Progressive Ad-Hoc Route Reconstruction Using Distributed UAV Relays after a Large-Scale Failure

Christina Suyong Shin, So-Yeon Park, JinYi Yoon, and HyungJune Lee
Department of Computer Science and Engineering
Ewha Womans University, South Korea
Email: hyungjune.lee@ewha.ac.kr

***Abstract*—In this paper, we address a route reconstruction problem using Unmanned Aerial Vehicles (UAVs) after a large-scale disaster where stationary ad-hoc networks are severely destructed. The main goal of this paper is to improve routing performance in a progressive manner by reconnecting partitioned networks through dispatched UAV relays. Our proposed algorithm uses two types of UAVs: *global* and *local* UAVs to collaboratively find the best deployment position in a dynamically changing environment. To obtain terrestrial network connectivity information and extract high-level network topology, we exploit the concept of *strongly connected component* in graph theory. Based on the understanding from a global point view, global UAVs recommend the most effective deployment positions to local UAVs so that they are deployed as relays in more critically disrupted areas. Simulation-based experiments validate that our distributed route reconstruction algorithm outperforms a counterpart algorithm in terms of steady-state and dynamic routing performance.**

I. Introduction

In a large-scale disaster, ad-hoc networks would severely be destructed, and a fast network recovery is an important task. There have been several efforts that try to reconnect the partitioned networks. For example, dispatching additional ad-hoc nodes (as relay nodes) to the network can be one of solutions. However, even grasping destructed ad-hoc network may be very challenging because network destruction occurs in a large-scale area, also limiting direct access to the area.

To tackle the problem, Unmanned Aerial Vehicles (UAVs) can effectively be applied for recovering disrupted network and have several advantages when they are utilized. Firstly, UAVs can freely fly over the network area where direct access is almost impossible. Secondly, UAVs can be used to gather network information from the air [11], [12]. UAVs can communicate not only with other UAVs, but also with terrestrial ad-hoc nodes so that they can be used to recover disjoint networks by acting as temporary relays [4], [7].

The problem of finding the optimal location of relay nodes has been investigated under the *relay node placement* problem [6]. Traditionally, this problem is transformable to *Steiner minimum tree with minimum number of Steiner points and bounded edge length problem* (SMT-MSPBEL), and it has been proved as NP-hard [1]. Researchers aim to solve this problem with some heuristic algorithms [5], [8], [9]. However, they still suffer from intrinsic high complexity and can not be directly applied to realistic disaster environments.

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2015R1D1A1A01057902).

Recently, dispatching UAVs as temporary relay nodes has been proposed in [3], [4], [7]. Our previous works [4], [7] perform network probing for diagnosis and determine UAV relays' deployment positions. [4] has a drawback of having high complexity because it relies on somewhat heavy optimization computation. To address the computation issue, [7] suggests a lightweight yet efficient algorithm. However, both works are based on centralized algorithms and find the one-shot deployment solution without taking into account dynamical deployment decision.

This paper presents a progressive route reconstruction algorithm using distributed UAV relays, which is more suitable to realistic disaster environments. Our proposed algorithm uses two types of UAVs, called *local* UAVs and *global* UAVs. We design them to collaboratively find out the best deployment position in terms of routing performance improvement. Local UAVs densely traverse with a short stride and probe local network information. Based on the collected local network information, local UAVs iteratively perform local optimization to find a deployment position with its own best effort.

Global UAVs, on the other hand, roughly traverse with a long stride and construct a condensed higher-level network topology by exploiting the concept of *strongly connected component* in graph theory. Global UAVs recommend globally more optimized deployment positions to their paired local UAVs to avoid the danger of falling into a local optimal deployment position determined by local UAVs. Accordingly, local UAVs can be deployed as relays in more critically disrupted areas from a global point of view.

Our proposed route reconstruction scheme progressively finds a way to improve end-to-end routing performance by utilizing a mixture of local and global deployment decision.

The rest of this paper is organized as follows: After explaining system model in Sec. II, we present our network traversing algorithm in Sec. III. In Sec. IV, we describe our progressive deployment algorithm. After demonstrating evaluation results in Sec. V, we conclude this paper in Sec. VI.

II. System Model

We consider a route reconstruction problem using UAVs in a large-scale disaster scenario where stationary ad-hoc networks are severely disconnected in parts. Our main objective is to improve routing performance in a progressive manner by repairing network connectivities using distributed UAVs as relay nodes as illustrated in Fig. 1.

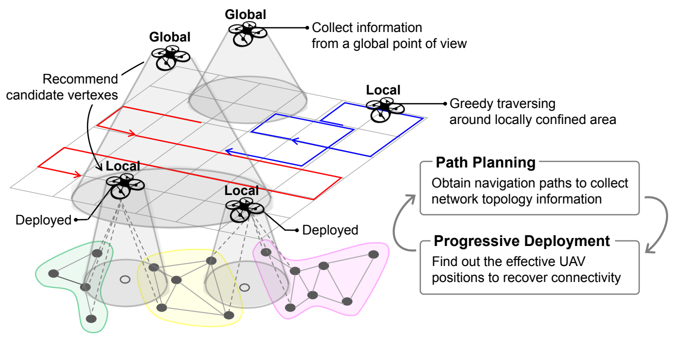

Global
Collect information from a global point of view
Global
Recommend candidate vertexes
Local
Greedy traversing around locally confined area
Local
Path Planning
Obtain navigation paths to collect network topology information
Local
Deployed
Deployed
Progressive Deployment
Find out the effective UAV positions to recover connectivity

Fig. 1. Overall procedure of our proposed algorithm

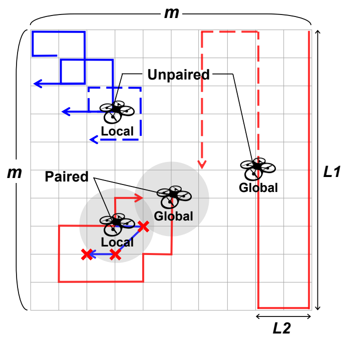


Fig. 2. Traversing mode evolution between local UAV and global UAV

We assume that UAVs are able to communicate with other UAVs or terrestrial ad-hoc nodes within the communication range using a wireless interface such as 802.11 or 802.15.4. It is also assumed that UAVs can keep track of their current positions over a certain designated Region of Interest (RoI) based on their equipped Global Positioning System (GPS). External restrictions such as weather, obstacles, and collisions with other UAVs, and battery shortage issues are not considered in this paper.

The problem of route reconstruction using UAV relays can be divided into two sub-problems: 1) network probing through traversing of UAVs for diagnosis, and 2) relay deployment decision that finds out the most effective UAV positions in terms of recovery performance for the disconnected network.

III. Network Traversing by UAVs

On the occurrence of a large-scale disaster, communication networks would be severely damaged. Further, the disaster event may not occur in one-shot, but would rather be followed by a series of after-shocks or lead to continuous and sometimes even severer network disruption. In this situation, UAVs can play crucial roles in both network status diagnosis and network relaying as a replacement of damaged nodes.

To cope with the dynamically changing network environment, one type of UAVs, called *local* UAVs, start to be deployed to one location and iteratively change its deployed location over time. The local UAVs locally move *with a short stride* over the nearby areas and find a better deployment position with the best effort based on its locally collected information. On the other hand, another type of UAVs, called *global* UAVs, traverse over a relatively larger area *with a long stride* to find out a critically disrupted area from a higher global point of view. A global UAV recommends several better deployment locations to its paired local UAV so that the local UAV can determine the next (better) deployment location among them. Both types of UAVs continuously collaborate to find a more globally efficient deployment position once they become paired as in Fig. 2.

In this section, we present path planning algorithms for local and global UAVs and communication protocols upon the encounter of two (or more) UAVs in the air (e.g., local-to-local, local-to-global, and global-to-global) for network traversing.

A. Path Planning

To obtain efficient navigation paths for local and global UAVs, we define a grid map over the RoI that consists of $m \times m$ virtual vertexes on a certain height in the air. UAVs move along these grid vertexes with a distributed manner according to their own state. Global UAVs do along a relatively wider *zigzag* trajectory in a relatively more global manner, while local UAVs initiate the network traversing with a greedy local manner.

Once a global UAV encounters a local UAV within the radio range, two UAVs are paired with a certain amount of time. The local UAV is relocated to one location of which the global UAV recommends.

1) Global UAVs: Global UAVs traverse the RoI area with a rough way by following a *zigzag* trajectory with the height of $L1$ and the width of $L2$ (as in Fig. 2). In this way, they obtain network connectivity status over a relatively larger area. During this zigzag movement, it may encounter one or more local UAVs that move with their own trajectory or have been deployed at some locations, and then become paired with it. Once paired, the global UAV recommends deployment candidates to the local UAVs. Then, it generates an encircling trajectory with one hop away from the recommended candidate locations and keeps collecting network connectivity information. We let each global UAV traverse near its paired local UAVs for suggesting new recommendations to each local UAV on the way. Unless it encounters again with its paired local UAVs, it keeps extending the encircling trajectory by increasing the number of hops up to n from the original candidate locations recommended by itself. In case that the global UAV meets its paired local UAVs again on the way, it recommends new deployment candidate locations based on the latest network information and follows the aforementioned steps. Otherwise, it finishes the encircling trajectory-based exploration with n hops away while becoming unpaired, and reverts back to the zigzag trajectory-based exploration.

2) Local UAVs: Local UAVs explore locally confined areas around the nearby vertexes by gradually moving along an outer clockwise or counter-clockwise trajectory. From the vertex where the local UAV is currently located, it generates its future trajectory that covers its nearby unvisited square area as much as possible to find out local network connectivity status. Otherwise, it randomly chooses one of four possible adjacent vertexes and then moves to it. Then, it randomly selects either clockwise or counter-clockwise direction, and generates its precedent trajectory path. Once the local UAV meets a global UAV and becomes paired, it changes the local traversing mode. The local UAVs attempt to visit all the candidates that the global UAV has recommended based on its globally collected network information. After exploring all the recommended vertexes, the local UAV finally determines its newly deployment position. As long as the local UAV keeps paired with a global UAV, it continues to move its deployment positions based on the global UAV's recommendation. Once the local UAV is no longer paired after a pairing time window W, it keeps staying where it is lastly deployed based on the latest recommendation from the previously paired global UAV, unless encountering another global UAV.

B. *Communication*

During traversing, a UAV can communicate with parts of terrestrial nodes for network diagnosis, and with other UAVs for sharing its visiting history and collected network information when they are within the radio range.

1) UAV-to-Ground: We want UAVs to collect locally connected network topology information for capturing the current network status. To do this, we borrow *path stitching* proposed in [7]. For *path stitching*, UAVs perform two tasks of sending out a new probing packet toward terrestrial nodes and collecting a previous probing packet that has been generated by another UAV. Once a terrestrial node receives a probing packet from a UAV or another node, it relays the packet up to k hops to its neighbors nodes by recording the relay path. In this way, each UAV continues to retrieve local route information that will be used to construct essential network topology.

2) UAV-to-UAV: A UAV may encounter another UAVs during its traversing in the air within the radio range. To avoid duplicate coverage in the future trajectory, they share their visited vertex list each other. Also, they share the collected local route information such that a local UAV can expand the localized network information and be likely to choose a more efficient deployment position than before. A global UAV, on the other hand, can make the whole network exploration faster and recommend more globally efficient deployment positions to paired local UAVs.

The pairing process is initiated when a local UAV meets a global UAV that has availability to pair where it can accommodate up to N local UAVs at the same time. In case that the global UAV has insufficient network information to recommend deployment positions to a specific local UAV that is attempted to be paired, it releases the pairing process and becomes unpaired with that local UAV. During a certain pairing window W, a local UAV avoids the danger of falling into a local optimal deployment position. This happens thanks to its paired global UAV that retains a more global knowledge of the higher-level network topology and recommends more globally optimal deployment positions.

IV. Progressive Deployment of UAV Relays

In this section, we present our progressive deployment algorithm that gradually keeps optimizing UAVs' deployment positions based on the latest global network topology information over time. One type of UAVs (i.e., global UAVs) extract high-level network topology based on the collected network connectivity information while traversing with a long stride

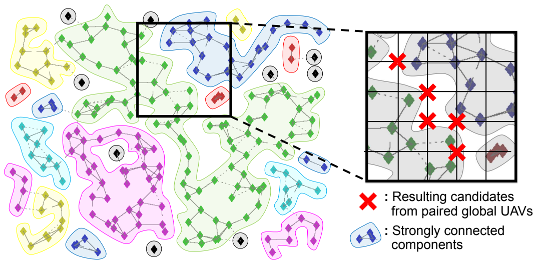

: Resulting candidates from paired global UAVs
: Strongly connected components

Fig. 3. Network partitioning based on strongly connected component and its resulting candidate vertexes that can effectively bridge two components

over the RoI. Another type of UAVs (i.e., local UAVs) make their own best to find a deployment position and be deployed with itself as relay based on localized network topology while traversing with a short stride over the RoI.

We exploit the concept of *strongly connected component* in graph theory to form a corresponding high-level network based on all the collected localized probing information. It allows us to discover an effective UAV deployment vertex through which two isolated sub-networks become tightly connected again.

A. *Global UAVs*

Global UAVs explore the disrupted network with a long stride of the zigzag pattern. Once a global UAV encounters a local UAV on the way, it tries to recommend good deployment positions for the local UAV based on its high-level understanding of the probed network information up to that point. Our algorithm for the global UAV case comprises three steps: 1) fundamental route stitching, 2) network partitioning, and 3) prioritizing grid vertexes for connecting isolated partitions, as illustrated in Fig. 3.

First, global UAVs construct the underlying network topology by stitching all the partial ad-hoc routing paths collected during network probing based on [7]. The partial ad-hoc routing paths are the ones that each global UAV has directly obtained, or the ones that it has received from other encountered UAVs.

Second, we let global UAVs perform high-level network partitioning where locally adjacent nodes with good network connection belong to one partition, called *strongly connected component* in graph theory [2], [10]. Strongly connected component can be more formally defined as a graph that every vertex is reachable from other vertexes. This means that every possible source-to-destination pair within a component has at least one valid route. In this environment, we focus on finding out effective vertex positions where local UAVs can be deployed as relays, and otherwise isolated components can be connected each other by creating new effective routes through the deployed UAV relay. In this way, connecting two components can significantly enhance routing performance by having routes from source nodes in one component to destination nodes in another component.

Third, based on the high-level partitioned network information, the global UAV decides a deployment vertex candidate list. It first calculates how much one component is strongly connected within itself by calculating the component priority as the number of edges divided by the number of nodes within the component. Then, the global UAV finds out all possible

Algorithm 1 Global UAV

```
1: Input: currentVertex, pairingState
2: Output: nextVertex, pairingState
3: if (any UAVs within radio range) then
4:     Share visiting history and global network information;
5:     if (encountered UAV is local UAV, $local$) then
6:         if (# of my pairs$<N$)&&($local$.pairingState==‘unpaired’) then
7:             Make a new pair with $local$;
8:             Process recommendation;
9:         else if (pairingState==‘paired’)&&($local$ is one of my pairs) then
10:            Initialize pairing time window to $W$;
11:            Process recommendation;
12:        end if
13:    end if
14: end if
15: if (pairingState == ‘unpaired’) then
16:     zigzagTrajectory = zigzag patten that starts with an unvisited vertex
        based on $L1$ and $L2$;
17:     if (zigzagTrajectory == $\emptyset$) then
18:         Done;
19:     end if
20:     nextVertex = the first vertex of zigzagTrajectory;
21: else
22:     for ($i$ = 1; $i \leq n$; $i$++) do
23:         encirclingTrajectory = unvisited $i$-hop vertexes from candidates;
24:         if (encirclingTrajectory $\neq \emptyset$) then
25:             nextVertex = the first vertex of encirclingTrajectory;
26:             break;
27:         end if
28:     end for
29:     if ($i > n$) then
30:         pairingState = ‘unpaired’;
31:         Regenerate zigzagTrajectory;
32:         nextVertex = the first vertex of zigzagTrajectory;
33:     end if
34: end if
35: Process UAV-to-Ground path probing;
```

two component pairs that are reachable via one grid vertex and chooses one pair of two components with the highest component priority summation among the pairs. Once the global UAV obtains one pair of two components with the highest priority for connection, it lists up all possible grid vertexes through which two components can be bridged.

As long as the global UAV keeps the pairing relationship with a local UAV, it can repeatedly recommend new deployment vertex candidates to the local UAV whenever encountering it again. A more detailed procedure for global UAVs is described in Algorithm 1.

B. Local UAVs

Local UAVs explore the disrupted network with a short stride along a greedy local encircling trajectory. Without any global knowledge of the disrupted network, it would be important to make the best effort to find out a locally disconnected area and be deployed at it for a progressive optimization.

Before a local UAV encounters any global UAV, it collects network probing packets at its visiting grid vertex and updates its own *local-inf-table*, which contains necessary information to calculate the percentage of source-to-destination pairs with no valid routes at each vertex. Whenever visiting a grid vertex, it finds out reachable stationary nodes as long as they are connected with multiple hops. Then, it calculates the percentage of the number of source-to-destination pairs with no valid route out of all possible node-to-node pairs. This metric implies a relative *route defect* at a vertex. A vertex with the highest

Algorithm 2 Local UAV

```
1: Input: currentVertex, pairingState, deployState
2: Output: nextVertex, pairingState, deployState
3: if (any UAVs within radio range) then
4:     Share visiting history and local network information with local-inf-
       table;
5:     if (encountered UAV is a global, $global$) then
6:         if (pairingState==‘unpaired’)&&($global$.pairNum$<N$) then
7:             Make a new pair with $global$;
8:             Update its candidate list;
9:         else if (pairingState==‘paired’)&&($global$ is my pair) then
10:            Initialize pairing time window to $W$;
11:            Update its candidate-list;
12:        end if
13:    end if
14: end if
15: if (pairingState == ‘unpaired’) then
16:     futureTrajectory = unvisited vertexes surrounding centerVertex;
17:     while (futureTrajectory == $\emptyset$) do
18:         newCenterVertex = selected vertex based on local-inf-table;
19:         futureTrajectory = unvisited vertexes surrounding newCenterVertex;
20:         if (newCenterVertex == centerVertex) then
21:             deployState = ‘deployed’;
22:             Deployed to centerVertex and wait;
23:         end if
24:     end while
25:     nextVertex = the first vertex of futureTrajectory;
26:     Calculate (# of no valid routes / # of possible src-dst pairs);
27:     Update its local-inf-table;
28: else
29:     futureTrajectory = unvisited vertexes in candidate-list;
30:     if (futureTrajectory == $\emptyset$) then
31:         Deploy to the worst vertex based on global-recommend-table;
32:     else
33:         nextVertex = the first vertex of futureTrajectory;
34:         Calculate priority1 and priority2;
35:         Update its global-recommend-table;
36:     end if
37: end if
```

value of the route defect metric has the highest priority to be deployed at. The local UAV traverses from a *centerVertex* and finds out a vertex with the highest priority. When the latest *centerVertex* is the same as its previous *centerVertex*, it starts to be deployed at the vertex and acts as a relay. The locally deployed UAV stays being deployed until it becomes paired with a global UAV.

Once a local UAV encounters a global UAV within its radio range, it checks whether the global UAV can accommodate to accept an additional pair request. After a successful pairing approval from the global UAV, the local UAV receives globally optimal deployment candidate vertexes from the global UAV. Given the candidate vertexes, the local UAV calculates two criterion metrics for each vertex. The first metric, *priority1* quantifies how many strongly connected components can be connected through the vertex if deployed there. A vertex that can connect with a larger number of components should be considered as a higher priority to be selected as a deployment vertex. The second metric, *priority2* measures how much the directly connected stationary nodes residing in separate components retain the centrality of the belonged component. We calculate the closeness centrality in a connected graph as the inverse sum of the length of a path from one node to another in the graph. A vertex with a higher closeness centrality should get higher attention to be selected as a deployment vertex. One vertex with the highest *priority1* is selected as a deployment vertex. If there are multiple vertexes with the same highest *priority1*, then we select a vertex that has the highest *priority2* among them. Similar to the unpaired case, the local UAV stays being deployed until its currently paired global UAV recommends a new candidate list if encountered.

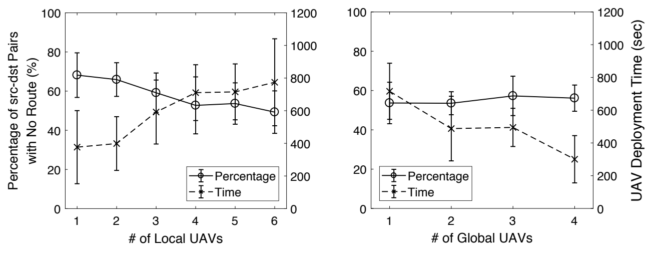
In case that two or more local UAVs are determined to be deployed at the same vertex, the UAVs start negotiation process: for unpaired-unpaired and unpaired-paired cases, one of unpaired local UAVs leaves the current deployment vertex to another vertex located at the second place in its *local-inf-table*. This is because another paired local UAV would rather be deployed at the current deployment vertex based on its paired global UAV's recommendation, not based on some myopic local information on the unpaired local UAV side. For the paired-paired case, a local UAV that has more vertex candidates as the second choice flies to the vertex from the current duplicated vertex. In case that there is no second choice to replace the current duplicated vertex, it just randomly selects one of adjacent vertex with the highest priority based on its *local-inf-table*. A more detailed procedure for local UAVs is described in Algorithm 2.

V. EVALUATION

We validate our proposed algorithm in a network of 218 stationary ad-hoc nodes over the RoI of 465 $\times$ 465 m^2 as in Fig. 3 where wireless links with packet reception rate larger than or equal to 75% are shown. To simulate a disrupted network after a disaster event, we make its sub-network isolated so that 74.95% of source-to-destination pairs have no existing route. A combined path-loss shadowing model with a path-loss exponent of 3, a reference loss of 46.66 dB, and an additive white Gaussian noise $N(0, 5^2)$ in dB is used as the wireless propagation model.

In our experiments, the virtual coordinate of 256 vertexes with $m = 16$ is used so that the geographic distance between two adjacent vertexes is within the communication range. We assume that UAVs fly at the height of 9 m with a speed of 11.1 m/s (as per Parrot AR.Drone 2.0). It is also assumed that the $zigzag$ trajectory for global UAVs is with the height $L1$ of 10 and the width $L2$ of 2, while the encircling trajectory is extended up to 4 hops where n is 4. For the pairing process, the pairing time window W of 280 sec is used, and a global UAV can accommodate up to 6 local UAVs at the same time. Regarding the usage of *path stitching*, the network probing packets are allowed to be passed up to 1 hop, which has been proved to be effective in [7]. We run 10 simulations for each experiment to obtain a statistically meaningful result and show the average performance.

Our evaluation is divided into four parts. First, we investigate steady-state network recovery performance as we can use more local or global UAVs. Second, we evaluate dynamic network recovery performance in terms of the number of source-to-destination pairs with no existing route over time. We compare our proposed algorithm against our previous work *DroneNet+*. We also devise a centralized version of our algorithm assuming that the algorithm runs on top of the global knowledge of all probed routes and their strongly



(a) Performance with respect to the number of local UAVs with one global UAV

(b) Performance with respect to the number of global UAVs with 5 local UAVs

Fig. 4. Steady-state routing hole percentage and deployment completion time performance

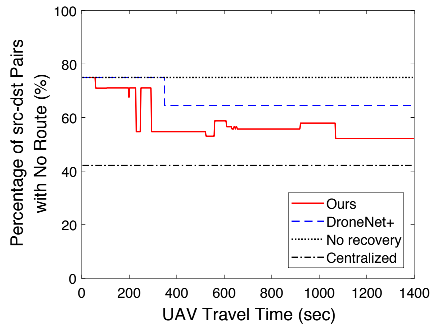


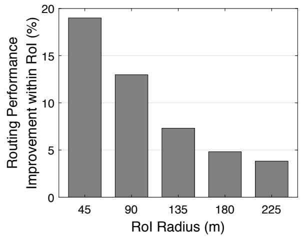
Fig. 5. Dynamic routing hole percentage over time with 5 local UAVs and one global UAV

connected components over the entire RoI, serving as an ideal upper bound of our proposed algorithm. Third, we examine how much routing performance is improved due to each different deployment decision from the local UAV itself or its paired global UAV's recommendation. Lastly, we quantify computation complexity in terms of running time.

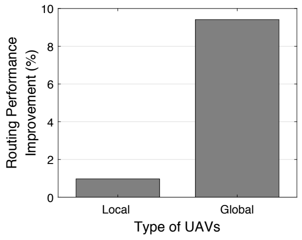
We explore steady-state network recovery performance in terms of network hole percentage as we increase available number of local or global UAVs. We quantify network hole percentage in terms of the number of source-to-destination pairs with no existing route and required time to reach at the steady-state as in Fig. 4. As the number of local UAVs increases, the network hole percentage decreases as shown in Fig. 4(a). Deploying more local UAVs as relays helps to dramatically reduce the number of critical network holes. On the other hand, the steady-state time for deployment completion increases since more local UAVs need to be deployed.

As we use more global UAVs, network hole percentage is not affected since they do not act as relays, but rather provide deployment guidance to local UAVs in the network as in Fig. 4(b). However, using more global UAVs contributes to finishing the total UAV deployment completion earlier by recommending globally optimal deployment positions to local UAVs more frequently.

We investigate dynamic network recovery performance over time in terms of network hole percentage. We run 10 simulations starting from a randomly selected vertex for each round and show one of the results that represents the general case. As in Fig. 5, our algorithm progressively finds a way to improve routing performance over time by utilizing a mixture of local and global deployment optimization. The network hole



(a) Localized routing performance improvement using local greedy deployment with 5 local UAVs



(b) Global routing performance improvement with a local UAV and a global UAV

Fig. 6. Routing performance improvement by local and global deployment

percentage of our work is given by 52.2%, whereas *DroneNet+* leads to 64.5% based on one-shot decision after spending 347 seconds for network traversing and deployment. Specifically, our algorithm starts performing local optimization for all 5 local UAVs at 59 seconds, showing some minor improvement. At 230 seconds, a global optimization for one local UAV by its paired global UAV has been executed, drastically reducing the percentage of source-to-destination pairs with no route from 71.1% to 54.7%. Right after that, the local UAV gets another global deployment recommendation from its paired global UAV again, and leaves the current deployment position for the newly recommended position, making the network temporarily return back to the previous before-deployed status. At 294 seconds, the local UAV starts being deployed at a globally recommended position again. The upper bound of our algorithm, which is a centralized version, *Centralized* drops network hole percentage down to 42.1%. Since our algorithm runs in a purely distributed manner, 10% performance gap can be a reasonable and expected result.

Now we analyze how routing performance is improved by each feature of local or global deployment decision on our algorithm in a more detail. We quantify routing performance improvement in terms of network hole reduction in a local confined area by controlling the RoI radius. Considering source-to-destination pairs within the RoI, local greedy deployment decision by local UAVs improves localized routing performance as in Fig. 6(a). As the RoI area increases, routing performance improvement within localized area becomes mediocre. On the other hand, for the entire network performance, global recommendation-based deployment contributes to routing performance improvement more effectively with a factor of 9.68, compared to local greedy deployment as in Fig. 6(b).

Lastly, we examine computation complexity in terms of running time. As varying the number of local UAVs deployed in the network, we measure execution time for running our algorithm, compared to *DroneNet+*. As can be seen in Fig. 7, our work offers a practically feasible solution even in computation complexity while outperforming routing performance.

VI. Conclusion

We have presented a route reconstruction algorithm based on distributed network diagnosis and progressive relay node placement using UAVs. For the diagnosis of a destructed network, we have used two types of UAVs, global UAVs traverse with a long stride and probe high-level information,

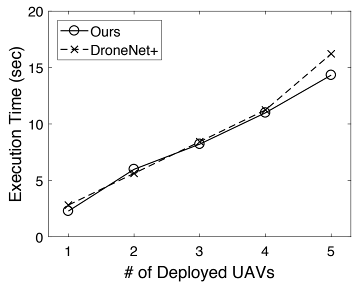


Fig. 7. Computation complexity in terms of running time with respect to the number of deployed UAVs

while local UAVs traverse with a short stride and collect information in detail. During their fully distributed traversing with their best effort, encountered global and local UAVs can make a pair and find progressively more optimized deployment places in a collaborative way. Global UAVs capture high-level network topology and recommend better places to their paired local UAVs. Local UAVs iteratively change their deployment place with or without the help of global UAVs, reconstructing end-to-end routing path in a progressive manner.

We have demonstrated that our algorithm effectively reconstructs network connectivity in terms of steady-state and dynamic routing performance. Also, our experiment results show that both local and global deployment decisions play a synergistic role in minimizing network hole percentage and improving end-to-end route repair.

For future work, we may relax designated roles for global and local UAVs so that a general UAV can act as a local or a global UAV from time to time depending on dynamic network status. Also, it would be interesting to consider external limitation such as battery outages or UAV failures for designing a more practical algorithm.

References

[1] X. Cheng, D.-Z. Du, L. Wang, and B. Xu. Relay sensor placement in wireless sensor networks. *Wirel. Netw.*, 14(3):347–355, June 2008.

[2] T. H. Cormen. *Introduction to algorithms*. MIT press, 2009.

[3] Z. Han, A. L. Swindlehurst, and K. J. R. Liu. Smart deployment/movement of unmanned air vehicle to improve connectivity in MANET. In *IEEE WCNC*, 2006.

[4] D. Jeong, S. Y. Park, and H. Lee. DroneNet: Network reconstruction through sparse connectivity probing using distributed UAVs. In *IEEE PIMRC*, pages 1797–1802, Aug 2015.

[5] S. Lee and M. Younis. Optimized relay node placement for connecting disjoint wireless sensor networks. *Computer Networks*, 56(12):2788–2804, 2012.

[6] E. L. Lloyd and G. Xue. Relay node placement in wireless sensor networks. *IEEE Transactions on Computers*, 56(1):134–138, Jan 2007.

[7] 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 *2017 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2017.

[8] F. Senel and M. Younis. Relay node placement in structurally damaged wireless sensor networks via triangular steiner tree approximation. *Computer Communications*, 34(16):1932–1941, 2011.

[9] I. F. Senturk, K. Akkaya, and S. Yilmaz. Relay placement for restoring connectivity in partitioned wireless sensor networks under limited information. *Ad Hoc Networks*, 13:487–503, 2014.

[10] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[11] G. Tuna, B. Nefzi, and G. Conte. Unmanned aerial vehicle-aided communications system for disaster recovery. *Journal of Network and Computer Applications*, 41:27–36, 2014.

[12] J. Yoon, Y. Jin, N. Batsoyol, and H. Lee. Adaptive path planning of uavs for delivering delay-sensitive information to ad-hoc nodes. In *IEEE Wireless Communications and Networking Conference (WCNC)*, 2017.