2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Dynamic UGV-UAV Cooperative Path Planning in Uncertain Environments

Ninh Nguyen and Srinivas Akella

Abstract— **This paper addresses the Dynamic UGV-UAV Cooperative Path Planning (DUCPP) problem involving one unmanned ground vehicle (UGV) assisted by one or more unmanned aerial vehicles (UAVs) operating on an uncertain road network with potentially impassable edges. DUCPP is particularly relevant for scenarios such as disaster response, emergency supply transport, and rescue operations, where a UGV must reach a specified destination in the presence of partially unknown road conditions. To enable the UGV to travel safely and efficiently to its destination, the UAV(s) dynamically inspect edges in the environment to identify and prune damaged or impassable edges from consideration.**

We present multiple strategies, including a bidirectional approach, to optimize UGV-UAV cooperation for finding a safe path in an uncertain road network. Furthermore, we explore the impact of using multiple UAVs on reducing the UGV's travel time, and evaluate the associated computation time. The proposed strategies are implemented and evaluated on 100 urban road networks. The results demonstrate that the bidirectional strategy achieves the best performance in most instances, and using multiple UAVs further reduces UGV travel time at the expense of increased computation time. This paper presents a robust framework for DUCPP to achieve efficient UGV-UAV cooperation for path planning and inspection, offering practical solutions for navigation in challenging and uncertain conditions.

I. Introduction

This paper presents autonomous planners for an unmanned ground vehicle (UGV) aided by unmanned aerial vehicles (UAVs) to identify safe paths through road networks obstructed or damaged due to natural disasters (e.g., floods, earthquakes, wildfires). The planners can also be used to suggest routes to human drivers. The goal is to perform path inspection to find safe and efficient routes for first responders, people seeking safe locations, and for trucks carrying emergency supplies (e.g., food, medicines) by adaptively updating paths in response to online sensor information.

Since blocked or impassable paths can cause UGVs and even manned vehicles to experience severe delays, it is critical to develop techniques for efficient path finding through partially uncertain road networks whose edge statuses are gradually determined. To achieve this, we can use the complementary abilities of unmanned aerial vehicles (UAVs). UAVs can perform aerial inspections to identify impassable paths and provide real-time updates to the UGV, enabling dynamic rerouting and safe path finding. This cooperation reduces delays and improves operational efficiency. While progress has recently been made in utilizing a single UAV to assist a UGV, incorporating multiple UAVs provides an opportunity to accelerate inspections and improve overall performance. However, this introduces additional challenges in coordination and edge inspection assignment.

This work was supported in part by UNC Charlotte. The authors are with the Department of Computer Science, University of North Carolina at Charlotte, NC 28223, USA. Email: {tnguy248, sakella}@charlotte.edu.

A. Problem Context and Motivation

In real-world applications such as disaster response, a UGV (or manned vehicle) must navigate through a road network represented as a graph to reach a target destination. UAVs can be deployed to inspect edges of this graph to identify damaged or impassable edges. This information is used by the UGV to dynamically adjust its route and avoid delays. The cooperation between the UGV and UAVs is particularly valuable in scenarios where road conditions are uncertain and impassable edges may disrupt UGV operations.

The key challenge lies in effectively coordinating UAV edge inspections to maximize their utility for UGV path finding. UAVs must prioritize edges based on their potential impact on the UGV's path. In settings with multiple UAVs, efficient task allocation is required to avoid redundant inspections and minimize computational overhead. Addressing these challenges is essential for achieving effective UGV-UAV cooperation and efficient and safe UGV path finding.

B. Contributions

This paper addresses the Dynamic UGV-UAV Cooperative Path Planning (DUCPP) problem, which focuses on using UAVs to enable efficient and safe path finding for a UGV in a graph-based environment with uncertain and potentially impassable edges. Our main contributions are:

- *Problem formulation.* We formalize DUCPP to allow impassable edges whose status is revealed only when vehicles reach the damaged point, and with UAV motion that can follow graph edges for inspection or deadhead (i.e., fly directly) between vertices.
- *Replanning strategies.* We develop several inspection and replanning strategies, including a bidirectional algorithm, and show how UAV edge inspection choices can reduce the UGV's travel time.
- *Multiple-UAV extension.* We extend the bidirectional strategy to multiple UAVs and evaluate the reduced UGV travel time and increased computation time.
- *Evaluation.* We perform extensive benchmarking across diverse road network instances, demonstrating significant improvements in efficiency and effectiveness over baseline approaches.

C. Related Work

Prior work has addressed replanning routes for single robots on networks with unknown obstacles [1], [2], inspecting static road networks using one or more UAVs [3], [4], and using UAVs to gather traffic flow information to dynamically improve resource transportation [5]. UGV-UAV cooperation has been widely studied for environmental monitoring, disaster response, and agriculture [6], [7].

1) Canadian Traveller Problem (CTP) and Variants: When planning optimal paths for a vehicle in uncertain road environments, the Canadian Traveller Problem (CTP) [2] has been used to model routing tasks where the traversability of edges is initially unknown and gradually discovered as the vehicle traverses the road network. It is PSPACE-complete to achieve a bounded competitive ratio for the CTP, and the stochastic version has been shown to be #P-hard. The CTP assumes that at a vertex, the vehicle can sense the traversability status of all incident edges. In contrast, our DUCPP problem assumes that the robot (UAV or UGV) only learns about the condition of the road when it reaches the *damaged point*, which could be in the middle of a long road segment. This realistic modeling in DUCPP reflects the challenges faced in real-world uncertain environments where costs and road conditions are often unknown until encountered.

Variants of the CTP, such as k-CTP, have explored problems involving multiple paths, but typically focus on a single UGV [8], [9], [10]. CTP with remote sensing [10] assumes a sensor can query the state of any edge for a known cost; in DUCPP, the UAV requires path planning and does not have a fixed cost. Our work extends this idea by proposing cooperative strategies where UAVs assist a UGV in dynamically selecting candidate paths to reach its goal in the shortest time, with the added complexity of planning for multiple agents.

2) UGV-UAV Coordination on Road Networks: Much work on UGV-UAV coordination has focused on package delivery using drones and trucks [11], [12], and assumes known road networks. Recent work [13] has studied truck and drone delivery in road networks damaged by natural disasters. It focuses on competitive ratio analysis of policies where either the truck or the drone can complete the delivery. It does not provide an explicit edge inspection/replanning strategy to minimize the UGV's travel time or benchmark the policies on realistic road-network instances.

The most closely related research in UGV-UAV coordination for road networks assumes passable edges with varying costs [14], [15] rather than scenarios where edges may be entirely impassable. In [14], they proposed a framework for UGV-UAV assisted path planning in stochastic networks, where UAVs inspect edges along the UGV's k-shortest paths to identify delays. Their model assumes that all edges remain passable but may incur higher traversal costs depending on road conditions (e.g., congestion). They model edge cost variability, with upper and lower limits on the edge costs. However, it differs from our problem in crucial ways. First, once the UGV decides to traverse an edge, it cannot reverse direction. Second, their approach relies on the UGV's expected travel time to compute the shortest path before any impeded edges are inspected. If the cost upper limit is set to infinity to model a damaged edge, the UGV fails to find a viable path since the expected travel time is infinite; using a large constant biases it to selection of paths with fewer impeded edges even though the paths may be longer. In contrast, in DUCPP, the UGV reroutes dynamically if its current edge is damaged — it returns to the previous vertex and replans its path. In [15], they studied assistive routing where a support vehicle can clear impeded road segments to assist a primary vehicle, with an attendant cost.

UAV-assisted UGV routing has been explored [5] in flood scenarios, where UAVs inspect roads to identify flooded or blocked conditions. Their approach optimizes traffic flow for multiple UGVs but does not address real-time task reallocation for UAVs or dynamically prioritize inspections based on UGV needs.

A few papers [16], [17], [18] have explored vision-based strategies for UAVs to assist UGVs in navigation. The UAVs capture aerial images of the environment, detect obstacles, generate and update maps in real-time, and continuously track the UGV's position to guide it toward the destination. In contrast, our paper assumes a road network with uncertain connectivity and focuses on prioritizing critical edges for UAV inspection.

As described above, prior UGV-UAV work on road networks has largely emphasized edge cost uncertainty (e.g., delays) rather than connectivity uncertainty due to impassable edges discovered during execution. This paper moves toward that setting by introducing strategies that explicitly handle impassable edges and dynamically prioritize inspections to reduce UGV detours. Additionally, it extends these strategies to multiple UAVs, providing scalable solutions to balance inspection efficiency with computational cost.

D. Organization

The remainder of this paper is organized as follows. Section II formally states the problem and defines the key concepts. Section III describes the proposed UGV-UAV coordination strategies and algorithms. Section IV details the simulation experiments, evaluation metrics, and performance of the strategies. Section V addresses the practical limitations of our current approach. Finally, Section VI summarizes the paper and discusses future research directions.

II. Problem Statement

We now state the *Dynamic UGV-UAV Cooperative Path Planning (DUCPP)* problem and introduce the relevant notation. Let $G = (V, E)$ be a connected, undirected graph, where V represents the set of vertices, and E is the set of edges connecting these vertices. The graph models the road network the UGV must traverse to reach its destination, with potential obstacles or damage on some edges.

The UGV and UAV start at vertices s_g and s_a respectively, where $s_g, s_a \in V$ and s_g and s_a need not be identical. The UGV is tasked with reaching its selected destination vertex

$d \in V$. In contrast, the UAV has no predefined destination and may terminate its journey at any vertex. Let v_g and v_a denote the speeds of the UGV and UAV, respectively. Each edge $e \in E$ has a length c_e, representing the distance of the edge. The time required for the UGV and UAV to traverse edge e is given by c_e/v_g and c_e/v_a, respectively. For simplicity, we use *cost* and *time* interchangeably.

The UGV is constrained to move along the edges of the graph G. The UAV can move along an edge of G to inspect it, or it can *deadhead* from its current position directly to any vertex $v \in V$ in straight-line flight, with travel time given by the Euclidean distance divided by its speed. For example, suppose the UGV's current path is blocked (and detected by either UGV or UAV). In this case, a new path is computed for the UGV, and the UAV may be reassigned to fly directly to a vertex of an uninspected edge of that path to inspect. This formulation captures the UAV's key advantage: its ability to move flexibly through free space to support the UGV's navigation task.

An edge is initially labeled *uninspected*. After traversal by a UAV or UGV, the edge is labeled as *safe* if it is passable or *damaged* if it is impassable. We define a *damaged edge* or an *impassable edge* as an edge with damage or an obstacle so that the UGV cannot traverse it. Let $D \subseteq E$ be the set of damaged edges in the graph that the UGV cannot traverse. The set of damaged edges is initially unknown and can only be determined gradually when either a UAV or the UGV reaches a damaged point or obstacle on an edge. If the UGV encounters a damaged edge during traversal, it must return to the last visited vertex and compute a new path. In such cases, the damaged edge is removed from the graph.

Objective: The objective is to minimize the total travel time C/v_g for the UGV to reach its destination vertex d, where C represents the total travel distance of the UGV. If no path exists from the starting vertex s_g to the destination vertex d due to damaged edges, then $C = \infty$. This paper proposes strategies to enable the UGV to reach its destination in the shortest possible time by leveraging UAV inspections to identify and eliminate damaged edges effectively.

We make the following assumptions to isolate the *planning* component of DUCPP: (i) all vehicles start simultaneously, (ii) edge status updates are communicated reliably and without delay, (iii) replanning time is negligible relative to edge traversal/inspection time, (iv) UAVs can deadhead in straight lines from their position to any vertices, (v) UAVs have no energy constraints, and (vi) once inspected, an edge's status remains constant.

III. Algorithms for DUCPP

Efficient coordinated inspection between the UGV and UAV(s) requires a framework that facilitates communication between the robots, rapid path recomputation in response to sensor information, and effective edge inspection assignment to minimize the total travel time of the UGV. To achieve this, the high-level strategy is divided into the following steps:

1) **Path Planning:** Determine the best possible path for the UGV to reach its destination based on the current state of the graph.
2) **Edge Assignment:** Identify critical edges from the UGV's current path and assign them to the UAV(s) for inspection. We assume each UAV is assigned one edge at a time.
3) **Path Traversal and Edge Inspection:** The UGV and UAV(s) follow their assigned paths and update the *edge status* (safe or damaged) upon completing an edge traversal or encountering an obstacle.

These steps are repeatedly executed whenever the graph is updated, continuing until either the UGV reaches its destination or finds that no feasible path exists.

Algorithm 1 UGV-UAV Coordinated Path Planner

```
1: procedure MAIN(strategy)
2:     totalTime = 0.0
3:     while True do
4:         strategy.find_path()
5:         stepTime, keepGoing=strategy.find_event_and_update()
6:         totalTime += stepTime
7:         if not keepGoing then
8:             break
9:         end if
10:    end while
11: end procedure
```

As outlined in Algorithm 1, communication and coordination between the robots (UGV and UAVs) are used to iteratively achieve the solution. The **Path Planning** and **Edge Assignment** tasks are executed by each strategy at line 4, determining and assigning paths to the robots. Subsequently, line 5 handles the **Path Traversal and Edge Inspection**, which calculates the robot positions at the next event where replanning becomes necessary. An *event* occurs when a robot (UGV or UAV) must stop due to encountering an obstacle or damage on an edge, a UAV completes an edge inspection, or the UGV reaches its destination.

This process enables efficient dynamic replanning and ensures that both the UGV and UAV(s) adapt to the evolving graph conditions. In this paper, we present and evaluate multiple strategies within this framework, evaluating their performance based on the UGV's total travel time and computational time. The modular framework allows ease of implementation and testing of new strategies.

While the UGV and UAV exhibit different behaviors, both share primary functions:

- `find_next_event()`: Executes the assigned path and returns the event when the vehicle stops or completes its task. If the vehicle encounters an obstacle or damage on an edge, it stops, and the affected edge is marked as a damaged edge and removed from the graph. A UAV is considered to have completed its task upon fully inspecting an edge, while the UGV completes its task upon reaching its destination.
- `update_position(event)`: Recalculate the new position of each vehicle at an event; all vehicles need to stop and recalculate their new paths.

A. Abstract Strategy

The main idea for solving the DUCPP problem is to find the best path for the UGV to reach the destination and then analyze that path to assign an edge to inspect to the UAV to support the UGV. The most important parts of the strategy are in two functions:

- `find_path()` finds the path for each robot. This differs for each strategy.
- `find_event_and_update()` returns the elapsed travel time to the next event on the assigned paths. Since `find_event_and_update()` is the same for all strategies, it is implemented in the $AbstractStrategy$ class.

Algorithm 2 Abstract Strategy Class

```
1: procedure FIND_EVENT_AND_UPDATE()
2:     if not ugv.path then
3:         return (0, False)
4:     end if
5:     # Find the next event when a robot has to stop
6:     uavTime = uav.find_next_event()
7:     ugvTime = ugv.find_next_event()
8:     nextEventTime = min(uavTime, ugvTime)
9:     uav.update_position(nextEventTime)
10:    ugv.update_position(nextEventTime)
11:
12:    if the UGV reaches the destination then
13:        return (nextEventTime, False)
14:    end if
15:    return (nextEventTime, True)
16: end procedure
```

The `find_event_and_update` function (Algorithm 2) returns both the next event time and the $keepGoing$ boolean value. As shown in lines 2-4, if no valid path exists from the UGV's current position to the destination, the $keepGoing$ value is set to $False$. Lines 6-10 compute the next event where all robots must stop and recalculate their paths. Since our primary focus in this paper is calculating UGV travel time and computation time, we utilize known information about obstacle positions for this process. However, in a real-world application, this function would need to be modified to dynamically receive event updates from the UGV and UAV during motion and inspection. Despite this difference, the results for travel time and computation time remain consistent between the two versions.

B. Implemented Strategies

1) Perfect knowledge: This strategy establishes a conservative *lower bound* for the problem by assuming the UGV is given complete knowledge of the status of each edge (safe or damaged) in the road network. All damaged edges are removed from the graph, and Dijkstra's algorithm is used to find the shortest path to the destination d. If no safe path exists, the UGV does not move and so its travel time is set to 0.

2) UGV-only: In this simple strategy, which serves as a baseline, we use only a single UGV to find a safe path to the destination. The UGV is always assigned the shortest path (computed using Dijkstra, A*, or D* Lite) from the current position to the destination. Whenever the UGV finds a damaged point on the path, the UGV must return to the last visited vertex and reroute on a new shortest path.

3) UGV-UAV with Kemeny constant: Here we pre-calculate the criticality of each edge in the graph. For this, we use a measure of centrality of an edge e in an undirected graph introduced by [19]. It is a modification of the Kemeny constant of the graph; for each edge, its value is the Kemeny constant after removing the edge. The UGV always follows the shortest path, and the UAV inspects the edge with the highest critical value on the UGV's path (Algorithm 3). Although the modified Kemeny constants should ideally be recomputed every time the graph is updated, for computational reasons, we compute the Kemeny constants only once at the beginning.

Algorithm 3 UGV-UAV with Kemeny constant

```
1: procedure FIND_PATH()
2:     # Find the shortest path and assign it to the UGV
3:     ugv.path = find_shortest_path(ugv.position, d)
4:     if not ugv.path then
5:         return False
6:     end if
7:     # Assign the edge with the highest critical value to the UAV
8:     for edge in ugv.path do
9:         if edge is not inspected and has higher critical value then
10:            uav.path = edge # Edge to be inspected
11:        end if
12:    end for
13: end procedure
```

4) UGV-UAV with k-shortest paths: This strategy (Algorithm 4) uses a different approach to identify the critical edge to be inspected by the UAV. Each time the UGV reroutes, the UAV uses Yen's algorithm [20] to find the k shortest paths from the current UGV position to the destination. It then inspects the edge that appears most frequently in the k shortest paths, provided the edge is also in the UGV's current path.

Algorithm 4 UGV-UAV with k-shortest paths

```
1: procedure FIND_PATH()
2:     # Find the shortest path and assign it to the UGV
3:     ugv.path = find_shortest_path(ugv.position, d)
4:     if not ugv.path then
5:         return False
6:     end if
7:     # Assign the edge that is most repeated in the k-shortest paths
8:     kPaths = find_k_paths(ugv.position, d)
9:     Assign uninspected edge that is most repeated in kPaths to UAV
10: end procedure
```

5) UGV-UAV with MPSP: The Most Probable Shortest Path (MPSP) problem in a graph aims to find the path that not only minimizes the cost but also maximizes the probability of successful traversal. This approach is particularly relevant in uncertain environments where edges have *existence probabilities*, i.e., probabilities representing their reliability or likelihood of being passable. In Algorithm 5, we use the algorithm in [21] to find the MPSP and assign that path to the UGV. Then, the UAV inspects the edge with the lowest existence probability (i.e., least likely to be passable).

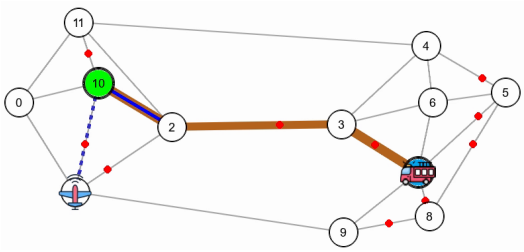
Algorithm 5 UGV-UAV with Most Probable Shortest Path

```
procedure FIND_PATH()
    # Find the MPSP and assign it to the UGV
    ugv.path = find_MPSP_path(ugv.position, d)
    if not ugv.path then
        return False
    end if
    # Assign the edge with the lowest existence probability to the UAV
    for edge in ugv.path do
        if edge not inspected and has lower existence probability then
            uav.path = edge # Edge to be inspected
        end if
    end for
end procedure
```

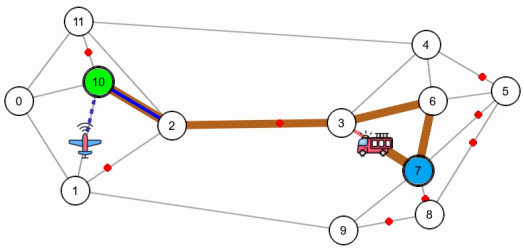
6) UGV-UAV with bidirectional approach: In bidirectional search, one search tree grows from the start node while another grows from the destination node, and a path is found when the two trees meet [22]. For many problems, bidirectional search significantly reduces the required exploration. Variants such as bidirectional Dijkstra and A* provide optimal solutions when positive edge costs or an admissible heuristic function are available. For dynamic networks, [23] proposed advanced bidirectional algorithms like Bidirectional Lifelong Planning A* (BLPA*) and fractional bidirectional D* Lite (fBD* Lite(dp)), which efficiently recompute shortest paths in response to environmental changes. However, these algorithms are designed for single-agent scenarios and do not address task assignment or inspection planning by supporting agents, such as UAVs. This limits their suitability for scenarios requiring multi-agent cooperation to handle dynamic, uncertain environments. Our work extends the bidirectional approach to address these challenges, adapting it for inspection by UAVs in UGV-UAV cooperative scenarios. We compute the shortest path from the UGV's current position to the destination and simultaneously assign the UAV to inspect the reversed path from the destination back toward the UGV (Algorithm 6). This strategy ensures the UAV can quickly inspect edges along the UGV's path without unnecessary UAV transit travel or redundant inspections. If either robot encounters an impassable edge, the affected edge is removed from the graph, and both robots recalculate their respective paths. This process repeats iteratively until either the UGV successfully reaches its destination or no viable path remains.

Algorithm 6 UGV-UAV with Bidirectional Strategy

```
procedure FIND_PATH()
    # Find the shortest path and assign it to the UGV
    ugv.path = find_shortest_path(ugv.position, d)
    if not ugv.path then
        return False
    end if
    # For UAV: Inspect the next uninspected edge on reversed UGV path
    uav.path = None
    for i ← length(ugv.path) − 1 to 1 do
        if edge(ugv.path[i], ugv.path[i − 1]) not inspected then
            uav.path = [ugv.path[i], ugv.path[i − 1]]
            break
        end if
    end for
end procedure
```



(a) Event 1

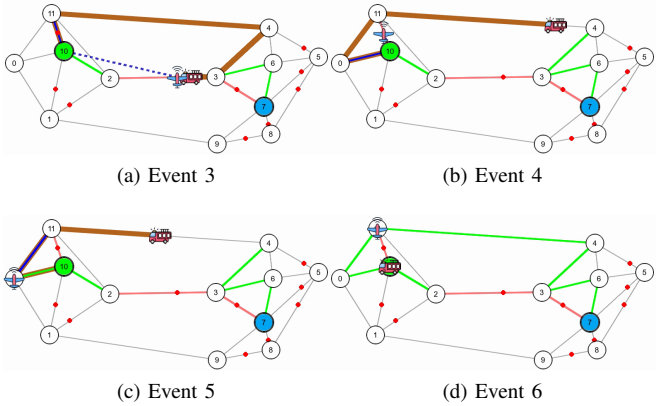


(b) Event 2

Fig. 1. (a) An example road network showing the start positions of the UGV and UAV and the initial shortest path (in brown) from start vertex 7 to the destination vertex 10. The UGV is represented by the firetruck, and the UAV by the plane. (b) The paths and positions of robots at event 2, when the UGV encounters an obstacle. The red dots represent (initially unknown) obstacles or obstructions on edge segments.

Our bidirectional approach emphasizes efficiency in UAV travel. This is in contrast to strategies that assign edges for inspection to the UAV based on edge priority metrics (e.g., criticality or appearance frequency in shortest paths); these may require the UAV to travel long distances to reach high-priority edges. In the bidirectional strategy, the UAV inspects uninspected edges in reverse order along the UGV's shortest path, typically starting with the edges closest to the destination. This reduces UAV travel time significantly and, in many cases, allows the UAV to inspect consecutive edges of the UGV path without interruption, as these edges are usually connected. The bidirectional approach is also more computationally efficient.

We illustrate the bidirectional strategy with a simple example (Figures 1 and 2). In this example, the UAV starts at node 1, the UGV starts at node 7, and the UGV's destination is node 10. At the beginning event (Fig. 1(a)), the UGV's shortest path from its start to the destination is $\langle 7, 3, 2, 10\rangle$ (highlighted in brown). Since the statuses of edges in the map are unknown, the UAV will try to inspect that path in reverse order. Hence, the first edge that the UAV needs to inspect is $(10, 2)$. The robots begin moving after the paths are assigned to them. Because there is a damaged point in the middle of the edge $(7, 3)$, the UGV will reach and find it before the UAV can complete inspecting the edge $(10, 2)$. Event 2 shows the positions of the robots and their planned paths after the UGV finds a damaged point. When any robot finds a damaged point, the graph is updated, and all robots recalculate their new paths. The UGV needs to return to the last safe node, and the UGV's updated path will be $\langle 7, 6, 3, 2, 10\rangle$. The UAV's path is still $(10, 2)$.



(a) Event 3 (b) Event 4

(c) Event 5 (d) Event 6

Fig. 2. The paths and positions of robots at selected events. The UGV arrives at the destination (event 6) after several edges have been inspected by the UGV and UAV. Safe edges are shown in green, damaged edges in red. The final safe path traversed by the UGV is $\langle 7, 6, 3, 4, 11, 0, 10\rangle$.

- **Event 3:** The UAV completed the inspection of the edge $(10, 2)$ and marked it as a safe edge, then continued to inspect the edge $(2, 3)$. The UGV found a damaged point first on edge $(2, 3)$ and stopped; the UAV received the information from the UGV and stopped on edge $(2, 3)$ where it was inspecting it. The next UGV path is $\langle 3, 4, 11, 10\rangle$, and the UAV is assigned edge $(10, 11)$.
- **Event 4:** The UAV found a damaged point on the edge $(10, 11)$ first, while the UGV was traversing the edge $(4, 11)$. The next UGV path is $\langle 11, 0, 10\rangle$, and so the UAV's assigned edge is $(10, 0)$.
- **Event 5:** The UAV completed inspecting the edge $(10, 0)$ when the UGV was still on the edge (4, 11). The UGV's path is $\langle 11, 0, 10\rangle$, and the UAV's edge is $(0, 11)$.
- **Event 6:** The UAV completed inspecting the edge $(0, 11)$. Since all edges in the UGV's current path are safe, the UAV stopped at node 11, and the UGV continues until it reaches its destination.

The example underscores the ability of the bidirectional strategy to dynamically adapt to graph updates while efficiently coordinating the UGV and UAV.

7) UGV and multiple UAVs with bidirectional approach: We extended the bidirectional approach to multiple UAVs by combining it with the k-shortest paths algorithm. Specifically, given k UAVs, we apply Yen's algorithm [20] to compute the k shortest paths from the UGV's current position to the destination d. Each UAV is then assigned to inspect the first uninspected edge along one of these reversed paths that has not already been allocated to another UAV. This assignment is performed iteratively until either all UAVs have been allocated or no further uninspected edges remain. This procedure guarantees that no edge is assigned more than one UAV, thereby eliminating redundant inspections. (This also avoids collisions along edges. If multiple UAVs are assigned edges with a common vertex, we assume they avoid collisions by flying at different altitudes.)

C. Time Complexity

Table I summarizes the time complexity of `find_path()` for each of the strategies; it does not include the preprocessing time for the Kemeny constant. The complexity depends primarily on the number of vertices (V) and edges (E) in the graph. The UGV-only and bidirectional strategies use Dijkstra's algorithm, ensuring efficient path computation. In contrast, strategies involving k-shortest paths and multiple UAVs introduce additional complexity due to the need for computing multiple shortest paths [20]. The MPSP strategy performs Dijkstra's algorithm m times to sample candidate paths; we use $m = 20$. Additionally, its complexity is influenced by N, the number of Monte Carlo (MC) runs in the Luby-Karp algorithm, contributing to the overall computational cost.

TABLE I

TIME COMPLEXITY OF DIFFERENT STRATEGIES.

Strategy	Complexity
UGV-only	$O((V+E)\log V)$
Kemeny	$O((V+E)\log V)$
k-Shortest Paths	$O(kV((V+E)\log V))$
MPSP	$O(m(NE+V\log V+\log m))$
Bidirectional	$O((V+E)\log V)$
Multiple UAVs with Bidirectional Strategy	
k UAVs	$O(kV((V+E)\log V))$

IV. RESULTS

A. Datasets

To evaluate the performance of the strategies, we utilized the Line Coverage dataset [4], derived from OpenStreetMap, containing road networks from the 50 most populous cities globally. Each city has two road networks: a small map spanning an area approximately 1 km $\times$ 1 km and a large map covering 3 km $\times$ 3 km.

For each road network, 50 unique instances were generated by randomly selecting damaged edges and assigning obstacle positions on these edges using a uniform distribution. Each edge was assigned an existence probability, randomly sampled from the interval $[0.6, 1.0]$. The minimum probability was set to 0.6 to ensure that the graphs remain sufficiently connected; lower probabilities resulted in too many damaged edges, making it difficult to find a viable path from the start to the destination. To maintain consistency across strategies, each instance was generated with a unique random seed, ensuring the same graph configurations are used for testing all strategies. The length of each edge is known, and is used to calculate the time a UGV needs to traverse it or a UAV needs to inspect it. However, when a UAV needs to reach an edge for inspection, we add the shortest straight-line UAV travel cost to the edge to perform the inspection.

The start and destination vertices of the UGV and UAV(s) were also randomly chosen for each instance.

B. Evaluation Metrics

To assess the effectiveness of the proposed strategies, we employ two evaluation metrics:

- **UGV Travel Time:** This is the total time taken by the UGV to reach its destination. In cases where no viable or safe path exists from the UGV's starting position to the destination, the travel time is defined as the time from the initial start time until the program determines that no viable path is available.
- **Computation Time:** This is the total compute time needed by the algorithm to recalculate paths for the UGV and UAV(s) whenever new information (e.g., a damaged edge) is discovered or an inspection task is completed. It reflects the computational efficiency of the strategy and its ability to handle real-time updates.

These two metrics provide complementary insights into the performance of the strategies. While computation time highlights the responsiveness of the algorithms, travel time evaluates their overall effectiveness in guiding the UGV to its destination under uncertain conditions.

C. Results

The travel time and computation time performance of the presented strategies were analyzed across both small and large maps. A multi-UAV bidirectional strategy with 3, 5, and 7 UAVs was also explored to assess scalability.

To ensure consistency across all scenarios, we set the UGV speed to a constant value of 20 m/s. To evaluate the impact of UAV speed on the strategies, we tested three different speeds for the UAV(s): 20 m/s, 30 m/s, and 40 m/s. These variations capture a wide range of realistic operational conditions and allow us to analyze how the relative speeds of the UGV and UAV(s) affect the overall performance of the strategies.

TABLE II

AVERAGE UGV TRAVEL TIMES (IN SECONDS) FOR FIVE SMALL MAPS. THE STRATEGIES ARE COMPARED FOR NO-UAV AND SINGLE-UAV CASES. MULTIPLE-UAV CASES USE THE BIDIRECTIONAL STRATEGY.

	Moscow	Sao Paulo	Lagos	Tokyo	Mexico City
Lower bound with perfect knowledge					
Perfect knowledge	9.334	21.401	17.543	8.919	25.187
No-UAV and Single-UAV strategies					
UGV-only	40.968	68.855	74.095	31.421	92.167
Kemeny	28.060	42.226	58.834	22.380	63.371
k-shortest paths	36.424	56.533	65.071	28.970	79.559
MPSP	28.609	50.164	**49.386**	22.286	73.552
Bidirectional	**25.620**	**37.292**	54.360	**21.085**	**59.758**
Multiple UAVs with bidirectional strategy					
3 UAVs	23.118	35.460	51.198	19.094	59.951
5 UAVs	22.664	35.824	50.111	18.230	57.659
7 UAVs	**22.506**	**35.160**	**49.111**	**17.221**	**54.526**

Due to space constraints, we only show the results for the most populous city from five continents (Africa, Asia, Europe, North America, and South America) for a UAV speed of 40 m/s. Tables II and III, and Tables IV and V provide the average UGV travel times and computation times, respectively, for small and large maps averaged across 50 random instances for each map. The Perfect knowledge and UGV-only strategies are included only as baselines.

Over the 50 large maps, the bidirectional strategy reduced the average travel time compared to the UGV-only strategy by an average of 26.7% for an equal speed 20:20 ratio, 33.2% for the 20:30 speed ratio, and 38.4% for the 20:40 speed ratio. For the 50 small maps, the average reductions were approximately 7% lower than those for the corresponding large maps at each speed ratio. These results confirm that UAV assistance consistently improves performance, with greater benefits as the UAV becomes faster than the UGV.

A key observation is that the bidirectional strategy outperformed other approaches in most maps by minimizing UGV travel times through effective UAV inspection. Incorporating multiple UAVs further reduced travel times at the expense of greater computational costs because we use the k-shortest paths algorithm for multiple UAVs.

TABLE III

AVERAGE UGV TRAVEL TIMES (IN SECONDS) FOR FIVE LARGE MAPS. THE STRATEGIES ARE COMPARED FOR NO-UAV AND SINGLE-UAV CASES. MULTIPLE-UAV CASES USE THE BIDIRECTIONAL STRATEGY.

	Moscow	Sao Paulo	Lagos	Tokyo	Mexico City
Lower bound with perfect knowledge					
Perfect knowledge	31.024	44.070	48.328	85.778	98.602
No-UAV and Single-UAV strategies					
UGV-only	135.051	212.231	317.366	273.283	364.676
Kemeny	85.720	147.122	216.049	207.027	273.582
k-shortest paths	116.162	174.797	262.704	249.550	320.561
MPSP	88.317	**133.532**	201.392	275.395	285.462
Bidirectional	**78.533**	152.844	**170.639**	**171.723**	**196.409**
Multiple UAVs with bidirectional strategy					
3 UAVs	75.907	117.698	147.994	164.417	193.672
5 UAVs	76.135	124.260	149.846	165.351	188.807
7 UAVs	**71.947**	**116.775**	**138.264**	**159.461**	**183.182**

TABLE IV

AVERAGE COMPUTATION TIMES (IN SECONDS) FOR FIVE SMALL MAPS.

	Moscow	Sao Paulo	Lagos	Tokyo	Mexico City
Lower bound with perfect knowledge					
Perfect knowledge	0.009	0.025	0.037	0.015	0.042
No-UAV and Single UAV strategies					
UGV-only	0.010	0.031	0.059	0.020	0.108
Kemeny	0.012	0.038	**0.085**	0.025	**0.168**
k-shortest paths	0.044	0.229	1.086	0.168	3.947
MPSP	0.029	0.182	0.476	0.093	1.642
Bidirectional	**0.011**	**0.035**	0.111	**0.024**	0.233
Multiple UAVs with bidirectional strategy					
3 UAVs	**0.038**	**0.166**	**1.424**	**0.126**	**4.766**
5 UAVs	0.075	0.336	3.085	0.254	10.700
7 UAVs	0.123	0.509	4.877	0.372	15.654

TABLE V

AVERAGE COMPUTATION TIMES (IN SECONDS) FOR FIVE LARGE MAPS.

	Moscow	Sao Paulo	Lagos	Tokyo	Mexico City
Lower bound with perfect knowledge					
Perfect knowledge	0.095	0.216	0.188	0.341	0.391
No-UAV and Single UAV strategies					
UGV-only	0.098	0.238	0.221	0.452	0.513
Kemeny	**0.102**	**0.254**	**0.244**	**0.528**	**0.614**
k-shortest paths	0.280	1.190	1.965	5.064	6.343
MPSP	0.148	0.582	0.726	6.213	4.735
Bidirectional	0.104	0.269	0.272	0.627	0.631
Multiple UAVs with bidirectional strategy					
3 UAVs	**0.289**	**0.991**	**1.768**	**6.475**	**5.074**
5 UAVs	0.697	2.586	5.043	15.026	10.623
7 UAVs	0.955	3.294	6.966	24.505	23.890

V. ADDRESSING PRACTICAL LIMITATIONS

In this paper, we focused on the *replanning* component of each strategy for the DUCPP problem to evaluate how

different replanning choices affect the UGV travel time. Towards this, we made several simplifying assumptions; relaxing them is important for practical systems.

- *Delay-free communication.* In a disaster, network connectivity may be intermittent. However, this can be mitigated in practice by leveraging resilient communication infrastructure (e.g., satellite internet) to maintain real-time coordination.
- *UAV energy limits.* UAVs have limited battery life, and energy constraints are often a key reason to use multiple UAVs. A natural next step is to perform *energy-aware* planning by assigning each UAV an energy budget and optimizing inspection routes to balance (i) expected reduction in UGV detours and (ii) energy cost, while guaranteeing that each UAV can return to a launch/recharge site. We also plan to study simple recharging strategies (e.g., selection of recharge station locations, online UAV recharge scheduling).
- *UAV straight-line deadheading.* UAV flights may encounter no-fly zones, obstacles, and altitude limits. These constraints can be addressed by generating physically feasible UAV flight paths to replace straight-line travel.

VI. Conclusion

This paper introduced and analyzed the Dynamic UGV-UAV Cooperative Path Planning (DUCPP) problem, addressing the challenges of path finding for a UGV in uncertain road environments. UAVs can perform edge inspections to efficiently identify and eliminate impassable edges and thus reduce UGV travel times and optimize overall performance. The presented strategies, particularly the bidirectional approach, demonstrated significant improvements in reducing UGV travel times and adapting to uncertain edge conditions. The use of multiple UAVs further reduced UGV travel times, at the cost of increased computation.

To maintain generality and ensure flexibility, we opted to use Dijkstra's algorithm for path finding. However, exploring heuristic methods such as A* or D* Lite in future work could provide valuable insights and computational speedups.

The experimental results across diverse road networks and scenarios underscore the robustness and scalability of the DUCPP framework in environments with uncertain conditions. The findings affirm the potential of UGV-UAV cooperative path planning and inspection as a practical solution for applications such as disaster response, supply transport, and rescue missions. Future research directions include integrating heuristic path planning approaches, optimizing the assignment of multiple UAVs to subgraphs, extending the framework to support multiple UGVs, and incorporating terrain data to estimate edge existence probabilities in challenging environments.

References

[1] S. Koenig and M. Likhachev, "D* Lite," in *The Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, (Edmonton, Canada), p. 476–483, July 2002.

[2] C. H. Papadimitriou and M. Yannakakis, "Shortest paths without a map," *Theoretical Computer Science*, vol. 84, no. 1, pp. 127–150, 1991.

[3] M. Dille and S. Singh, "Efficient aerial coverage search in road networks," in *AIAA Guidance, Navigation, and Control Conference*, (Boston, MA), pp. 5048–5067, Aug. 2013.

[4] S. Agarwal and S. Akella, "Line coverage with multiple robots: Algorithms and experiments," *IEEE Transactions on Robotics*, vol. 40, pp. 1664–1683, 2024.

[5] A. Kashyap, D. Ghose, P. P. Menon, P. Sujit, and K. Das, "UAV aided dynamic routing of resources in a flood scenario," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 328–335, June 2019.

[6] I. Munasinghe, A. Perera, and R. C. Deo, "A comprehensive review of UAV-UGV collaboration: Advancements and challenges," *Journal of Sensor and Actuator Networks*, vol. 13, no. 6, 2024.

[7] C. Liu, J. Zhao, and N. Sun, "A review of collaborative air-ground robots research," *Journal of Intelligent & Robotic Systems*, vol. 106, no. 3, p. 60, 2022.

[8] A. Bar-Noy and B. Schieber, "The Canadian traveller problem," in *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '91, (USA), p. 261–270, Society for Industrial and Applied Mathematics, 1991.

[9] E. Bampis, B. Escoffier, and M. Xefteris, "Canadian traveller problem with predictions," in *20th International Workshop on Approximation and Online Algorithms (WAOA 2022)*, p. 116–133, 2022.

[10] Z. Bnaya, A. Felner, and S. E. Shimony, "Canadian traveler problem with remote sensing," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, p. 437–442, 2009.

[11] G. Macrina, L. Di Puglia Pugliese, F. Guerriero, and G. Laporte, "Drone-aided routing: A literature review," *Transportation Research Part C: Emerging Technologies*, vol. 120, p. 102762, 2020.

[12] S. Choudhury, K. Solovey, M. Kochenderfer, and M. Pavone, "Coordinated multi-agent pathfinding for drones and trucks over road networks," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, p. 272–280, 2022.

[13] A. Otto, B. Golden, C. Lorenz, Y. Luo, E. Pesch, and L. Rocha, "On delivery policies for a truck-and-drone tandem in disaster relief," *IISE Transactions*, vol. 57, pp. 1–31, 10 2024.

[14] A. S. Bhadoriya, S. Rathinam, S. Darbha, D. W. Casbeer, and S. G. Manyam, "Assisted path planning for a UGV-UAV team through a stochastic network," *Journal of the Indian Institute of Science*, vol. 104, pp. 691–710, 2024.

[15] A. S. Bhadoriya, C. M. Montez, S. Rathinam, S. Darbha, D. W. Casbeer, and S. G. Manyam, "Optimal path planning for a convoy-support vehicle pair through a repairable network," *IEEE Transactions on Automation Science and Engineering*, vol. 21, pp. 4936–4947, Oct 2024.

[16] A. Lakas, B. Belkhouche, O. Benkraouda, A. Shuaib, and H. J. Alasmawi, "A framework for a cooperative UAV-UGV system for path discovery and planning," in *2018 International Conference on Innovations in Information Technology (IIT)*, pp. 42–46, 2018.

[17] S. Zhang, H. Wang, S. He, C. Zhang, and J. Liu, "An autonomous air-ground cooperative field surveillance system with quadrotor UAV and unmanned ATV robots," in *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 1527–1532, 2018.

[18] N. Chen, Z. Li, L. Quan, X. Chen, C. Xu, F. Gao, and Y. Cao, "Cost-effective swarm navigation system via close cooperation," *IEEE Robotics and Automation Letters*, vol. 9, no. 11, pp. 9343–9350, 2024.

[19] D. Altafini, D. A. Bini, V. Cutini, B. Meini, and F. Poloni, "An edge centrality measure based on the Kemeny constant," *SIAM Journal on Matrix Analysis and Applications*, vol. 44, no. 2, pp. 648–669, 2023.

[20] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[21] A. Saha, R. Brokkelkamp, Y. Velaj, A. Khan, and F. Bonchi, "Shortest paths and centrality in uncertain networks," *Proc. VLDB Endow.*, vol. 14, p. 1188–1201, Mar. 2021.

[22] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[23] C. Li, H. Ma, J. Wang, and M. Q.-H. Meng, "Bidirectional search strategy for incremental search-based path planning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7311–7317, 2023.