*

(b) Valid route inference performance and communication overhead of broadcast control packets in *DroneNet+*

Fig. 9. Effect of design parameters in *DroneNet* and *DroneNet+*.

D. Computation Complexity

We measure computation complexity in terms of running time in Fig. 11. Our experiments have used LG B70CV desktop with Intel i7-4790 3.60 GHz CPU and 8 GB RAM. We quantify the running time between *DroneNet* and *DroneNet+*, and between *DroneNet+* and *Adv-DroneNet+*. As in Fig. 11(a), our *DroneNet+* spends a little time to compute the solution within seconds based on a heuristic distributed algorithm, whereas *DroneNet* takes much more time to solve its optimization problem based on a centralized computation. This implies that our algorithm provides a lightweight practical approach, making it feasible with a larger number of UAVs. Regarding the comparison between *DroneNet+* and *Adv-DroneNet+*, as in Fig. 11(b), *Adv-DroneNet+* takes more running time for partitioning into strongly connected components than *DroneNet+*. This is a performance trade-off between network recovery and computation complexity for choosing either *DroneNet+* or *Adv-DroneNet+*.



(a) Complete coverage time with respect to the window size

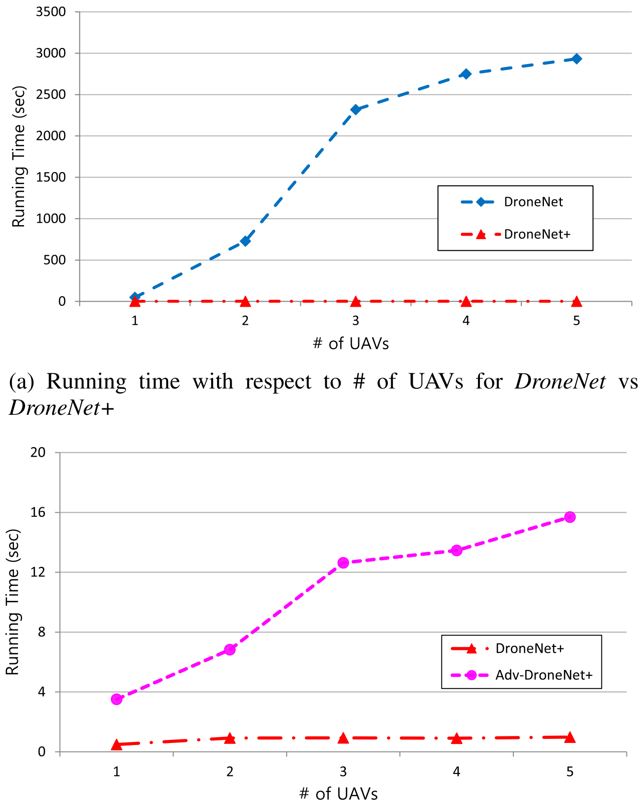
(b) Travel distance with respect to the window size

(c) Duplicate coverage rate with respect to the window size

Fig. 10. Network exploration performance comparison with respect to the window size in *Adv-DroneNet+*.

We discuss the scalability of our deployment algorithms. Both *DroneNet+* and *Adv-DroneNet+* first find out crucial clusters, and then decide the deployment position in the space of the vertex grid coordinate, which is independent of the number of stationary nodes. Thus, our deployment algorithms are also resilient with the number of nodes in the network, achieving scalability.

E. Dynamic Performance



(a) Running time with respect to # of UAVs for *DroneNet* vs. *DroneNet+*

(b) Running time with respect to # of UAVs for *DroneNet+* vs. *Adv-DroneNet+*

Fig. 11. Computation complexity in terms of running time.

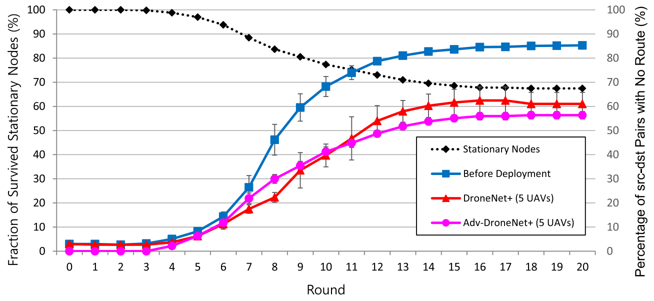


Fig. 12. Dynamic network recovery performance as stationary nodes become dying out over time.

We examine dynamic network recovery performance of *DroneNet+* and *Adv-DroneNet+* in a gradual network breakdown scenario. We use a simulated network of 100 stationary nodes with the initial power budget of 2.5 W over the RoI of 144 × 144 m^2. It is assumed that the radio transmission and reception drive the current of 17.4 mA and 19.7 mA, respectively, with the external power supply of 3.3 V, according to the MicaZ mote specification. If a node consumes all the remaining power, we let it inactive for any network operation so that the network can get disconnected gradually over time. At each round, 30 source-to-destination pairs randomly chosen perform data transmission along their own shortest path. As in Fig. 12, as the number of active stationary nodes even gradually decreases, the network gets dramatically disconnected, significantly breaking down existing routes. If both *DroneNet+* and *Adv-DroneNet+* are allowed to apply their own adaptive route recovery procedure using 5 UAVs at each round, the speed of the network breakdown becomes slower compared to no deployment case. While *Adv-DroneNet+* outperforms *DroneNet+* in general under the steady-state performance, *DroneNet+* can sometimes work better in some cases (e.g., in the round of 6–10). This is because our current *Adv-DroneNet+* finds out a deployment position that can directly connect two partitioned components via only one newly deployed UAV, whereas *DroneNet+* finds out a series of deployment position that can connect two crucial isolated nodes via at most two newly deployed UAVs in series. Although the effective number of nodes participating in the network keeps decreasing due to the battery outage even after deploying 5 UAV relays, our both adaptive UAV deployment algorithms keep reorganizing their effective deployment positions at each round with fault-tolerance, avoiding substantial route outages as much as possible.

VII. Discussions

We discuss some crucial aspects: 1) how the effective number of UAVs is associated with the network collapse degree; 2) when a certain factor between network traversing and deployment decision is more important; 3) the possibility of integrating network traversing and deployment into one stage with a progressive manner for responsiveness; and 4) practical issues for real-world applications.

A. Relationship of the Number of UAVs With the Network Collapse Degree

Depending on the network collapse degree after a disaster, the suitable number of UAVs for network recovery varies. In a damaged network with almost half broken source-to-destination pairs with no route as in Fig. 6, using 5 UAVs is a cost-effective choice, showing feasible network traversing and routing performance. If the network becomes more critically damaged over time or after a subsequent disaster, the required number of UAVs may increase. By employing more UAV resource, the advantage is two-fold. First, the overall network traversing time can be reduced due to our collaborative exploration algorithm by letting distributing the given coverage area to more UAVs with less overlap. Second, the overall network recovery performance can be enhanced by deploying more number of relays where critically damaged sub-networks are located. On the other hand, some computation overhead for finding their suitable deployment positions is inevitable in return. Thus, the number of UAVs to use as relays should be determined by considering the trade-off relationship among network collapse degree, network traversing time, network recovery performance, and computation complexity all together.

B. Network Traversing vs. Deployment Decision

Our proposed scheme consists of network traversing and deployment decision with two phases. In terms of running time, the network traversing phase takes a longer portion than the deployment phase since the physical vertex-to-vertex movement by UAVs usually takes more time than running an algorithm in an embedded system. In case that the execution time is the most critical factor for operating this system under emergency, the network traversing can be a bottleneck. Leveraging a large number of UAVs (irrespective of the network collapse degree) would decrease the network traversing time. On the other hand, calculating the deployment positions for the increased number of UAVs takes more time in the subsequent deployment phase. In this case, we may use only necessary UAVs for deployment where the number of UAVs in the deployment phase is not necessarily the same as that in the former phase. Since the network traversing is decoupled with the UAV deployment, we can separately choose the necessary number of UAVs for network traversing and deployment, respectively.

Although the network traversing and the deployment are decoupled in their separate phases, the deployment performance is highly associated with the former network traversing performance. If the network traversing granularity becomes sparse, both network traversing time and communication overhead would be saved. However, the subsequent deployment position should be determined among rough candidate vertexes, leading to less accurate deployment decision. Therefore, the network traversing and the deployment decision should be co-optimized considering the constraints of given physical resource, time, and routing performance.

C. Progressive Network Traversing and Deployment

Our two-phase network reconstruction scheme may not be an optimal solution for quickly recovering a damaged network since UAVs can start contributing as relays only after the precedent network traversing procedure is completed. In this situation, integrating both traversing and deployment into one stage can be a more effective solution. Each UAV starts exploring its local space and determining its temporary deployment position among possible locally optimal candidates. As a UAV becomes more knowledgeable by itself or from other encountered UAVs, it can gradually find a more globally optimal deployment position with a progressive manner. In this way, the response time for network recovery can significantly be reduced even if its initial performance is not optimal, and the performance gets more improved as the operation continues. Also, this progressive network traversing and deployment can be a practically better fit to a dynamically changing disaster scenario than a one-shot (unrealistic) disaster scenario thanks to its reduced response time.

Our two-phase network reconstruction scheme runs a centralized computation for obtaining the optimal deployment positions. On the other hand, the progressive one-phase scheme can compute them in a distributed manner based on the currently obtained information at the UAV, without collecting global information from all the UAVs. In this way, the algorithm can be more resilient against unexpected data collection or communication failure scenarios between UAVs.

D. Practical Issues

We discuss some real-world considerations for the practical feasibility of our proposed algorithms, and how we could extend our work to reflect more practical issues.

1) Battery of UAVs: To reflect the battery issue of UAVs, we need to estimate the empirical relationship between the remaining energy and how long a UAV can operate. Based on this empirical model, we can let a UAV go back to a predetermined charger station in need or via an energy-aware path-planning among communicable UAVs. Upon deployment, we can extend our algorithm such that for example, more powered UAVs may be deployed to more critically disconnected areas to achieve more energy-tolerant deployment decisions under energy-constrained UAVs.

2) Physical Constraints: A real-world environment is surrounded by various physical constraints such as obstacles, and building structures that may impede the UAVs' mobility. To overcome the challenge, we can extend to a path planning in 3D space. During the probing procedure, UAVs may need to collect obstacle information as well as network information. We also have to consider the effect of obstacles on UAVs' communication coverage during the deployment.

3) Mobility: If some nodes with moderate mobility are included in a terrestrial network, the constructed network topology may be outdated soon, and accordingly, its subsequent one-shot deployment decision would not provide accurate results. In this scenario, the network traversing and the deployment procedures should be more tightly coupled so that the network topology needs to be updated over time, and a dynamic deployment decision based on the up-to-date probing information should be made.

VIII. Conclusion

We have presented a self-organizing UAV deployment algorithm based on sparse network status probing from the air along with a distributed coverage path planning by UAVs. We have designed several variants of network traversing algorithm based on a fully distributed local decision for its next movement, while minimizing the duplicate coverage and guaranteeing complete coverage. Then, we have presented topology discovery schemes for capturing both local and non-local route topology that embeds inherent route skeletons based on broadcast and path stitching techniques. By pinpointing network hole locations, we have developed several practical network hole replacement algorithms that dispatch a limited number of UAVs to the selected hole spots, leading to both local and global improvement on routing performance.

Our experiments demonstrate that our scheme has significantly achieved both network probing efficiency and routing recovery performance by exploiting the efficiency in coverage path planning and UAV deployment compared to a baseline counterpart, the popularly used multi-agent exploration algorithm *Ants* and a sub-optimal brute-force algorithm. Our proposed framework provides a well-balanced mixture of network traversing and relay deployment by letting each UAV play a suitable role at each necessary step.

For future work, we may interleave the route topology discovery together with progressive UAV deployment, achieving higher efficiency and responsiveness, and suppressing unnecessary discovery. It would be interesting to consider more practical aspects such as battery recharging of UAVs to reflect in the path planning of UAVs, and considering mobile nodes in terrestrial networks. Also, we could extend the problem by allowing UAV's transmission power control, and validate our algorithms in a real-world testbed consisting of terrestrial sensors and aerial drones.

References

[1] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in UAV communication networks," *IEEE Commun. Surv. Tut.*, vol. 18, no. 2, pp. 1123–1152, Secondquarter 2016.

[2] I. Rubin and R. Zhang, "Placement of UAVs as communication relays aiding mobile ad hoc wireless networks," in *Proc. IEEE Mil. Commun. Conf.*, Oct. 2007, pp. 1–7.

[3] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme, "Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2004, pp. 3602–3608.

[4] M. Erdelj, M. Krl, and E. Natalizio, "Wireless sensor networks and multi-UAV systems for natural disaster management," *Comput. Netw.*, vol. 124, pp. 72–86, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128617302220

[5] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the sky: Leveraging UAVs for disaster management," *IEEE Pervasive Comput.*, vol. 16, no. 1, pp. 24–32, Jan. 2017.

[6] E. Ferranti, N. Trigoni, and M. Levene, "Brick& mortar: An on-line multi-agent exploration algorithm," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2007, pp. 761–767.

[7] N. Hazon, F. Mieli, and G. A. Kaminka, "Towards robust on-line multi-robot coverage," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2006, pp. 1710–1715.

[8] J. Svennebring and S. Koenig, "Building terrain-covering ant robots: A feasibility study," *Auton. Robots*, vol. 16, no. 3, pp. 313–332, 2004.

[9] E. Ferranti, N. Trigoni, and M. Levene, "Rapid exploration of unknown areas through dynamic deployment of mobile and stationary sensor nodes," *Auton. Agents Multi-Agent Syst.*, vol. 19, no. 2, pp. 210–243, 2009.

[10] D. Devaurs, T. Siméon, and J. Cortés, "Optimal path planning in complex cost spaces with sampling-based algorithms," *IEEE Trans. Automat. Sci. Eng.*, vol. 13, no. 2, pp. 415–424, Apr. 2016.

[11] A. Yazici, G. Kirlik, O. Parlaktuna, and A. Sipahioglu, "A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints," *IEEE Trans. Cybern.*, vol. 44, pp. 305–314, Mar. 2014.

[12] N. Ahmed, S. S. Kanhere, and S. Jha, "The holes problem in wireless sensor networks: A survey," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 9, no. 2, pp. 4–18, 2005.

[13] M. Asadpour, D. Giustiniano, and K. A. Hummel, "From ground to aerial communication: Dissecting WLAN 802.11 n for the drones," in *Proc. 8th ACM Int. Workshop Wireless Netw. Testbeds Exp. Eval. Characterization.*, 2013, pp. 25–32.

[14] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus, "Deploying sensor networks with guaranteed capacity and fault tolerance," in *Proc. 6th ACM Int. Symp. Mobile ad hoc Netw. Comput.*, 2005, pp. 309–319.

[15] V. Isler, S. Kannan, and K. Daniilidis, "Sampling based sensor-network deployment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2004, vol. 2, pp. 1780–1785.

[16] E. Yanmaz, R. Kuschnig, and C. Bettstetter, "Achieving air-ground communications in 802.11 networks with three-dimensional aerial mobility," in *Proc. IEEE INFOCOM*, 2013, pp. 120–124.

[17] Z. Han, A. L. Swindlehurst, and K. J. R. Liu, "Optimization of MANET connectivity via smart deployment/movement of unmanned air vehicles," *IEEE Trans. Veh. Technol.*, vol. 58, no. 7, pp. 3533–3546, Sep. 2009.

[18] I. F. Senturk, K. Akkaya, and S. Yilmaz, "Relay placement for restoring connectivity in partitioned wireless sensor networks under limited information," *Ad Hoc Netw.*, vol. 13, Part B, pp. 487–503, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870513002084

[19] D. Lee, K. Jang, C. Lee, G. Iannaccone, and S. Moon, "Path stitching: Internet-wide path and delay estimation from existing measurements," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–5.

[20] D. Jeong, S. Y. Park, and H. Lee, "DroneNet: Network reconstruction through sparse connectivity probing using distributed UAVs," in *Proc. IEEE 26th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun.*, Aug. 2015, pp. 1797–1802.

[21] S.-Y. Park, D. Jeong, C. S. Shin, and H. Lee, "DroneNet+: Adaptive route recovery using path stitching of UAVs in ad-hoc networks," in *Proc. IEEE Global. Commun. Conf.*, 2017.

[22] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous Robot Vehicles*. New York, NY, USA: Springer, 1986, pp. 396–404.

[23] A. Zelinsky, R. A. Jarvis, J. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Proc. Int. Conf. Adv. Robot.*, 1993, vol. 13, pp. 533–538.

[24] S. Hert, S. Tiwari, and V. Lumelsky, "A terrain-covering algorithm for an AUV," *Autonomous Robots*, vol. 3, pp. 91–119, Jun. 1996.

[25] V. J. Lumelsky, S. Mukhopadhyay, and K. Sun, "Dynamic path planning in sensor-based terrain acquisition," *IEEE Trans. Robot. Automat.*, vol. 6, no. 4, pp. 462–472, Aug. 1990.

[26] J.-C. Latombe, *Robot Motion Planning*, vol. 124. Berlin, Germany: Springer Science & Business Media, 2012.

[27] H. Choset, "Coverage for robotics–a survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, no. 1–4, pp. 113–126, 2001.

[28] M. Arzamendia, D. Gregor, D. G. Reina, and S. L. Toral, "An evolutionary approach to constrained path planning of an autonomous surface vehicle for maximizing the covered area of ypacarai lake," *Soft Comput.*, pp. 1–12, 2017.

[29] K. Cesare, R. Skeele, S.-H. Yoo, Y. Zhang, and G. Hollinger, "Multi-UAV exploration with limited communication and battery," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 2230–2235.

[30] S. K. Gan and S. Sukkarieh, "Multi-UAV target search using explicit decentralized gradient-based negotiation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 751–756.

[31] T.-S. Lee, J.-S. Choi, J.-H. Lee, and B.-H. Lee, "3-d terrain covering and map building algorithm for an auv," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2009, pp. 4420–4425.

[32] P. N. Atkar, H. Choset, A. A. Rizzi, and E. U. Acar, "Exact cellular decomposition of closed orientable surfaces embedded in r3," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2001, vol. 1, pp. 699–704.

[33] P. N. Atkar, A. Greenfield, D. C. Conner, H. Choset, and A. A. Rizzi, "Uniform coverage of automotive surface patches," *Int. J. Robot. Res.*, vol. 24, no. 11, pp. 883–898, 2005.

[34] M. Younis and K. Akkaya, "Strategies and techniques for node placement in wireless sensor networks: A survey," *Ad Hoc Netw.*, vol. 6, no. 4, pp. 621–655, 2008.

[35] C. Zhu, C. Zheng, L. Shu, and G. Han, "A survey on coverage and connectivity issues in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 35, no. 2, pp. 619–632, 2012.

[36] G. Wang, G. Cao, and T. F. La Porta, "Movement-assisted sensor deployment," *IEEE Trans. Mobile Comput.*, vol. 5, no. 6, pp. 640–652, Jun. 2006.

[37] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems 5*. New York, NY, USA: Springer, 2002, pp. 299–308.

[38] N. Heo and P. K. Varshney, "An intelligent deployment and clustering algorithm for a distributed mobile sensor network," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2003, pp. 4576–4581.

[39] A. Howard, M. J. Matarić, and G. S. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Auton. Robots*, vol. 13, no. 2, pp. 113–126, 2002.

[40] Z. Han, A. L. Swindlehurst, and K. R. Liu, "Smart deployment/movement of unmanned air vehicle to improve connectivity in MANET," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2006, vol. 1, pp. 252–257.

[41] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage," *IEEE Commun. Lett.*, vol. 20, no. 8, pp. 1647–1650, Aug. 2016.

[42] D. Reina, H. Tawfik, and S. Toral, "Multi-subpopulation evolutionary algorithms for coverage deployment of UAV-networks," *Ad Hoc Netw.*, vol. 68, pp. 16–32, 2018.

[43] R. Magán-Carrión, R. A. Rodríguez-Gómez, J. Camacho, and P. García-Teodoro, "Optimal relay placement in multi-hop wireless networks," *Ad Hoc Netw.*, vol. 46, pp. 23–36, 2016.

[44] M. Gerla and J. T.-C. Tsai, "Multicluster, mobile, multimedia radio network," *Wireless Netw.*, vol. 1, no. 3, pp. 255–265, Aug. 1995. [Online]. Available: http://dx.doi.org/10.1007/BF01200845

[45] F. G. Nocetti, J. S. Gonzalez, and I. Stojmenovic, "Connectivity based k-hop clustering in wireless networks," *Telecommun. Syst.*, vol. 22, no. 1, pp. 205–220, 2003. [Online]. Available: http://dx.doi.org/10.1023/A:1023447105713

[46] T. H. Cormen, *Introduction to Algorithms*. Cambridge, MA, USA: MIT press, 2009.

[47] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, 1972.

[48] H. Lee, A. Cerpa, and P. Levis, "Improving wireless simulation through noise modeling," in *Proc. 6th Int. Symp. Inf. Process. Sens. Netw.*, Apr. 2007, pp. 21–30.

[49] J. Kirk, "Multiple traveling salesmen problem—genetic algorithm," 2008. [Online]. Available: https://mathworks.com/matlabcentral/fileexchange/19049-multiple-traveling-salesmen-problem-genetic-algorithm

So-Yeon Park received the B.S. and M.S. degrees all in computer science from Ewha Womans University, Seoul, South Korea, in 2016 and 2018, respectively. Her research interests include handover decision algorithm, network resource optimization, network reconstruction by unmanned aerial vehicles, IoT security system, and ad hoc networks.

Christina Suyong Shin received the B.S. degree from Ewha Womans University, Seoul, South Korea, in 2017. She is currently working toward the M.S. degree from Ewha Womans University. Her research interests include wireless sensor ad hoc networks, fog computing, and currently on vehicular ad hoc networks.

Dahee Jeong received the B.S. and M.S. degrees in computer science from Ewha Womans University, Seoul, South Korea, in 2015 and 2017, respectively. She was on intelligent security systems based on mobility pattern analysis and ad hoc network reconstruction using unmanned aerial vehicles.

HyungJune Lee received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2001, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 2006 and 2010, respectively. He is an Associate Professor with the Department of Computer Science and Engineering, Ewha Womans University, Seoul, South Korea. He was with Broadcom as Sr. Staff Scientist for working on research and development of 60GHz 802.11ad SoC MAC. Also, he was with AT&T Labs as Principal Member of Technical Staff with the involvement of LTE overload estimation, LTE-WiFi interworking, and heterogeneous networks. His current research interests include future wireless networks on IoT, fog computing, VANET, 60GHz Wi-Fi, and heterogeneous networks.