DroneNet+: Adaptive Route Recovery Using Path Stitching of UAVs in Ad-Hoc Networks

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

***Abstract*—In this paper, we consider a route recovery problem using Unmanned Aerial Vehicles (UAVs) as relay nodes to connect with terrestrial ad-hoc networks in realistic disaster scenarios. Our main goal is to perform network probing from the air by UAVs and find out crucial spots where both local and global routing performance can significantly be recovered if they are deployed. We propose a route topology discovery scheme that extracts the inherent route skeletons by *stitching* partial local paths obtained from simple packet probing by UAVs, while exploring a designated Region of Interest (RoI) by an adaptive traversing scheme. By leveraging the captured topology, we dispatch a limited number of UAVs by an iterative UAV deployment algorithm and provide a lightweight yet effective network hole replacement decision in a heuristic manner. Simulation results demonstrate that our traversing algorithm reduces the complete coverage time, the travel distance, and the duplicate coverage compared to a previous work, *DroneNet*. Our subsequent iterative deployment algorithm greatly recovers severely impaired routes in a damaged network, while substantially reducing computational complexity.**

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 [5], [8], [12]. 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, employing UAVs can be a rescue to address the network hole problem by being deployed as temporary relay nodes [1], [2].

Regarding the network coverage of agents such as vehicles or robots, researchers in robotics have extensively investigated the problem of path planning and space exploration [3], [14], [16]. Although most of them aim to mitigate duplicate coverage among agents, they can not directly be applied to the UAV context because there is no consideration of networking capability and lightweight computation.

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).

There have been several efforts to address the network hole problem in an alternative way by repairing a damaged network with UAVs [6], [7], [13]. 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 network connectivity by utilizing UAVs based on Delaunay triangulation [6] or a game theoretic approach [13]. However, both approaches are hard to be applied to dynamically changing networks because of their high computational complexity. Our previous work [7] frames the network reconstruction as two-phase problems of network probing and UAV deployment. After collecting empirical link connectivity information over the terrestrial network, UAVs are determined to be dispatched into some link-impaired regions. Although this approach finds effective spots to repair local network connectivity, it does not capture non-local routing dynamics, also suffering from computation complexity for solving an optimization problem.

In this paper, we consider both local and *non*-local network connectivity using UAVs to accomplish a more suitable UAV deployment in terms of route reconstruction. Based on a newly designed cost effective motion planning algorithm, UAVs drop off probing packets that keep being relayed within several hops and retrieve them to parse their distinctive partial 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 [9].

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 most effective deployment locations, leading to significant route improvement over the damaged network.

Our main contributions can be summarized as:

- We design an adaptive UAV network traversing algorithm that outperforms existing previous algorithms in terms of traveling time and distance, and duplicate coverage.
- We present a route topology discovery scheme to extract the inherent route skeletons by stitching partial local paths obtained from simple path probing by UAVs.
- We propose a lightweight network hole replacement

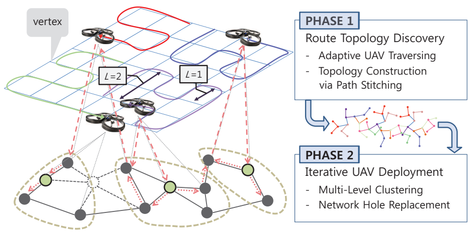


Fig. 1. Overall procedure of our proposed algorithm

algorithm that dispatches a limited number of UAVs to the crucial spots which can achieve both local and global improvement of routing performance.

The remainder of this paper is organized as follows. After introducing our problem and system model in Sec. II, we present our route topology discovery scheme with motion planning in Sec. III and our UAV deployment algorithm in Sec. IV. After presenting the evaluation results of our proposed approach in Sec. V, we conclude this paper in Sec. VI.

II. System Model

We consider a route recovery problem using UAVs such that they are deployed as relays to be connected with terrestrial ad-hoc networks in realistic disaster scenarios. The ad-hoc networks are supposed to support any to any communication. Our main objective is to perform network probing from the air by UAVs and find out crucial spots where both local and global routing performance can greatly be repaired if deployed.

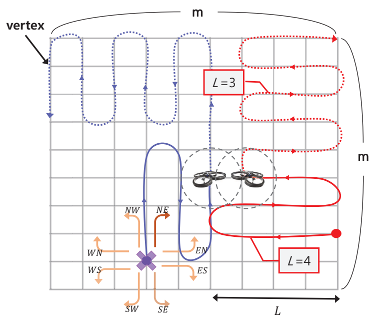
We assume that a UAV can communicate with terrestrial ad-hoc networks and other UAVs using a designated wireless interface such as 802.11 or 802.15.4 within its radio range, while all of ad-hoc nodes and UAVs use the same transmit power. It is also assumed that UAVs are equipped with the Global Positioning System (GPS), and free to move to a certain position without physical restrictions or obstacle collisions.

The route recovery problem can be divided into two sub-problems: 1) route topology discovery from the air through network traversing of UAVs, and 2) lightweight UAV deployment as additional network relays to efficiently cope with network breakdown in disaster scenarios.

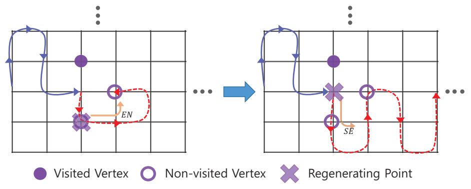
After all of UAVs finish network exploration over a given Region of Interest (RoI), they gather at a designated place to share the collected network probing information. Based on the constructed route topology, their deployment location is computed, and UAVs are accordingly dispatched to be used as relays as illustrated in Fig. 1.

III. Route Topology Discovery

When a catastrophic disaster occurs, a terrestrial network of stationary ad-hoc nodes may be damaged severely. To maintain reliable route paths over stationary ad-hoc networks, preserving local wireless connectivity to neighboring nodes is essential [7]. 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



(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. 2. Adaptive UAV traversing with an effective navigation decision

important to diagnose the route status on the damaged network by discovering its global route topology beyond the local one.

In this section, 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*.

A. Adaptive UAV Traversing

Each UAV explores a designated RoI where its logical grid coordinate consists of $m \times m$ vertexes as in Fig. 2(a). It follows an independent motion planning based on one of eight predefined *zigzag* patterns, e.g., North-East, North-West, South-West, East-North, East-South, West-North, and West-South with the orthogonal traversal width L.

Each UAV performs its navigation by generating one zigzag trajectory out of eight patterns with the longest path toward its moving direction up to either the boundary of RoI or an already-visited vertex. To keep track of visited vertexes, it records the visiting vertex ID in its *vertex-visit-list*. If a UAV becomes connected with another UAV within the radio range, they exchange their own *vertex-visit-list* each other and merge them to avoid duplicate coverage on its future navigation trajectory. Since this work follows similar high-level motion planning taken from our previous work, please refer to *DroneNet* [7] for more detailed information.

DroneNet has a drawback of using the fixed L regardless of on-going navigation progress with other UAVs. This work 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

Algorithm 1 Adaptive Multi-UAV Network Traversing

```
1: Input: CurrentVertexID
2: Output: NextVertexID
   // Part I: Motion planning
3: if (future-vertex-visit-trajectory == ∅) or (future-vertex-visit-trajectory's
   next vertex is taken or null) then
4:   if (any unvisited neighboring vertex in North, East, South, West)
     then
5:     Regenerate the future-vertex-visit-trajectory
          with the longest length that starts from an unvisited vertex;
6:     NextVertexID = future-vertex-visit-trajectory's first vertex ID;
7:     Move with one step to the next vertex;
8:   else
9:     if (there exists any unvisited vertex) then
10:      NextVertexID = the nearest vertex's ID on the grid coordinate
         from CurrentVertexID;
11:      Invoke path-probing();
12:      Fly to the next vertex;
13:    else
14:      Terminate;
15:    end if
16:  end if
17: else
18:  NextVertexID = future-vertex-visit-trajectory's next vertex ID;
19:  Invoke path-probing();
20:  Move with one step to the next vertex;
21: end if
   // Part II: Path probing
22: Function path-probing()
23:   Broadcast a path-probing packet;
24:   Any neighboring stationary nodes keep relaying the probing packet
      up to $n$ hops, while recording a series of relay node ID in the header;
25:   Receive completed path-probing packets stored at the currently
      visiting node initiated by itself or other UAVs;
26:   if (any UAVs within radio range) then
27:     Exchange vertex-visit-list and update it;
28:     if (any duplicated visited vertexes) then
29:       Decrement the navigation width $L$ by 1;
30:       future-vertex-visit-trajectory = ∅;
31:     end if
32:   end if
33: EndFunction
```

range. In this case, it decrements the navigation width L by 1 so that it can lessen potential duplicate coverage on its future movement progress as depicted in Fig. 2(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. 2(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. 2(b)).

This adaptive UAV traversing scheme *DroneNet+* greatly advances the previous *DroneNet* in motion planning efficiency by lowering duplicate coverage and travel distance until all the vertexes are completely covered by UAVs.

B. 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*.

1) Probing partial local path by relaying over a few hops: When a UAV visits a vertex during network traversing, it

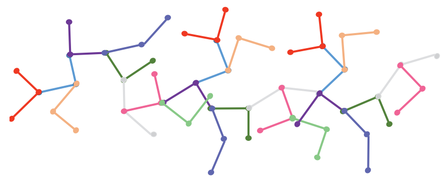


Fig. 3. Topology discovery by stitching partial local route paths (over two hops) via *path stitching*

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.

2) 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. 3.

IV. Iterative UAV Deployment

In this section, we present an iterative UAV deployment algorithm that improves routing performance over the damaged networks through a heuristic approach. We locate and prioritize network holes based on the captured route topology in Sec. III and deploy UAVs as relays to connect with terrestrial networks.

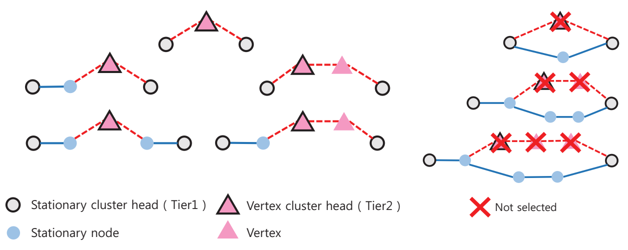
To understand the inherent global routing structure over the networks, it is necessary to find out crucial *skeleton* nodes that connect not only local neighboring nodes but rather other neighboring sub-networks. To extract those skeleton nodes in the networks, we incorporate a connectivity-based clustering 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 candidate 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.

A. Connectivity-based k-hop Clustering

We present a simple yet efficient connectivity-based k-hop clustering method performed in a centralized manner similar to [4], [11]. Using a constructed route topology, we count the


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. 4. Iterative clustering and deployment decision procedure

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.

B. 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.

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. 4(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

Algorithm 2 Deployment of UAV Relays at Network Holes

```
 1: Input: Route topology & # of available UAVs for relay deployment
 2: Output: selectedVertexesForUAVs
 3: selectedVertexesForUAVs = ∅;
    // Deployment case 1 to 3: multi-level clustering & deployment
 4: deploymentCase = 1;
 5: while (deploymentCase <= 3)
        // 1st-tier clustering for all stationary nodes using $k$ hops
        // return stationary cluster heads
 6:     sClusterHeads = connectivity-clustering(stationaryNodes);
        // 2nd-tier clustering for stationary cluster heads and all vertexes
        // using deploymentCase hops
        // return vertex cluster heads
 7:     vClusterHeads = connectivity-clustering(sClusterHeads ∪ vertexes,
                                                deploymentCase hops);
 8:     Find parts of vClusterHeads connecting two sClusterHeads through;
 9:     Exclude ones with any existing route within deploymentCase hops;
10:     Prioritize all possible vertex candidates in terms of route effectiveness;
11:     Deploy all possible UAVs to the prioritized vertexes;
12:     Update selectedVertexsForUAVs with them;
13:     vertexes ← vertexes − selectedVertexsForUAVs;
14:     stationaryNodes ← stationaryNodes ∪ selectedVertexesForUAVs;
15:     deploymentCase++;
16: endwhile
17: if (any UAVs still left) then
18:     Deploy remaining UAVs to vertexes with the highest # of neighbors
          within $k$ hops;
19:     Update selectedVertexesForUAVs with them;
20: end if
```

candidates as illustrated in Fig. 4(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.

This iterative algorithm provides a lightweight yet effective deployment decision for multiple UAVs, contributing to significant improvement in routing performance.

V. Evaluation

We evaluate our adaptive route recovery algorithm based on path stitching of UAVs, named as *DroneNet+* in a network of 64 stationary nodes over the RoI of 144 $\times$ 144 m^2 as in Fig. 5. 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 [10] with `meyer-light` noise traces.

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.

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+* in terms of complete coverage time, travel distance, and duplicate coverage rate by varying the number of UAVs compared to *Ants* [14], *DroneNet* [7], and a centralized optimal solution that solves the Multiple Traveling Salesman Problem, *mTSP* [15]. Second, we investigate network repair performance of *DroneNet+* in terms of end-to-end routing cost and source-to-destination pairs with no route, as opposed to *DroneNet* and an upper-bound algorithm. We quantify the computation complexity of our iterative deployment algorithm 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. Adaptive UAV Traversing

We explore the efficiency of our adaptive UAV traversing algorithm, *DroneNet+* against our previous *DroneNet*, which does not reflect dynamic coverage progress of other UAVs by using a fixed navigation width. The flying speed of UAVs is assumed to be 11.1 m/s (as per Parrot AR.Drone 2.0). 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 in *DroneNet+*. We also quantify the duplicate

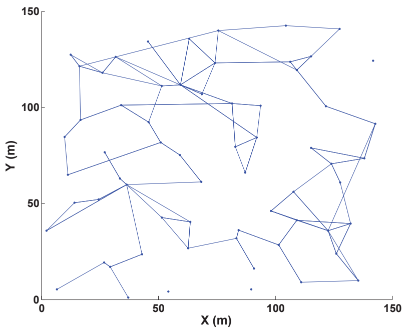
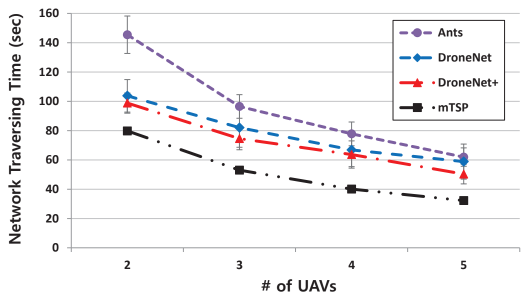
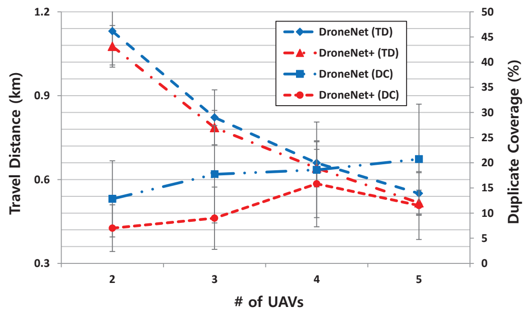


Fig. 5. Network topology of 64 sensor nodes over RoI in a simulated network, having almost half source-to-destination route pairs with no existing path (where good communication links are shown for PRR $\geq$ 75%)



(a) Complete coverage time per UAV for network traversing



(b) Travel distance and duplicate coverage rate per UAV

Fig. 6. Network exploration performance comparison with respect to the number of UAVs with the error bars of standard deviation

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, *DroneNet+* outperforms *Ants* that does not share visited vertex information with others. Both *DroneNet* and *DroneNet+* reduce the network traversing time spent for the complete coverage over RoI in Fig. 6(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*. This means that the adaptive control of navigation width L 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 a practical algorithm, we find a theoretical limit of traversing time by solving an *mTSP*, which offers an optimal solution in a centralized manner.

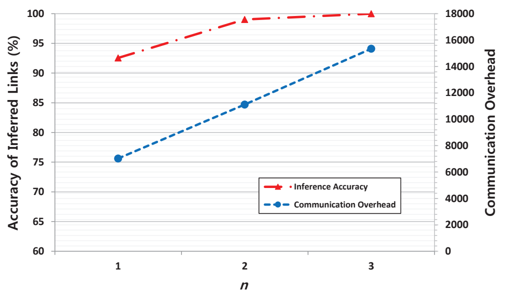


Fig. 7. Valid route inference performance and communication overhead of broadcast control packets

Although our algorithm is a heuristic distributed one, its resulting performance has a relatively similar tendency with a theoretical bound of *mTSP* as the number of UAVs increases.

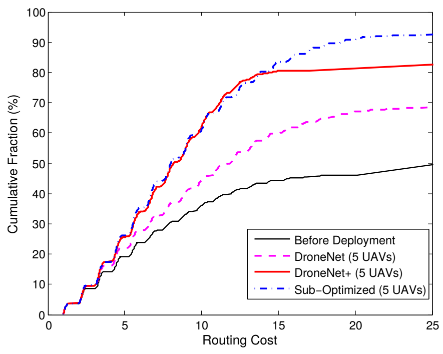
We measure travel distance and duplicate coverage in Fig. 6(b). Although *DroneNet+* has a slightly lower travel distance than *DroneNet*, both motion planning algorithms reduce travel distance as the number of UAVs increases and show the similar performance on navigation efficiency in the space domain. As for duplicate coverage, *DroneNet+* reduces the duplicate coverage up to 49.3% compared to *DroneNet* thanks to the adaptive navigation control. This implies that even intermittent duplicate coverage status that can be only shared with other UAVs within communication range should be considered as a valuable information to change the navigation pattern of UAVs.

B. Network Recovery

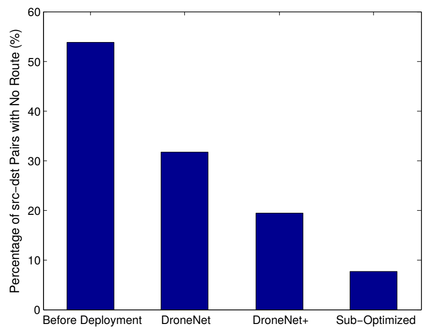
We investigate network recovery performance of our algorithm to measure how the selection of UAV deployment positions reduces the number of network holes and improves routing cost compared to other counterpart algorithms.

We compare *DroneNet+* with *DroneNet* and an upper bound algorithm. *DroneNet* [7] deploys UAVs to the locations where the local connectivity with neighboring stationary nodes is the worst by considering all the possible combinations of deployment candidate positions through an optimization methodology. 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.

In our experiments, routing cost is measured as the sum of the expected number of transmissions over routing hops. As the maximum hop distance of relaying path-probing packet, n increases, the accuracy of correctly inferred links among ground-truth links 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 92.55% of links are correctly inferred, as demonstrated in Fig. 7. The parameters of $k = 1$ on connectivity-based clustering and $\lambda = 1$ on network hole replacement are tuned to be used in our simulations. We show the average performance over 10 independent simulations.



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



(b) Routing hole percentage of *DroneNet+* with *DroneNet* and *Sub-Optimized* algorithms (for 5 UAVs)

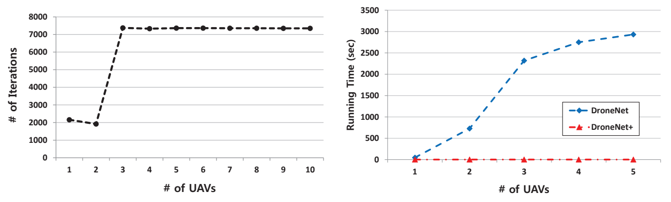
Fig. 8. Network recovery performance comparison with other counterpart algorithms in terms of the end-to-end routing cost

We compare *DroneNet+* against *DroneNet* and a *Sub-Optimized* upper bound algorithm in a damaged network where 53.8% of source-to-destination routing pairs have no existing path in Fig. 8. As the cumulative distributions of routing cost are shown in Fig. 8(a), *DroneNet+* outperforms *DroneNet*, while having the lower routing cost. Also, our *DroneNet+* based on a relatively lightweight iterative approach works closely to the *Sub-Optimized* upper bound algorithm that suffers from high computation complexity. As measured in Fig. 8(b), *DroneNet* leads to the network hole percentage of 31.7%, whereas *DroneNet+* further reduces the percentage down to 19.3%, with a factor of 1.64.

C. Computation Complexity

We measure computation complexity in terms of two metrics of the number of computation iterations and running time in Fig. 9. As the number of UAVs increases beyond two UAV deployment, our algorithm computes desirable deployment decisions within 7500 iterations in Fig. 9(a). This means that our iterative deployment algorithm is scalable even with a larger number of UAVs for the deployment problem.

We quantify running time and compare *DroneNet+* with *DroneNet*. As in Fig. 9(b), our *DroneNet+* spends a little time to compute the solution within seconds, whereas *DroneNet* takes much more time to solve its optimization problem. This implies that our algorithm provides a lightweight practical approach, making it feasible with a larger number of UAVs.



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

Fig. 9. Computation complexity in terms of # of computation iterations and running time

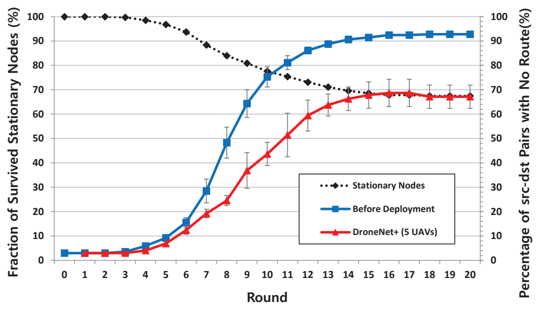


Fig. 10. Dynamic network recovery performance as stationary nodes become dying out over time

D. Dynamic Performance

We examine dynamic network recovery performance of *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 $\times$ 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. 10, as the number of active stationary nodes even gradually decreases, the network gets dramatically disconnected, significantly breaking down existing routes. If *DroneNet+* is allowed to apply its adaptive route recovery procedure using 5 UAVs at each round, the speed of the network breakdown becomes much slower. Although the effective number of nodes still decreases even after deploying 5 UAV relays, our adaptive UAV deployment keeps reorganizing their effective deployment positions at each round, avoiding substantial route outages as much as possible.

VI. Conclusion

We have presented an adaptive route recovery algorithm based on topology discovery and network hole replacement with UAV relays. To extract the global route topology of a terrestrial ad-hoc network in post-disaster scenarios, we let UAVs traverse the network according to their own distributed motion planning, while stitching partial local paths collected from route probing information by UAVs.

We have incorporated a computationally lightweight UAV deployment algorithm to replace network holes with UAV relays. To capture the inherent global route connectivity throughout the network, we have applied a connectivity-based clustering algorithm to terrestrial nodes and all possible deployment vertex candidates. We iteratively deploy a UAV or several UAVs at some selected vertex location that would lead to the most significant routing enhancement after its deployment.

Our experiments demonstrate that our scheme has significantly improved routing performance and computation complexity by exploiting the efficiency in motion planning and UAV deployment compared to a baseline counterpart and a sub-optimal brute-force algorithm.

For future work, we may interleave the route topology discovery with a provisional UAV deployment, achieving higher efficiency and suppressing unnecessary discovery. It would be interesting to consider more practical aspects such as battery recharging of UAVs to reflect in the motion planning of UAVs. Also, we could validate our algorithms in a real-world testbed consisting of terrestrial sensors and aerial drones.

References

[1] M. Erdelj, M. Krl, and E. Natalizio. Wireless sensor networks and multi-UAV systems for natural disaster management. *Computer Networks*, 124:72 – 86, 2017.

[2] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz. Help from the sky: Leveraging UAVs for disaster management. *IEEE Pervasive Computing*, 16:24–32, Jan 2017.

[3] E. Ferranti, N. Trigoni, and M. Levene. Brick& mortar: an on-line multi-agent exploration algorithm. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 761–767, April 2007.

[4] M. Gerla and J. T.-C. Tsai. Multicluster, mobile, multimedia radio network. *Wirel. Netw.*, 1(3):255–265, Aug. 1995.

[5] L. Gupta, R. Jain, and G. Vaszkun. Survey of important issues in UAV communication networks. *IEEE Communications Surveys Tutorials*, 18(2):1123–1152, Secondquarter 2016.

[6] Z. Han, A. L. Swindlehurst, and K. J. R. Liu. Optimization of manet connectivity via smart deployment/movement of unmanned air vehicles. *IEEE Transactions on Vehicular Technology*, 58(7):3533–3546, Sept 2009.

[7] 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.

[8] S. Kandeepan, K. Gomez, T. Rasheed, and L. Reynaud. Energy efficient cooperative strategies in hybrid aerial-terrestrial networks for emergencies. In *IEEE PIMRC*, pages 294–299, Sept 2011.

[9] D. Lee, K. Jang, C. Lee, G. Iannaccone, and S. Moon. Path stitching: Internet-wide path and delay estimation from existing measurements. In *IEEE INFOCOM*, pages 1–5, March 2010.

[10] H. Lee, A. Cerpa, and P. Levis. Improving wireless simulation through noise modeling. In *2007 6th International Symposium on Information Processing in Sensor Networks*, pages 21–30, April 2007.

[11] F. G. Nocetti, J. S. Gonzalez, and I. Stojmenovic. Connectivity based k-hop clustering in wireless networks. *Telecommunication Systems*, 22(1):205–220, 2003.

[12] I. Rubin and R. Zhang. Placement of UAVs as communication relays aiding mobile ad hoc wireless networks. In *IEEE MILCOM*, Oct 2007.

[13] 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, Part B:487 – 503, 2014.

[14] J. Svennebring and S. Koenig. Building terrain-covering ant robots: A feasibility study. *Autonomous Robots*, 16(3):313–332, 2004.

[15] L. Tang, J. Liu, A. Rong, and Z. Yang. A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex. *European Journal of Operational Research*, 124(2):267–282, 2000.

[16] A. Yazici, G. Kirlik, O. Parlaktuna, and A. Sipahioglu. A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints. *IEEE Transactions on Cybernetics*, 44:305–314, March 2014.