IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Online Path Repair: Adapting to Robot Failures in Multi-Robot Aerial Surveys

Jaden Clark[1], Kunal Shah[2], Mac Schwager[1]

***Abstract*— Multiple Unpiloted Aerial Vehicles (UAVs) working together have the potential to efficiently survey large geographical areas. Unfortunately, UAVs in the field may fail midway through a survey due to adverse weather, faster-than-expected battery drain, or mechanical malfunction, leaving part of the survey area uncovered. Here we propose an algorithm to re-plan coverage routes online for multiple UAVs to take over the remaining route of a failed team member. We first present a greedy path recovery algorithm whereby each UAV greedily absorbs the closest remaining vertices from the failed UAV's route into its own route. This method is then extended using a Tabu search method for multi-agent path repair to give successively better quality paths. We call the new path repair algorithm GRIT (Greedy Repair Initializes Tabu search), and demonstrate it performing path repair for nominal paths planned with both a traditional lawnmower-style planner and a more sophisticated integer program based planner. We show that GRIT achieves adequate re-plans 10-50 times faster than two benchmark planners, making it ideal for online path repair in mid-flight, although the benchmarks eventually outperform GRIT if given unlimited computation time.**

I. INTRODUCTION

Unpiloted aerial vehicles (UAVs) are useful for a range of surveillance and monitoring tasks in both natural and artificial environments. For example, UAVs have been deployed for search and rescue missions [1], forest fire monitoring [2], and animal population monitoring [3], [4]. In most of these applications, multiple UAVs working together can perform the task faster, safer, and more efficiently than traditional methods [5]. These UAV surveys require efficient multi-robot coverage planning algorithms.

The most simple coverage path planning algorithms generate lawnmower-style paths, which trace a forward and backward path across the desired area of coverage. This method has been generalized to multi-robot coverage planning using efficient cellular decomposition of a region, allowing each robot to plan a lawnmower-style path within its own cell [6], [7], [8]. While generating lawnmower-style paths is fast and efficient, lawnmower-style paths generate unnecessary backtracking during coverage (lawnmower-style paths usually start and end at different locations on the graph, and do not take into consideration the battery drain from routing the robot back to the starting position). More recent multi-robot coverage path planning algorithms, such as POPCORN [3], consider such backtracking and plan efficient cyclic paths over the desired coverage area (see Fig. 1).

Unfortunately, such pre-planned paths may be disrupted during execution. For example, if the UAV is performing a photographic survey the camera may become obscured or a memory card could malfunction. Battery life is unpredictable during flight, particularly during windy or cold weather conditions [3]. Hence, UAVs may unexpectedly run out of battery mid-flight, requiring a UAV to leave its route unfinished. In this case, if the remaining UAVs do not adapt to take over the remainder of the failed UAV's route, a hole will be left in the survey area. Thus, effective UAV survey systems should incorporate online methods for re-planning UAV paths in case a member of the team fails during execution.

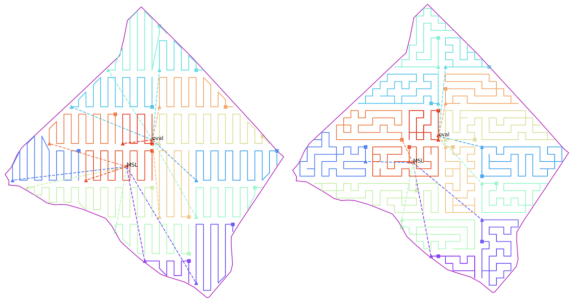

Fig. 1. In multi-robot coverage planning, multiple UAVs survey a designated area of coverage. Routes are planned using lawnmower-style coverage (left) and POPCORN (right). Coverage surveys were planned over a geofence on Stanford University. Different colors indicate routes for different UAVs. MSL and Oval are the intial UAV locations. Dashed lines indicate intial drone routes to their coverage area. Triangles and squares indicate the start and end of each UAV's route, respectively.

In this paper we propose an algorithm to re-plan multi-robot coverage paths online to compensate for a failed robot unable to complete its original route, the first part of which is shown in Fig. 2. Specifically, we consider a system of multi-rotors that can translate in any direction to efficiently visit a series of waypoints, without minimum speed or other kinematic constraints - unlike fixed wings UAVs. We formulate this problem as a constrained path planning problem on a graph. We consider a system with n robots, one of which fails mid-execution, leaving one partial route that will not be completed under the current plan. Therefore this partial route needs to be integrated into the $n-1$ routes that are already in progress. The goal of the re-planning process is to re-route the remaining robots to visit every vertex on the

This work was partially supported by NSF award 1834986. We are grateful for this support.

[1]Jaden Clark and Mac Schwager are with the Department of Aerospace Engineering, Stanford University, 496 Lomita Mall, Stanford, CA, United States jvclark@stanford.edu, schwager@stanford.edu

[2]Kunal Shah is with Dexerity, Inc., 1205 Veterans Blvd, Redwood City, CA 94063 k2shah@alumni.stanford.edu

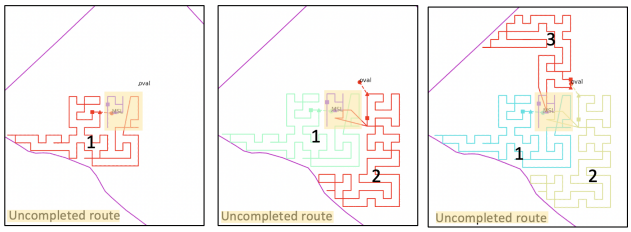

Fig. 2. Iterative path repair using our proposed greedy heuristic. Here we show only the failed route, and the routes used to repair its remaining vertices (other routes proceed unchanged). The purple robot fails leaving most of its path (highlighted in yellow) uncovered. The robot with greatest remaining battery life is shown in red in each iteration. (Left) Robot 1 has the most remaining battery life from its original plan, so it takes on all the remaining vertices it can given its battery limits. (Middle) Robot 2 has the second most remaining battery life, so it takes as many remaining vertices as it can, subject to its battery limits. This repeats greedily until all uncovered points from the failed robot are taken by a healthy one (the route is fully repaired), or all healthy robots have used up their battery limits (the route is partially repaired, and there is no more robot capacity). In this example, only three robots need to be re-routed to fully repair the survey.

graph such that no more UAVs must be added to the survey and no UAV exceeds its battery constraints. In addition, the re-plan must be computed quickly, so that battery life is not wasted during online computation, and UAVs can re-route their paths before more of the original paths are completed. In the case of two or more failed robots, our algorithm can be run sequentially to incorporate each failed path one-at-a-time into the existing routes. If multiple robots fail at once, their waypoints can be considered from the same route and the algorithm can be run as usual. We illustrate this re-planning problem in Fig. 3.

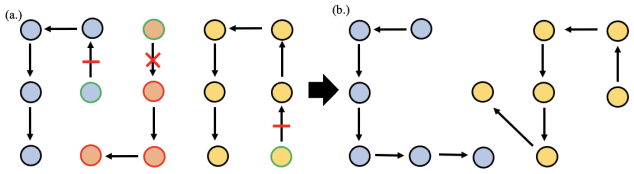

Fig. 3. Illustration of the online replanning problem. On the left, each vertex color indicates a different initial route. Green outline indicates a vertex that has already been completed, black outline indicates a vertex that has not yet been completed but is planned, and red outline indicates vertices that will not be completed under the current plan because the corresponding robot failed. Red X indicates where the robot failed, and red lines indicate where the remaining healthy robots were at time of failure. Here, the orange route failed after completion of one vertex. Our Greedy Repair algorithm allocates the remaining uncompleted vertices (in red outline) into the other two routes. First, the route with the least cost (blue) inserts as many vertices as possible (b.). Then the yellow route inserts as many vertices as possible (c.), successfully allocating all vertices.

Our algorithm has two parts. We first introduce a greedy path repair algorithm that quickly and efficiently re-allocates parts of the uncompleted route to in-progress routes. Unlike other re-planning algorithms, this method leverages previous route information to develop new routes that fully allocate uncompleted vertices in a survey. We then use this greedy solutions as an initialization for a Tabu-search based re-planning method. The Tabu-search iterates through local solutions to the re-planning problem to improve route efficiency. We show that the combined algorithm GRIT (Greedy Repair Initializes Tabu search) can quickly repair a failed survey, and iterate on the initial solutions to progressively improve path quality using as many or as few iterations as computation time allows.

GRIT is able to solve for feasible paths (i.e. respecting all UAV battery constraints while visiting all vertices in the survey) using just a few Tabu iterations, whereas Tabu search alone fails to generate feasible routes under appropriate time constraints. We note that GRIT is a centralized algorithm which would be solved at a base station during execution of a survey, and the replanned routes sent out the UAVs for execution. We do not consider this to be too strong of an assumption as in current fielded UAV surveillance systems, routes are pre-planned centrally by a computer on the ground, and UAVs remain linked to the ground station throughout the survey. Federal Aviation Administration (FAA) regulations require that a human pilot can always take direct command of the aircraft in case of an emergency [9]. Specifically, a central ground station would be able to identify a failed UAV, re-plan routes for all remaining UAVs, and send those updated route commands to the UAVs in a matter of seconds. However, we do expect that the round-robin nature of the Tabu search may admit a naturally distributed variant. In future work, we will explore distributing this computation to take place on board the UAVs communicating with one another over a wireless network.

We conduct a simulation study to compare GRIT with two other baseline path repair algorithms: (i) POPCORN [3] running in an online fashion, and (ii) Tabu search from [10]. When tested on initial routes planned over a geofence over Stanford, we find that GRIT solves for feasible new paths in 61% of cases within 10 sec, compared with 20% for POPCORN and 0% for Tabu search. We note that POPCORN and Tabu search ultimately find similar re-planned paths than GRIT, but they take too long (300s or more) to be practical for online replanning.

In the rest of this paper, we discuss related work in Sec. II, formalize the online path repair problem in Sec. III, describe GRIT, our algorithm for path repair, in Sec. IV, discuss results of simulated survey re-planning scenarios in Sec. V, and offer conclusions in Sec. VI

II. Related Work

A. Online Multi-Robot Re-planning

Although there is little literature on online re-planning for multi-robot coverage due to robot malfunction, other online re-planning methods typically share similar objectives. For example, re-planning usually necessitates a faster heuristic with different route initialization than the original planner [11]. Re-planning can also make use of initial, efficient routes.

Re-planning is useful whenever real-time changes occur to the environment or objective. For example, in [1] robots replan their routes in reponse to new information obtained during a route. Changes to the robots' environment may also

occur during execution. For example, unanticipated obstacles or weather conditions may necessitate online adjustments to the route, as considered in [12]. In addition if the graph that routes were planned on is dynamic or inaccurate, re-planning is essential, as considered in [1], [13]. Some works also use an ergodic control approach to a changing environment, which enables continuous optimization online but does not guarantee completion of a set of discrete waypoints [14] [15].

The most relevant work on robot failure in multi-robot planning exists in the adaptive coverage planning literature, in which robots have to deal with changing environmental conditions or adversarial agents. DEC-PPCPP [16], BNNB [17], and BoB [18] all handle coverage path planning problems under time constraints. However, each of these methods considers a significantly different problem from the one we define. For example, a major limitation of many coverage algorithms for real-world application is that they fail to consider battery life constraints, which are essential constraints in UAV aerial surveys, and are an important aspect of our planner.

DEC-PPCPP [16] is an adaptive algorithm based on a predator-prey model. This method handles non-stationary obstacles and other environmental changes in a decentralised fashion. DEC-PPCPP leverages offline routing, and tunes the previous path online. However, it does not consider battery constraints of agents. BNNB [17] is a method based on local sensing. Each route is successively updated based on information learned from noisy sensor measurements. However, like DEC-PPCPP, it also does not consider battery constraints of agents. BoB does handle battery constraints, however it does not enforce cyclic paths - allowing robots to fail at random vertices throughout the survey. This is impractical for real-world deployment of UAVs, as UAVs must often be recalled to a starting point for the survey for safety, legal, or practical reasons. For example, if a UAV is surveying a penguin colony, and it cannot land in the middle of the colony when it runs out of battery life.

To our knowledge, no work in the multi-robot coverage planning literature considers robot failures and limited battery life, while preserving cyclic routes. For this reason, we look to literature on the disrupted vehicle routing problem (VRP), while leveraging insights from online multi-robot re-planning.

B. Disrupted Vehicle Routing Problem

Solving for a complete route through a graph with a single agent is akin to the traveling salesperson's problem (TSP). We consider the multi-agent extension of the TSP, the vehicle routing problem (VRP). The vehicle routing problem has been applied to a number of operations tasks, including freight transportation, delivery-driver routing, and aircraft routing [19].

The VRP has been modified for a number of industrial applications. The capacitated VRP handles problems when agents have maximum payload [20]. The VRP with time windows satisfies problems that each payload is dropped off during a specific time range [21]. The disrupted VRP applies to situations in which a route or agent is changed during execution. The VRP with multiple trips deals with cases where vehicles can complete more than one route. These variants of the VRP have frequently been combined for a number of applications.

The disrupted VRP with capacity constraints is the VRP variant most closely aligned with our online replanning scenario. The disrupted VRP typically requires solutions with fast computation and different constraints from the original routing problem, similarly to our multi-robot re-planning problem. Some variants of the disrupted VRP also handle vehicle failure [10], [22], [23]. We will leverage such solutions for our own re-planning method.

The disrupted VRP with vehicle failure and capacity constraints is often solved using an iterative Tabu-search algorithm [10], [22]. When a vehicle fails, an initial (typically infeasible) route is constructed by inserting uncompleted vertices into in-progress routes based on distance. Then an iterative "neighborhood search" is performed, in which uncompleted destinations are attempted to be re-merged into other routes until a feasible solution is found. In one neighborhood search, a cost is calculated for each destination merge, and the best one is kept. This is called the "move" of that iteration. A "Tabu list" keeps track of past moves, and prevents the same move from being taken multiple times within the length of the Tabu list. Neighborhood searches are computed until a desired number of iterations or desired cost has been reached. This algorithm is advantageous in that it has flexible runtime and is faster than existing VRP heuristics when combined with good initial routes. We leverage a modification of this algorithm as the second component of GRIT, and we introduce a new greedy initialization heuristic as the first component.

III. Problem Formulation

Starting from an initial geofence of the survey area, we grid up the area into a lattice of vertices whose spacing is determined by the desired resolution of the survey. We then construct a graph by connecting nearest neighbors in the lattice. In the resulting graph $G(V, E)$ embedded in R^2 all vertices $v \in V$ correspond to positions in the lattice, and all $e \in E$ are weighted edges connecting vertices. The weight w_e of each edge $e = \{v_i, v_j\}$ is the battery life required to fly between v_i, v_j, which is proportional to the Euclidean distance between the two corresponding lattice vertices. The coverage planner constructs paths for n robots on G, in which each path is an ordered sequence of vertices $p_i = (v_{i1}, v_{i2}, v_{i3}, \ldots)$. We denote the set of elements in the tuple p_i as $\{p_i\}$, and the battery life exhausted by path p_i we write as $\|p_i\| := \sum_{j=1}^{N_i-1} w_{v_{ij}, v_{i(j+1)}}$ for a path with N_i vertices. Every path p_i must also start and end at a pre-determined home vertex H on the geofence such that $p_i[0] = p_i[N_i] = H$, which is assumed to be the same for all robot. Each path is the route for a single UAV, and the union of all n paths should cover all $v \in V$,

$$\{p_1\} \cup \{p_2\} \cup \{p_3\} \cup \ldots \cup \{p_n\} = V. \tag{1}$$

Furthermore, each path must satisfy the battery constraints of its respective robot (it must not travel greater than a certain distance during completion of its path), $||p_i|| \leq p_i^{max}$. The goal of the planner is to find plans for the n robots that minimize the length of the paths summed over all robots, subject to the coverage constraint (1) and the battery constraints.

In the re-planning scenario, a coverage planner has already computed all n complete paths. However, midway through the survey, the k-th robot fails, and can no longer complete its route. The corresponding uncompleted route portion is p_k^u, and the completed route portion p_k^c such that $\{p_k^c\} \cup \{p_k^u\} = \{p_k\}$. All remaining robots then have completed paths p_i^c, and a remaining planned path p_i^p. Because route p_k^u will not be covered by the failed robot k, the survey area will not be completed if the remaining robots continue on their initial paths. Then, the goal of the remaining $n-1$ robots is to adjust each p_i^p such that the union of new paths p_i^p, old paths p_i^c and the partial path completed by the failed robot p_k^c cover all $v \in V$, while adhering to battery constraints,

$$(\cup_{i \neq k}\{p_i^c\}) \cup (\cup_{i \neq k}\{p_i^p\}) \cup \{p_k^c\} = V. \tag{2}$$

We note that this is different from the original coverage planning problem in several ways:

- *Initial robot locations and constraints.* In the original coverage path planning problem, all robots start with the same locations and battery constraints. In the re-planning problem, robots start at a location midway through the original survey (wherever each robot is when the failed robot malfunctions). In addition, robots may have varying amounts of excess battery life, depending on the route that was planned at initialization, wind conditions encountered during execution so far, and the natural variations of individual battery discharge rates. We estimate the expected remaining battery life to be proportional to the distance a UAV can travel.
- *Planning time.* Initial planning occurs before the flight, so there are no constraints on how long the solver takes to plan routes. For example, a planner using integer programming may take upwards of 10 hours to solve for optimal paths [3]. However, for re-planning during flight, new routes should be planned as quickly as possible to avoid wasting valuable flight time, and the whole plan must be solved before any robot proceeds to the next vertex in its original route.
- *Prior knowledge.* The initial coverage problem is solved without any initial information (there are no existing routes that can be used to inform the planning step). However, the re-planning process can use the initial route plan. This may speed up re-planning and increase the number of possible strategies for re-planning.

IV. The GRIT Algorithm

Our algorithm consists of two phases, (i) Greedy Path Repair, followed by (ii) a Tabu search. We describe each below. We assume the initial coverage problem has been solved.

A. Greedy Path Repair

We propose a greedy heuristic that preserves much of the original routes, while being quick and efficient. The algorithm we present maintains the order of vertices in the initial routes, while inserting uncompleted vertices from the failed robot's path. By preserving the original order of vertices, the method is able to leverage the efficiency of pre-planned route for quick and efficient replanning. It also ensures that all paths are cyclic, verifying that no extra backtracking is added. This method is also fast—our Python implementation typically completing 2.4 times faster than a Tabu search iteration.

Algorithm 1 GreedyPathRepair

Input: $P = \{p_1^p, p_2^p ...\}$ in-progress routes
$p_k^u = (u_1, u_2 ...)$ ordered unallocated vertices
Output: P, p_k^u
sort(P)
for $p_i^p \in P$ **do**
  while $||p_i^p|| \leq p_i^{max}$ **do**
    insert(p_i^p,min(u_1, u_{-1})) ▷ insert end vertex of p_k^u
    if p_k^u empty **then**
      break
    end if
  end while
end for
return P, p_k^u

The greedy heuristic starts by sorting the set of all in-progress routes P based on how much battery life they will retain if they were to travel to the uncompleted route, then proceed with their existing route. Specifically, this excess battery life is calculated for each route by finding the closest distance vertex on the in-progress route from the uncompleted route and estimating the battery cost to traverse that distance, w_e, twice (once to leave the in-progress route, and once to return to it).

The algorithm begins with the i-th route with the most excess battery life, and performs a greedy insertion of vertices from the failed robot's uncompleted path p_k^u into the healthy robot's planned path p_i^p, while preserving the order of the vertices in p_i^p. This is accomplished by selecting the closer end-vertex of p_k^u (either the first or last vertex on p_k^u) and performing an insertion: removing the end-vertex from p_k^u and adding it to p_i^p. These vertices of p_k^u are kept in order, and inserted as an ordered sequence after the closest end vertex in p_i^p. Vertices are added to p_i^p as long as the amount of battery the route requires $||p_i^p||$ stays under the battery limit p_i^{max}. We continue to add vertices to the healthy robot's path in this way until the robot runs out of battery life.

This process is repeated for every route with excess battery until the uncompleted route is fully accounted for. If the route cannot be fully accounted for, a Tabu search is initialized. If the route is still uncompleted, the in-progress routes proceed to cover the most vertices possible from the uncompleted

path using GRIT, and another UAV will have to be added to the survey. We show an example implementation in Fig. 2 and an illustration of the planning problem in Fig. 3. The full method is described in Algorithm 1. We note that u_{-1} indicates the last index. The full algorithm is visualized in the multimedia supplementary file.

B. Tabu Search

After the initial Greedy Path Repair procedure, we implement a Tabu search algorithm similar to [10]. We choose Tabu Search to augment Greedy Path Repair since it is a proven baseline in the disrupted VRP. Tabu Search requires an initial guess for viable routes, typically created by inserting each point in p_k^u to the closest p_i^p. In this way, vertices are initially inserted in an order that is not necessarily most efficient, and is likely infeasible. Instead, we use Greedy Repair as the initial guess. If some vertices are left unallocated (if Greedy Repair failed), they are inserted into the closest p_i^p (based on distance from any point in the route, not just the end points as in greedy repair), thus initializing with routes that potentially violate UAV battery constraints. The algorithm then iterates through neighborhood search solutions (Algorithm 2) until a feasible solution is found (i.e. all UAV battery constraints are satisfied and all vertices are covered) or a maximum number of iterations has been reached.

Algorithm 2 NeighborhoodSearch

Input: $P = \{p_1^p, p_2^p ...\}$ in-progress routes
$p_k^u = (u_1, u_2 ...)$ ordered unallocated vertices
Output: P
$T \leftarrow []$
for $u_i \in p_k^u$ **do**
  $C \leftarrow cost(P)$, (1)
  for $p_j^p \in P$ **do**
    insert(p_j^p,u_i) ▷ move vertex u_i into route p_j
    if cost(P)> C or $\{p_j^p, u_i\}$ in T, (1) **then**
      remove(p_j, u_i) ▷ move u_i from p_j back to p_k^u
    else
      $C \leftarrow cost(P)$
      $T \leftarrow [T, (p_j^p u_i)]$
    end if
  end for
end for
return P

Each neighborhood solution is based on an exhaustive vertex insertion process. For every neighborhood search, each vertex in p_k^u is attempted to be inserted to each p_i^p. If moving a vertex to another route decreases total route cost, then the change is preserved, and that vertex is added to the Tabu list. The cost C for a set of routes P is calculated as

$$cost(P) = \sum_{p_i \in P} \|p_i\| + \rho E, \text{ (1)}$$

where ρ is a penalty for constraint violation (when the distance traveled is infeasible due to battery constraints) and E is the excess distance beyond that allowed by the battery constraints. $\rho > 1$ is included to allow exploration of vertex allocations that violate battery constraints at early Tabu iterations, while still requiring plans to ultimately satisfy battery constraints. Once a desired amount of neighborhood iterations have been computed, the feasible route with least cost is returned. If no feasible route is found, then the Tabu Search fails. The full method is described in Algorithm 2.

C. GRIT

GRIT (Greedy Repair Initializes Tabu search) conducts an initial re-route using the greedy path repair method. Then, it performs a Tabu Search. For a desired max number of iterations M, the neighborhood search is conducted. If a feasible solution arises, it is saved and the route is deemed complete. The Tabu search can then use remaining iterations to improve the efficiency of the re-planned route. The full method is in Algorithm 3.

Algorithm 3 GRIT

Input: $P = \{p_1^p, p_2^p ...\}$ in-progress routes
$p_k^u = (u_1, u_2 ...)$ ordered, unallocated vertices
i iterations

Output: s
$P, p_k^u \leftarrow$ GreedyRepair(P, p_k^u)
if p_k^u not empty **then**
  insert p_k^u into nearest $p_i^p \in P$
end if
for $i < M$ **do**
  $P \leftarrow$NeighborhoodSearch(P, p_k^u)
  if P is feasible **then**
    $s \leftarrow P$
  end if
end for
return s

D. Complexity Analysis

We consider the aforementioned path re-planning problem. Suppose we store all vertices on the graph in ordered lists. There exist $h = ||p_k^u||$ unallocated vertices from the failed robot and $n-1$ routes that are midway through completion. In each route, there exist v total vertices remaining to be completed. In Greedy Repair, uncompleted routes are sorted based on minimum distance added by inserting an uncompleted vertex from the failed robot p_k^u into the route of a healthy robot. Sorting involves inserting each of h unallocated vertices into each of the $n-1$ routes. Insertion is $\mathcal{O}(v)$, as each vertex in the uncompleted route must be checked to find the best insertion location. This insertion occurs once for each of $n-1$ routes, so sorting has time complexity $\mathcal{O}(nv)$.

After sorting, GRIT begins the greedy insertion process. The first uncompleted route in the queue adds as many unallocated vertices as possible to the route. This process has time complexity $\mathcal{O}(nh)$, since each route checks as many as h vertices. Checking route feasibility is $\mathcal{O}(1)$, as one must

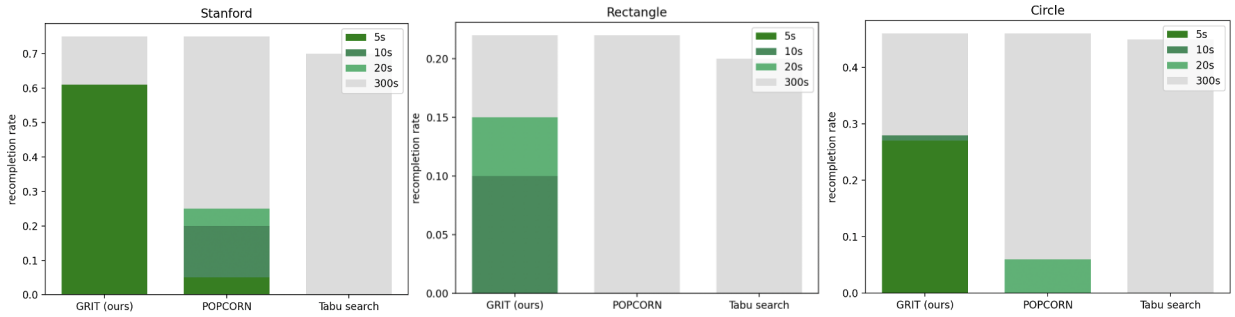

Fig. 4. Replan performance and computation time using POPCORN, GRIT, and Tabu search over 40 simulations. The bars show the percent of routes for which the entire survey is still completed with re-planning after a single UAV fails at a random vertex mid-route. We note that each of the 3 plots has a different maximum value, as different shapes have varying difficulty of recompletion. Different colors indicate different replan computation times. GRIT significantly outperforms POPCORN and Tabu search at short time scales (5-20 seconds) required for online path repair. If given sufficient time (300s) all three methods provide similar performance, however this time range is not practical for replanning online during a survey. We note that the completion rate never reaches 1.0, as recompletion is not always feasible. In other words, there is no way with the given battery life of all the drones for every survey point to be reached (no matter how optimal the routing is).

only compute the distance between the unallocated vertex, and where it is inserted into the uncompleted route, and add this distance to the previous route cost. Thus Greedy Repair has overall time complexity of $\mathcal{O}(n(v+h))$.

The Tabu Search is an exhaustive algorithm. In every neighborhood search, each of the $n-1$ routes attempt to accept each of the h unallocated vertices. Insertion is $\mathcal{O}(v)$. Thus a neighborhood search is $\mathcal{O}(vnh)$. Then, a Tabu Search with i iterations has complexity $\mathcal{O}(ivnh)$. We see that Greedy Path Repair and Tabu Search scale proportionally to n, v and h. However, Greedy Path Repair should be faster than a single Tabu iteration as runtime scales proportionally to $v+h$, as opposed to vh.

V. Simulation Results

Algorithms were tested based on coverage over multiple difference survey regions. Here, we considered a UAV with 13.5 minutes of flight time, the amount of time available for the DJI Matrice 100. Coverage algorithms were implemented in Python and run on a machine with a 2.6 GHz 6-Core Intel Core i7 processor. Initial route planning was conducted using a simple lawnmower-style planner. We consider a system of 16 robots, a penalty parameters $\rho = 20$ for the Tabu search, and a Tabu history of 10 moves. Initial coverage paths were static for each survey region, but for every simulation a random route and vertex along that route were selected for failure. When a failure was simulated, all routes were halted, and re-planning methods were employed. We compared performance of GRIT with a naively initialized Tabu search, and a re-deployment of a Satisfiability Modulo Theory (SMT) - based method: POPCORN [10], [3]. POPCORN discretizes the geofenced region into tiles, for which optimal cyclic paths are solved using an SMT solver. These tiles are then linked together using breadth first partitioning. The re-deployment of POPCORN removes each vertex v from the original graph G that had already been completed at the vertex of failure to produce a new graph G'. It then attempts to plan paths over G'.

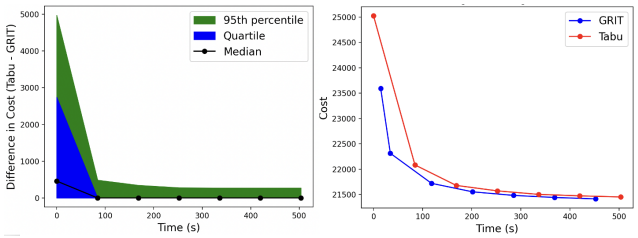

Fig. 5. Average route cost as a function of timing in GRIT compared to Tabu search (right) and median difference in route cost with quartiles and 5th and 95th percentiles on the left. 1000 simulations were conducted in which a random route failed at a random vertex each time. Both methods run 6 iterations of Tabu search. GRIT finds a feasible route within seconds, and iteratively improves its cost using Tabu iterations. Tabu Search alone starts with an infeasible solution, resulting in a very high cost. As a result, GRIT has a lower cost solution 97.17 percent of the time.

A. Performance

GRIT (with just 3 Tabu iterations) is tested on routes initially planned with a lawnmower-style method. A modified version of POPCORN (adjusted to accept a set of uncompleted points, instead of a full geofence), as well as a raw Tabu search implementation are compared for re-planning performance. Fig. 4 shows replan success statistics for 40 simulations in each of three environments: (i) a simulated survey of the Stanford campus with 739 vertices, (ii) a rectangular grid with 646 vertices, and (iii) a circular grid with 760 vertices. GRIT repairs significantly more surveys than both POPCORN and Tabu search over short time durations (5-20 seconds), whereas as POPCORN and Tabu search show stronger performance when given more time (the greedy aspect of GRIT is suboptimal, although it has a significant speed advantage). Notably, all three methods struggle to re-plan routes in the rectangular grid, possibly because the regular geometry leads to similar-length initial UAV routes that are already close to their battery limits. Hence, after a failure, there is less discrepancy in remaining battery life among the UAVs to take up the uncovered vertices of the failed UAV. Tabu search consistently improves

efficiency of routes initially replanned using GRIT. Cost C for a set of routes was plotted as a function of timing for 6 Tabu iterations in Fig. 5. We found that in 90.51 percent of scenarios, GRIT had the same or lower cost than Tabu search for the entire replanning window (up to 6 Tabu iterations). In the 9.49 percent of scenarios where Tabu search is at some point faster than GRIT, GRIT is still faster for 70.14 percent of the survey. This means that considering all replanning scenarios, GRIT is faster than Tabu 97.17 percent of the time. We note that no method reach a recompletion rate of 1.0, as not all routes are possible to be recompleted within the UAV's allocated battery life.

We also compare Greedy Repair, GRIT, Tabu search and POPCORN as a function of number of UAVs 6. We produced 3 circular geofences. We simulated 16, 32, and 64 UAV teams which were tested on the smallest, medium, and largest geofences respectively. We simulated 100 random UAV failures for each setup. For each geofence and UAV number setup, GRIT (and then Greedy Repair) always outperforms existing methods. In particular, we found that other methods scale more poorly with respect to number of UAVs.

UAVs	Greedy	GRIT	Tabu	POPCORN
16	0.85	**0.61**	0.40	0.45
32	0.71	**0.40**	0.14	0.13
64	0.65	**0.35**	0.20	0.0

Fig. 6. Completion rate for Greedy Repair, GRIT, Tabu search, and POPCORN as a function of number of UAVs. We simulate 100 random UAV failures midway through planning and allow each method 60 seconds to plan. We find that GRIT consistently outperforms other methods - particularly Tabu search and POPCORN which scale poorly for larger coverage areas and number of agents.

B. Runtime

Greedy repair, Tabu iterations, and the modified POPCORN algorithm were timed on simulated surveys in which one random route failed at a random vertex. POPCORN takes significantly more time and scales poorly compared to Greedy Path Repair. Overall Greedy Repair is much faster than both Tabu Search and POPCORN, as shown in Fig. 7.

C. Discussion

The results show that GRIT performs significantly better than existing coverage planning and replanning methods at time ranges practical for online path repair (5s-10s). In most cases, GRIT completes in under 10 seconds, which for our UAV setup is the time it takes to translate between waypoints. This is essential for in-field path repair as excess time used to compute new routes leaves less battery available to complete these routes - limiting re-routing possibilities. Furthermore, routes that take a long time to replan may not even be possible after UAVs have continued their preplanned routes while the new routes are being computed. GRIT can also flexibly be improved based on runtime constraints, by adjusting the number of Tabu iterations performed. We note that while all methods show similar results when given 300s seconds to complete, we believe this time is too long to be practical for online path repair.

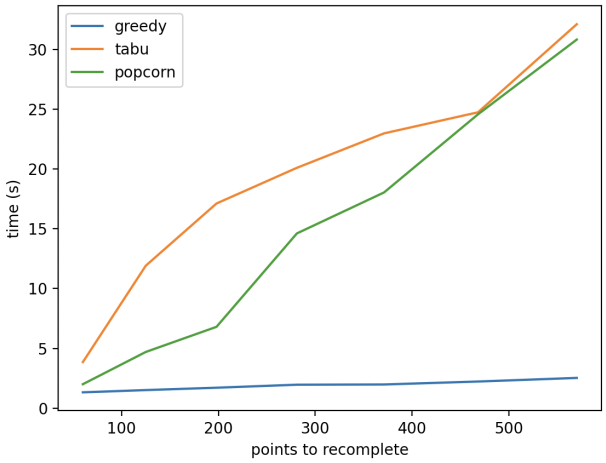

Fig. 7. Average time to replan for paths replanned using greedy repair and POPCORN compared with the runtime of Tabu search until completion. Routes were initially planned with POPCORN. A single UAV fails at a random vertex midway through the route. Recompletion time is plotted as a function of vertices left uncompleted when the robot failed. POPCORN and Tabu search are both far slower than Greedy Search.

Runtime analysis shows that our algorithm is faster than alternative methods (POPCORN, raw Tabu Search), scaling proportionally to the number of vertices unallocated. While each Tabu iteration is costly, the amount of iterations performed in GRIT is flexible and can be tuned to the desired task. Furthermore, just a few Tabu iterations are needed to approach performance of the POPCORN approach or raw Tabu search, both of which take far longer to complete.

VI. Conclusions

We present a new path repair algorithm, GRIT, for multi-robot survey planning. In scenarios where a robot fails midway through its path, GRIT assigns its uncovered vertices to other robots paths so that the survey can still be completed with the remaining robots. The algorithm is significantly faster than an existing integer program based solver, and is also much faster than a Tabu search method used for the disrupted vehicle routing problem. The algorithm quickly re-allocates vertices using an initial greedy solution, Greedy Path Repair, and routes can flexibly be optimized using the same Tabu search it is tested against. In the future, we will explore how to distribute the computation in this method over multiple UAVs communicating with one another on a wireless network.

References

[1] A. Davids. Urban search and rescue robots: from tragedy to technology. *IEEE Intelligent Systems*, 17(2):81–83, 2002.

[2] Caballero F. Martínez-de-Dios J.R. et al. Merino, L. An unmanned aircraft system for automatic forest fire monitoring and measurement. *Intell Robot Syst*, 65(8):533–548, 2011.

[3] Kunal Shah, Grant Ballard, Annie Schmidt, and Mac Schwager. Multidrone aerial surveys of penguin colonies in antarctica. *Science Robotics*, 5(47), 2020.

[4] Katherine S Christie, Sophie L Gilbert, Casey L Brown, Michael Hatfield, and Leanne Hanson. Unmanned aircraft systems in wildlife research: current and future applications of a transformative technology. *Frontiers in Ecology and the Environment*, 14(5):241–251, 2016.

[5] Tauã Cabreira, Lisane Brisolara, and Paulo R. Ferreira Jr. Survey on Coverage Path Planning with Unmanned Aerial Vehicles. *Drones*, 3(1):4, jan 2019.

[6] Ivan Maza and Anibal Ollero. Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In Rachid Alami, Raja Chatila, and Hajime Asama, editors, *Distributed Autonomous Robotic Systems 6*, pages 221–230, Tokyo, 2007. Springer Japan.

[7] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon cellular decomposition. In *Field and service robotics*, pages 203–209. Springer, 1998.

[8] Athanasios Ch Kapoutsis, Savvas A Chatzichristofis, and Elias B Kosmatopoulos. Darp: divide areas algorithm for optimal multi-robot coverage path planning. *Journal of Intelligent & Robotic Systems*, 86(3-4):663–680, 2017.

[9] Federal Aviation Administration. Part 107—small unmanned aircraft systems, 2021.

[10] J Lysgaard R Eglese Q Mu, Z Fu. Disruption management of the vehicle routing problem with vehicle breakdown. *Journal of the Operational Research Society*, 62(4):742–749, 2011.

[11] Juan David Hernández, Eduard Vidal, Jennifer Greer, Romain Fiasco, Patrick Jaussaud, Marc Carreras, and Rafael García. Auv online mission replanning for gap filling and target inspection. In *OCEANS 2017 - Aberdeen*, pages 1–4, 2017.

[12] Ahmet Yazici, Gokhan Kirlik, Osman Parlaktuna, and Aydin Sipahioglu. A dynamic path planning approach for multi-robot sensor-based coverage considering energy constraints. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5930–5935, 2009.

[13] Ling Xu and Anthony Stentz. An efficient algorithm for environmental coverage with multiple robots. In *2011 IEEE International Conference on Robotics and Automation*, pages 4950–4955, 2011.

[14] Anastasia Mavrommati, Emmanouil Tzorakoleftherakis, Ian Abraham, and Todd D Murphey. Real-time area coverage and target localization using receding-horizon ergodic exploration. *IEEE Transactions on Robotics*, 34(1):62–80, 2017.

[15] Ian Abraham and Todd D Murphey. Decentralized ergodic control: distribution-driven sensing and exploration for multiagent systems. *IEEE Robotics and Automation Letters*, 3(4):2987–2994, 2018.

[16] Mahdi Hassan, Daut Mustafic, and Dikai Liu. Dec-ppcpp: A decentralized predator–prey-based approach to adaptive coverage path planning amid moving obstacles. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11732–11739. IEEE, 2020.

[17] Hong Liu, Jiayao Ma, and Weibo Huang. Sensor-based complete coverage path planning in dynamic environment for cleaning robot. *CAAI Transactions on Intelligence Technology*, 3(1):65–72, 2018.

[18] Anton Koval, Sina Sharif Mansouri, and George Nikolakopoulos. Multi-agent collaborative path planning based on staying alive policy. *Robotics*, 9(4):101, 2020.

[19] Zambirinis S. Eglese, R. Disruption management in vehicle routing and scheduling for road freight transport: a review. *TOP*, 26:1–17, 2018.

[20] Amir Ahmadi-Javid and Amir Hossein Seddighi. A location-routing problem with disruption risk. *Transportation Research Part E: Logistics and Transportation Review*, 53:63–82, 2013.

[21] Chunhua Ju, Guanglan Zhou, and Tinggui Chen. Disruption management for vehicle routing problem with time-window changes. *International Journal of Shipping and Transport Logistics*, 9(1):4–28, 2017.

[22] Qianxin Mu and Richard W Eglese. Disrupted capacitated vehicle routing problem with order release delay. *Annals of Operations Research*, 207(1):201–216, 2013.

[23] Xuping Wang, Xu Wu, and Xiangpei Hu. A study of urgency vehicle routing disruption management problem. In *2010 WASE International Conference on Information Engineering*, volume 3, pages 452–455. IEEE, 2010.