Algorithms and Experiments on Routing of Unmanned Aerial Vehicles with Mobile Recharging Stations

Kevin Yu
Department of Electrical & Computer Engineering
Virginia Tech
Blacksburg, VA 24060, USA
klyu@vt.edu

Ashish Kumar Budhiraja
Department of Electrical & Computer Engineering
Virginia Tech
Blacksburg, VA 24060, USA
ashishkb@vt.edu

Spencer Buebel
Department of Electrical & Computer Engineering
Virginia Tech
Blacksburg, VA 24060, USA
stbuebel@vt.edu

Pratap Tokekar
Department of Electrical & Computer Engineering
Virginia Tech
Blacksburg, VA 24060, USA
tokekar@vt.edu

Abstract

We study the problem of planning a tour for an energy-limited Unmanned Aerial Vehicle (UAV) to visit a set of sites in the least amount of time. We envision scenarios where the UAV can be recharged at a site or along an edge either by landing on stationary recharging stations or on Unmanned Ground Vehicles (UGVs) acting as mobile recharging stations. This leads to a new variant of the Traveling Salesperson Problem (TSP) with mobile recharging stations. We present an algorithm that finds not only the order in which to visit the sites but also when and where to land on the charging stations to recharge. Our algorithm plans tours for the UGVs as well as determines the best locations to place stationary charging stations.

We study three variants for charging: multiple stationary charging stations; single mobile charging station; and multiple mobile charging stations. While the problems we study are NP-Hard, we present a practical solution using Generalized TSP that finds the optimal solution that minimizes the total time, subject to discretization of battery levels. If the UGVs are slower than the UAVs, the algorithm also finds the minimum number of UGVs required to support the UAV mission such that the UAV is not required to wait for the UGV. Our simulation results show that the running time is acceptable for reasonably sized instances in practice. We evaluate the performance of our algorithm through simulations and proof-of-concept field experiments with a fully autonomous system of one UAV and UGV.

1 Introduction

Unmanned Aerial Vehicles (UAVs) are being increasingly used for applications such as surveillance [28], package delivery [46], infrastructure inspection [25, 35], environmental monitoring [14], and precision agriculture [10, 48]. However, most small multi-rotor UAVs have limited battery lifetime (typically < 30 minutes) which prevents them from being used for long-term or large scale missions. There is significant work that is focused on extending the lifetimes of UAVs through new energy harvesting designs [31], automated battery swapping [49], low-level energy-efficient controllers [17], and low-level path planning [30]. In this paper, we investigate the complementary aspect of high-level path planning with an emphasis on energy optimization.

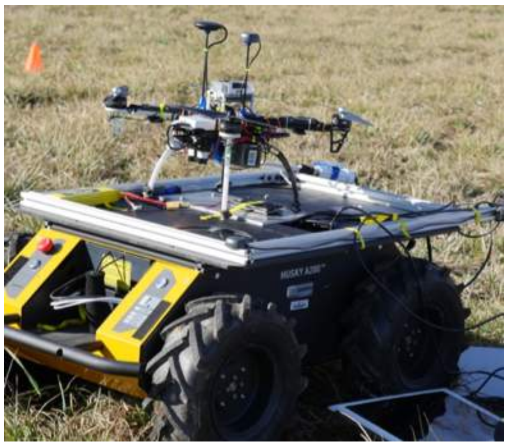


Figure 1: DJI F450 with Pixhawk autopilot running APM with onboard Jetson TX1 along with the Clearpath Husky UGV used for the field experiments.

This work is motivated by persistent monitoring applications [41] where the UAVs are tasked with monitoring a finite set of sites on the ground by flying above these sites. The objective is to minimize the time required to visit all sites. In the absence of any additional constraints, this can be formulated as Traveling Salesperson Problem (TSP), which is a classic optimization problem [7]. However, when the sites are located far apart, the UAV may not have enough battery capacity to fly the entire tour. We consider scenarios where the UAVs are capable of landing on recharging stations and then taking off and continuing the mission. The recharging stations can either be stationary or placed on Unmanned Ground Vehicles (UGVs) which can charge the UAV while simultaneously transporting it from one site to another (Figure 1). This leads to a new variant of TSP where the output is not only a path for the UAV, but also a charging schedule that determines where and how much to recharge the UAV battery as well as paths for the UGVs. For a single UGV to keep up with the UAV, we also study the problem of minimizing the number of UGVs. Since it is not always possible for the UGV to keep up with the UAV's speed we implement a solver that can find the minimum number of UGVs necessary to service the UAV.

This problem generalizes Euclidean TSP [7] and is consequently NP-Hard. As such, assuming P$\neq$NP, no algorithm can guarantee the optimal solution in polynomial time. Instead, we seek algorithms that find the optimal solution in reasonable time for practical instances, similar to recent work [27, 47]. Our main contribution is to show how to formulate both problems, described in section 3, as Generalized TSP (GTSP) [33] instances. Earlier work have shown that a GTSP-based algorithm finds solutions faster than an Integer Programming approach [42, 27, 47]. We empirically evaluate two approaches to solve the GTSP instances: (1) Generalized Large Neighborhood Search (GLNS) solver [42] which uses heuristics to find potentially sub-optimal solutions in short time; and (2) an exact solver which reduces GTSP into TSP instances which are solved using concorde [6]. We compare the time required to find the solution in both approaches. We implemented this algorithm on a custom-built UAV and Clearpath Husky UGV (Figure 1) and carried out field experiments to demonstrate the efficacy of our algorithm.

This work is a direct extension of our preliminary work to be presented at ICRA '18 [50]. This paper builds on the prior work with larger scale experiments and fully autonomous operations including automatic take-off and landing on the UGV, as described in Section 6. In contrast to this work, the preliminary experiments in [50] had a manually operated UGV, manual landing of the UAV, and no battery swapping.

The rest of the paper is organized as follows. We begin with a discussion of the related work in Section 2. We describe our problem formulation and assumptions in Section 3. The algorithm is described in Section 4.

We evaluate the performance of the algorithm through simulations reported in Section 5 and through proof-of-concept experiments reported in Section 6. We conclude with a discussion of future work.

2 Related Work

In this section we briefly describe the work related to UAV recharging stations.

2.1 Recharging and Replacing UAV Batteries

A number of solutions for autonomous charging of UAVs have been proposed in the recent past. Cocchioni et al. [9] presented a vision system to align the UAV with a stationary charging station. A similar design was presented by Mulgaonkar and Kumar [32] including magnetic contact points which improved the landing of the UAV. There are also commercial products (*e.g.*, the SkySense system [5]) that provide similar capabilities.

The alternative to recharging a discharged battery is to swap it with a charged one. Swieringa et al. [45] presented a "cold" swap system for exchanging the batteries for one or more helicopters, which is where the system is turned off and the battery is swapped out. The authors evaluated their system through simulations with three helicopters where they demonstrated an increase in system lifetime from six minutes to thirty two minutes. Toksoz et al. [49] presented the design of a stationary battery swapping station for multi-rotor systems. Their design has a "dual-drum structure" that can hold a maximum of eight batteries which can be "hot" swapped. "Hot" swapping being the changing of batteries without shutting down the system.

Kemper et al. [18] investigated the needs of consumers and design constraints for battery recharging stations for helicopters. They performed a cost-analysis of existing designs and consumer needs for recharging and swapping of batteries. Suzuki et al. [44] followed up on this work by further analyzing two types of landing platforms proposed in [18].

The work presented in this paper is complementary to these hardware designs — any of the existing systems could be leveraged. Instead we show how to optimize the performance by careful placement of charging stations or planning of paths for mobile charging stations.

2.2 Planning for Energy Limited UAVs

A typical strategy to deal with limited battery life of UAVs is to use multiple robots with possible redundancy built in. Derenick et al. [11] presented a control strategy to carry out persistent coverage missions with robot teams which balances a weighted sum of mission performance and the safety of the UAVs. The UAVs reconfigure based on their energy levels and coverage performance. Mitchell et al. [29] presented an online approach for maintaining formations while substituting UAVs running low on charge with recharged UAVs.

There have been work on planning the paths for UAVs using multiple stationary recharging stations [19, 39, 4]. Kim et al. [19] used Mixed Integer Linear Programing (MILP) to allow teams of robots to trade off the task objective within the team when some UAVs are low on energy. Shakhatreh et al. [39] studied the problem of maintaining persistent coverage of an area by partitioning the coverage tour amongst the UAVs. Ahmed et al. [4] presented a method to discretize the state space which allows for an algorithm to minimize the energy used by UAVs for longer missions. Liu and Michael [24] presented a matching algorithm for assigning UAVs with UGVs acting as recharging stations.

Our work falls broadly in the category of area coverage with energy-limited UAVs. Franco et al. [12] presented an algorithm that reduces the energy consumption of a UAV while satisfying coverage and resolution requirements. Sipahioglu et al. [40] considered a similar problem. Their algorithm first finds a route that ensures complete coverage and then partitions the route among multiple robots by considering respective energy capacities. Most recently, Wei and Isler [1] presented a approximation algorithm for covering a polygonal grid environment where the UAV is allowed to return back to a fixed basestation multiple times for recharging. However, they do not address the problem of optimizing the recharging locations.

In our previous work [48], we showed how to plan tours for a symbiotic UAV+UGV where the UGV can mule the UAV between two deployment locations such that the UAV does not spend any energy. However, the previous work did not model the capability of UGV recharging the UAV along the way. Consequently, the goal was to maximize the number of sites that can be visited in a single charge. This results in a variant of an NP-Hard problem known as orienteering [8]. In the current work, we allow for a more general model which has the added complication of keeping track of the energy level of the UAV as well as deciding where and how much to charge along the tour.

The work by Rathinam et al. [43] looks into how to plan paths for mobile charging stations. The authors of this paper study the problem of planning UAV paths with fuel constraints and stationary refueling locations. The algorithms that the author presents allow for the planning of multiple UAVs to visit every site. This paper is slightly different than ours in the sense that the refueling stations are stationary and the UAV has to adjust to the refueling station. In our paper we allow the refueling station to be mobile and adjust for the UAV allowing for shorter tour times.

The work most closely related to ours is that of Maini and Sujit [26]. They present an algorithm that plans paths for one UAV and one recharging UGV to carry out surveillance in an area. The UGV moves on a road network. The authors create an initial path for the UGV and then create a path for the UAV. In this paper, we simultaneously create paths for the UAV and UGV. We compare the empirical performance of our algorithm with that from [26] in Section 5. We find that our algorithm outperforms this baseline. Additionally, we guarantee that our algorithm finds the optimal solution for the problem, conditioned on the discretization unlike that from [26]

2.3 Vehicle Routing Problems

Our algorithm is based on a solution to the Generalized Traveling Salesperson Problem. Both, GTSP and TSP, belong to the class of vehicle routing problems [21]. Variants of TSP have been used for many robotic applications such as routing [21], surveillance [38], and patrolling [36]. Solutions to GTSP have been used in applications such as location-routing, loop material flow system design, post-box collection, stochastic vehicle routing and arc routing [22].

Both, TSP and GTSP are NP-Hard problems. If the costs are Euclidean distances, then there exists an efficient polynomial time approximation scheme for solving TSP [33]. However, there does not exist a constant-factor approximation for solving Euclidean GTSP. Nevertheless, a number of solvers have been developed that can find good solutions for TSP and GTSP in practice [6, 42]. In this paper, we use two state-of-the-art solvers, concorde [6] and GLNS [42], for solving TSP and GTSP respectively.

2.4 Autonomous UAV Landing

There are many studies on controls and planning techniques for autonomous landing of aerial vehicles. Kong et al. [20] surveyed various autonomous landing methods for five classes of aerial vehicles (full-scale, medium-scale, small-scale, mini-scale, and micro). A common approach is to use vision-based autonomous landing [23, 16, 34]. Oh et al. [34] presented a technique for landing on a swaying ship in the ocean. Kim et al. [20] presented the design of a system that can land on a mobile platform using color-based detection with an RGB camera. This approach may not be robust due to inconsistent lighting conditions. Instead, we use an IR-camera along with IR beacon for detection and landing on the recharging station. This approach is robust to poor lighting conditions, indifferent to color, and can deal with other IR sources when the exposure is adjusted correctly.

Goldin [15] and Dougherty [13] investigated the ability for quadrotors to land on inclined surfaces (i.e., perching). The system proposed by Dougherty [13] used downward-facing laser sensors to measure the distance to the landing platform and to detect the angle of the platform. The system was demonstrated to work for platforms with up to 30° incline. Goldin [15] presented a completely autonomous system that allows for real-time simultaneous localization and mapping for perching with UAVs.

Our main novelty is in jointly optimizing the routes for the UAV as well as the mobile recharging stations (equivalently, placement of stationary recharging stations). The exact formulation is defined in the next section.

3 Problem Formulation

The input to our algorithm is a set of n sites, x_i, that must be visited by the UAV. We use x_i to refer to a site as well as the vector representing its coordinates. We start with a list of common assumptions:

1. unit rate of discharge (1% per second);
2. UAV has an initial battery charge of 100%;
3. UAV and UGVs start at a common *depot*, d;
4. all the sites are at the same altitude;
5. UGVs have unlimited fuel/battery capacity;
6. UAV can fly between any two sites if it starts at 100% battery level.

All but assumption 5 are only for the sake of convenience and ease of presentation and can be easily relaxed. Although UGVs cannot have unlimited operational time, it is a reasonable assumption since UGVs can have much larger batteries or can be refueled quickly. In this paper, we only allow for the UAV to be recharged at a site or while being ferried between two sites by the UGV. It is possible to extend the algorithm to also allow for recharging at arbitrary sites, as we discuss in Section 7.

We also provide a list of standard terminology that will be used throughout this paper:

- x_i denotes the i^{th} site that must be visited[1] by flying to a fixed altitude;
- r represents the time required to recharge the battery by a unit %;
- t_{TO} is the time it takes to take off from the UGV;
- t_L is the time it takes to land on the UGV;
- $t_{UAV}(x_i, x_j)$ and $t_{UGV}(x_i, x_j)$ give the time taken by the UAV and UGV to travel from x_i to x_j.

Suppose Π is a path that visits the sites in the order given by $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ where $\sigma(j) = i$ implies x_i is the j^{th} point visited along Π. The cost of an edge from $x_{\sigma(j)}$ to $x_{\sigma(j+1)}$ along Π depends on whether the UAV flies between the two sites or if it is muled by the UGV between the two sites while being

[1]Note that x_i does not mean that is the i^{th} point that will be visited. The order of visiting the points is determined by the algorithm.

recharged (denoted as a Type II edge, defined formally in Section 4.1). Let k and k' be the battery levels at $\sigma(j)$ and $\sigma(j+1)$. Therefore,

$$T(j,j+1) = \begin{cases} t_{UAV}(x_{\sigma(j)}, x_{\sigma(j+1)}) \\ \max\{t_{UGV}(x_{\sigma(j)}, x_{\sigma(j+1)}), r(k'-k)\} \end{cases} \tag{1}$$

In addition, we also have non-zero node costs if the UAV is charged from battery level k to k' at a site x_i rather than along an edge:

$$T(j) = r(k'-k). \tag{2}$$

Therefore, the total path cost is given by,

$$T(\Pi) = T(1) + \sum_{j=1}^{n-1} T(j+1) + T(j,j+1) \tag{3}$$

We are now ready to define the problems studied in this paper.

Problem 1 (Multiple Stationary Charging Stations (MSCS))**.** *Given a set of sites, x_i, to be visited by the UAV, find a path Π^* for the UAV that visits all the sites as well as selects one or more sites (if needed) to place recharging stations so as to minimize the total time given by Equation 3 under the assumptions 1–6 given above.*

Problem 2 (Single Mobile Charging Station (SMCS))**.** *Given a set of sites, x_i, to be visited by the UAV, find a path Π^* for the UAV that visits all the sites as well as another path for the UGV acting as a mobile basestation so as to minimize the total time given by Equation 3 under the assumptions 1–5 given above. Assume that the UAV and UGV travel at the same speed.*

The assumption that the UGV is as fast as the UAV is not necessary to find a solution; it is required to guarantee optimality for one UGV. If the UGV is slower than the UAV, we can still use the paths returned by the algorithm for one UGV, but the UAV may have to wait. An alternative is to minimize the number of UGVs required to ensure the UAV never has to wait for a recharging station.

Problem 3 (Multiple Mobile Charging Stations (MMCS))**.** *Given a path, Π^*, for a UAV and a set of charging sites as well as* Type II *edges for the UGVs, find the minimum number of slower UGVs necessary to service the UAV, without the UAV having to wait for a UGV under the assumptions 1–5 given above. The input to MMCS is obtained by solving SMCS, without assuming the single UGV in SMCS is as fast as the UAV.*

Our main contribution is a GTSP-based algorithm that solves the first two problems optimally and an Integer Linear Programming (ILP)-based algorithm that solves Problem 3. As mentioned previously, Problems 1 and 2 are NP-Hard and consequently finding optimal algorithms with running time polynomial in n is infeasible under standard assumptions. Instead, we provide a practical solution that is able to solve the three problems in reasonable time (quantified in Section 5).

4 GTSP-Based Algorithm

In this section we show how to formulate Problems 1 and 2 as GTSP instances [33]. We first formally define GTSP. Let G be a graph (can be directed) with vertices V and edges E. Each edge $e \in E$ has a corresponding cost c. The vertices are grouped into m mutually exclusive clusters. An edge exists only between vertices belonging to different clusters. The Generalized Traveling Salesperson Problem asks for a minimum cost cycle which includes exactly one vertex from each cluster. When each cluster contains only one vertex, the GTSP reduces to TSP.

Solving GTSP is at least as hard as solving TSP. However, Noon and Bean [33] presented a technique to convert any GTSP input instance into an equivalent TSP instance on a modified graph such that finding the optimal TSP tour in the modified graph yields the optimal GTSP tour in the original graph. We can solve GTSP by solving TSP optimally using a numerical solver and we use *concorde* [6], which is the state-of-the-art TSP solver or GLNS [42], that is a heuristics-based GTSP solver. The results in Section 5 show that GLNS is significantly faster than *concorde*. However, only the *concorde* approach is guaranteed to find the optimal solution.

We start by showing how to formulate the SMCS and MSCS problems as GTSP instances. After obtaining an output, we can convert the TSP solution back into a GTSP solution, then into a solution for the SMCS or MSCS problems. The process of converting SMCS and MSCS into TSP is the same. Only the process of converting the solution of TSP to solutions of SMCS and MSCS differ.

4.1 Transforming SMCS/MSCS to GTSP

Given an SMCS or MSCS instance, we show how to create a GTSP instance consisting of a directed graph where the vertices are partitioned into non-overlapping clusters. We create one cluster, g_i, for each input site x_i. Each cluster, g_i has m vertices, each one corresponding to a discretized battery level. That is, $g_i = \{x_i^k \mid \forall i \in [1:n], \forall k \in \{1, 2, \ldots, m\}\}$. x_i^k represents the UAV reaching site x_i with $k\dfrac{100\%}{m}$ battery remaining. m is an input discretization parameter. Figure 2 shows the six clusters for six input sites with $m = 5$.

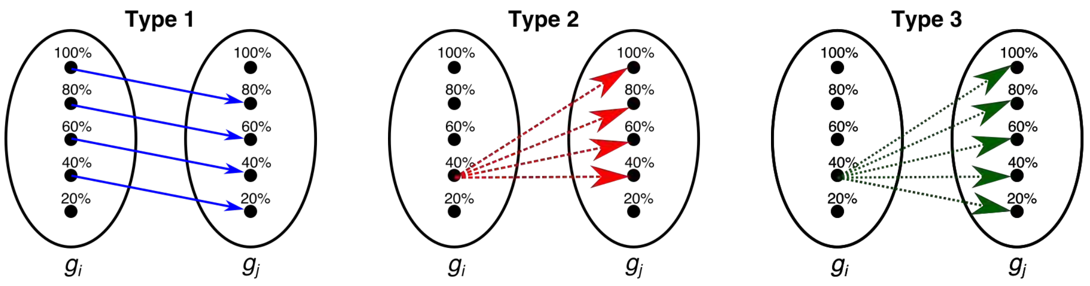


Figure 2: Different types of edges that are created. Type I are the edges where the UAV flies directly between the two sites and the battery level strictly decreases. Type II are the edges where the UGV carries and recharges the UAV. Lastly, Type III are the edges where the UAV and UGV meet up to charge at a site. The depot station d is shown on the graph. Note that only a subset of all possible edges are shown.

Next we describe how to create the edges amongst the vertices in the n clusters. We create three types of edges. Type I edge between x_i^k and $x_j^{k'}$ models the case where the UAV directly flies from x_i to x_j. The cost of a Type I edge is given by:

$$T_{\text{I}}(x_i^k, x_j^{k'}) = t_{UAV}(x_i, x_j)$$

A Type I edge exists between x_i^k and $x_j^{k'}$ if and only if $k - k'$ equals the distance between x_i and x_j. For ease of exposition, we assume that taking-off and landing energy consumption is negligible. Nevertheless, we can easily incorporate this in the edge definitions. These types of edges are shown in Figure 2-left.

A Type II edge from x_i^k to $x_j^{k'}$ models the UAV landing on the UGV at x_i and recharging while being muled to x_j by the UGV. The cost of a Type II edge is given by:

$$T_{\text{II}}(x_i^k, x_j^{k'}) = \max(r(k' - k), t_{UGV}(x_i, x_j)) + t_{TO} + t_L$$

The cost is the maximum of the time taken to recharge from k to k' and the time it takes the UGV to travel from x_i to x_j. Note that a Type II edge exists only if $k' \geq k$. Type II edges are shown in Figure 2-middle.

Finally, we have Type III edges that represent the UAV flying from x_i to x_j and then landing on the UGV at x_j and recharging up to k' battery level. The cost of a Type III edge is given by:

$$T_{\mathrm{III}}(x_i^k, x_j^{k'}) = t_{UAV}(x_i, x_j) + r(k' - k + ||x_i - x_j||_2) + t_{TO} + t_L$$

A Type III edge exists if and only if $k' \geq k - ||x_i - x_j||_2$. Figure 2-right shows the Type III edges.

Only Type I and Type III edges exist when solving MSCS whereas all three edges are possible when solving MMCS. Note that Type II and Type III edges require the UAV to take off and land at every site. This prevents the UAV from not taking off between two consecutive Type II edges. This is because in order to visit a site it must fly to a fixed altitude to consider the site visited.

There are certain pairs of vertices for which more than one type of edge may be allowed. In such a case, we pick the minimum of the three edge costs (assuming the edge cost is ∞ if the edge does not exist) and assign the minimum cost for the edge. That is, the edge cost $T(x_i^k, x_j^{k'})$ is given by:

$$T(x_i^k, x_j^{k'}) = \min\{T_{\mathrm{I}}(x_i^k, x_j^{k'}), T_{\mathrm{II}}(x_i^k, x_j^{k'}), T_{\mathrm{III}}(x_i^k, x_j^{k'})\}$$

We also create an $n + 1^{th}$ cluster containing a dummy vertex called the depot, d. We add a zero cost edge from d to all vertices, x_i^k, with $k = m$ and edges from all vertices back to d. The reason to create a depot node is that the TSP solver finds a closed tour whereas we are interested in finding paths.[2] The depot node serves to ensure that we can find a closed tour without charging for the extra edges.

4.2 Converting Optimal TSP Tour to UAV and UGV Paths

An optimal TSP tour immediately yields an optimal GTSP solution. The order in which the clusters are visited gives the sequence of vertices on the UAV paths. What remains is deciding the UGV path for SMCS and recharging station placements for MSCS.

In MSCS, we only have Type I and Type III edges. If a Type III edge, say from x_i^k to $x_j^{k'}$, appears in the GTSP solution, then we will place a recharging station at the site x_j. No recharging stations are placed for Type I edges in the solution.

In MMCS, all three edges are possible, whereas only Type I and Type II in SMCS. We check the type of each edge in the GTSP solution, one by one. If a Type I edge appears in the GTSP solution, then it does not affect the UGV tour. If a Type II edge, say from x_i^k to $x_j^{k'}$, appears in the GTSP solution, we add x_i and x_j to the UGV path (in this order). If a Type III edge, say from x_i^k to $x_j^{k'}$, appears in the GTSP solution we add only x_j to the UGV path. The UGV path, as a result, visits a subset of the input sites. If the UGV is slower than the UAV, then it is possible that the UAV will reach a site before the UGV does and will be forced to wait. We implement an ILP that allows us to solve for the minimum number of UGVs necessary to service the UAV without waiting (shown in section 4.3).

Theorem 1. *The GTSP-Based algorithm finds the optimal solution for SMCS and MSCS assuming that there exists an optimal TSP solution.*

The proof follows directly from the proof of optimality of the GTSP reduction given by Noon and Bean [33].

Problem 2 assumes that the UGV is as fast as the UAV, and therefore only one UGV suffices. Next, we show how to address the case when the UGV is slower than the UAV. Multiple UGVs may be required. We show how to minimize the number of UGVs.

[2]A path visits a vertex exactly once whereas a tour has the same starting and ending vertices.

4.3 Solution for Problem 3

We present a solution to the MMCS problem based on an ILP formulation. The input is obtained by solving Problem 2, where we are given a UAV path and a set of UGV sites and Type II edges. The UGV path visits only a subset of the sites in $\{x_i\}$. We denote these sites by $\{g_1, g_2, \ldots, g_l\}$, where $l \leq n$. For each directed edge from g_i to g_j, where $j > i$, we associate a binary decision variable y_{ij}. The solution to Problem 3 will assign binary values to all y_{ij}. If y_{ij} equals 1, then some UGV will travel from g_i to g_j. If y_{ij} equals 0, then no UGV will travel from g_i to g_j. Note that j does not necessarily have to be $i+1$ but must satisfy $j > i$.

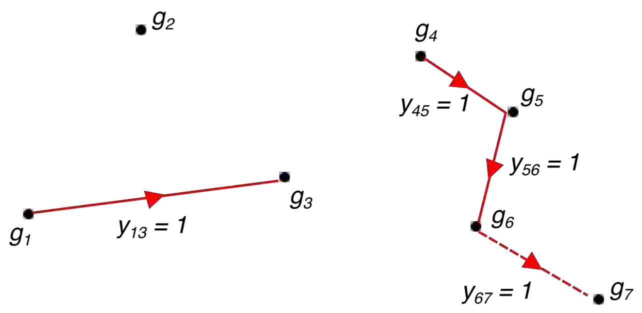


Figure 3: Example case of UGV path with site and edge labels. The binary decision variables that are equal to 1 are shown; all other y_{ij} are returned as 0 by the solver. g_2 has no incoming or outgoing edge selected and requires a separate UGV. g_1 and g_4 have only one outgoing edge selected and therefore indicate the start of a new UGV path. Similarly, g_3 and g_7 have only one incoming edge and are therefore the end of a UGV path. g_5 and g_6 have incoming and outgoing edges and are therefore in the middle of a path. The edge between g_6 and g_7 is Type II and therefore y_{67} is forced to be selected always.

If there is a site g_e such that a solution has one incoming edge $y_{df} = 1$ and one outgoing edge $y_{ef} = 1$, then the same UGV can be used for both edges (Figure 3). That is, the site g_e is in the middle of some UGV path. If there is a site g_c such that the solution has an incoming edge $y_{ac} = 1$ or a site g_a such that the solution has an outgoing edge $y_{ac} = 1$ but not both, then the corresponding UGV path either ends or starts at g_c or g_a respectively. If there is a site g_b such that a solution has no incoming or outgoing edge selected, then the corresponding UGV path visits only g_b and no other site. The number of distinct UGV paths selected is given by l minus the total number of edges selected. Therefore, to minimize the number of UGVs required, we maximize the number of edges selected.

Also associated with each edge is the time to come for the UAV and UGV denoted by $T^{\Pi^*}(g_i, g_j)$ and $t_{UGV}(g_i, g_j)$ respectively. Here $T^{\Pi^*}(g_i, g_j)$ is the time taken by the UAV to fly the subpath of Π^* from g_i to g_j, which may contain some intermediate sites. $t_{UGV}(g_i, g_j)$, on the other hand is the time for the UGV to directly go from g_i to g_j.

Using the above notation we present the following ILP formulation:

$$\max \sum_{i=1}^{l-1} \sum_{j=i+1}^{l} y_{ij} \tag{4}$$

subject to:

$$\sum_{i=1}^{j-1} y_{ij} \leq 1 \ \forall \ j, \tag{5}$$

$$\sum_{j=i+1}^{l} y_{ij} \leq 1 \ \forall \ j, \tag{6}$$

$$y_{ij} = 0 \text{ if } T^{\Pi^*}(g_i, g_j) < t_{UGV}(g_i, g_j), \text{ and} \tag{7}$$

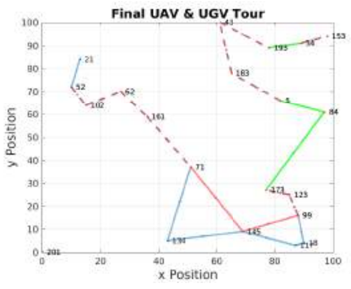
$$y_{ij} = 1 \text{ if } y_{ij} \text{ is Type II edge.} \tag{8}$$

Equation 5 and 6 only allow a maximum of one incoming edge and a maximum of one outgoing edge. The constraint given by Equation 7 removes all UGV edges where the UAV would have to wait for the UGV. Lastly Equation 8 forces our problem to use Type II edges if present.

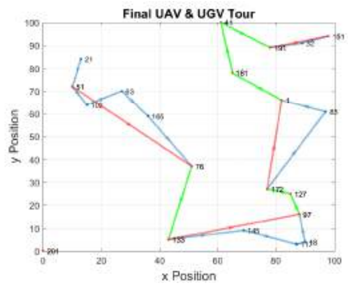
5 Simulations

In this section, we present simulation results using the proposed algorithm. The implementation is available online.[3] We conduct four types of simulation studies. First, study the effect of the input parameters and computational time for SMCS (MSCS has similar computational cost as SMCS). Next, we compare the performance of our algorithm for SMCS with that of [26]. Finally, we present simulation results for MMCS.

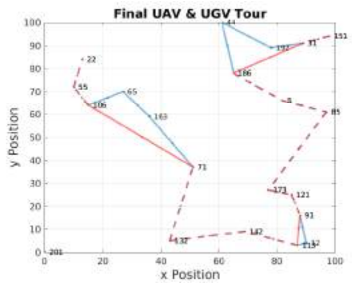
5.1 Effect of the Parameters for SMCS



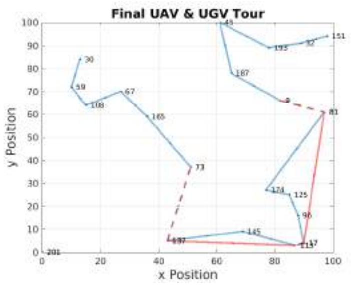
(a) UAV Tour and Cluster {0,0,0,1} in order of visit.



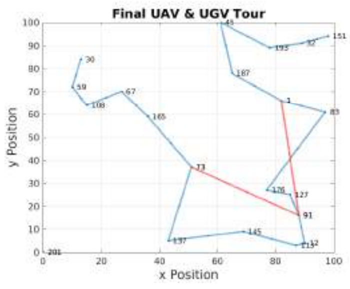
(b) UAV Tour and Cluster {0,0,0,4} in order of visit.



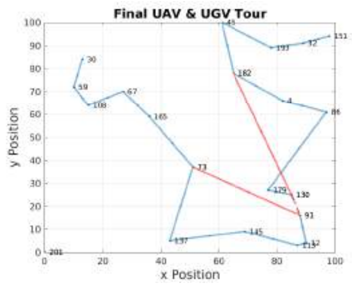
(c) UAV Tour and Cluster {0,0,4,1} in order of visit.



(d) UAV Tour and Cluster {4,4,0,1} in order of visit.



(e) UAV Tour and Cluster {4,4,0,4} in order of visit.



(f) UAV Tour and Cluster {4,4,4,4} in order of visit.

Figure 4: The above figures are multiple runs using the same initial points. We use 20 sites with 10 battery levels. Each figure has the vertex number in the GTSP input graph and a set in the form {W,X,Y,Z} in the caption. This set {W,X,Y,Z} denotes: $t_{TO} = W$, $t_L = X$, $r = Y$, and $t_{UGV} = Zt_{UAV}$. The colors represent different edge types with blue being only UAV travel, red being only UGV travel, green being UAV and UGV travel separate, and dashed red being UAV+UGV travel together.

Figure 4 shows the outputs obtained for different configurations of the t_{TO}, t_L, r, t_{UGV} parameters for the same 20 input sites and with $m = 10$ battery levels. Each figure has the UAV+UGV tour with blue solid edges (only UAV), red solid edges (only UGV), green solid edges (UAV and UGV separate), and red dashed edges (UAV+UGV together).

We make the following intuitive observations using the six cases shown in Figure 4:

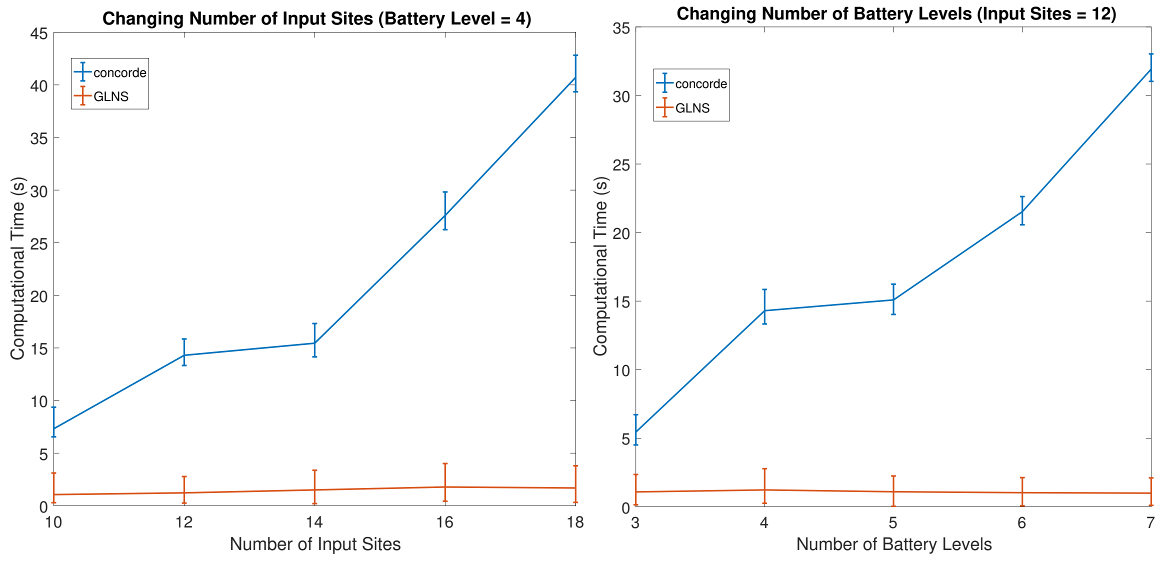
[3]https://github.com/raaslab/Heterogeneous-Teams

- $t_{TO} = 0$, $t_L = 0$ and $r = 0$: UAV does not differentiate between the type of the edge because there is no penalty to recharge (Figure 4a);
- $t_{TO} + t_L > 0$: recharging has a penalty and as such the number of recharging stops are reduced (Figures 4d, 4e and 4f);
- $t_{TO} = 0$, $t_L = 0$, $r = 0$ and $t_{UGV} > t_{UAV}$: the UAV will use Type III edges for charging because t_{UGV} will make Type II edges higher cost (Figure 4b and 4e);
- $t_{TO} = 0$, $t_L = 0$, $r > 0$ and $t_{UGV} = t_{UAV}$: the UAV will use Type II edges for charging instead of Type III edges (Figure 4c);

We observe that the recharge time r and UGV speed t_{UGV} affect which type of edges are used. If the time it takes to recharge is much larger than t_{UGV} then the UAV will favor Type II edges and when the time it takes to recharge is much less than t_{UGV} then the UAV will favor Type III edges.

5.2 Computational Time for SMCS

We use two solvers for the SMCS (and MSCS) problem. When using *concorde* we obtain an optimal solution, but with more computational time. Therefore, it can only solve smaller instances. The GLNS solver can solve larger instances, but cannot always guarantee optimality. Nevertheless, we observe that the GLNS was able to find the optimal solution for all of the cases reported in Figure 5.



(a) Computational time with battery levels set to 4. (b) Computational time with input sites set to 12.

Figure 5: Comparison of computational times of the GTSP to TSP transformation [33] solved using concorde and the direct GTSP solver, GLNS [42] on the default "medium" setting. The output costs for the concorde and GLNS solutions were the same for the given instances, but may differ for larger instances. This was done over 10 trials.

Figure 5 shows a direct comparison of the computational times of the two methods. We compared the two methods by first varying the amount of input sites (Figure 5a), with $m = 4$, and then comparing the two by varying the amount of battery levels (Figure 5b), given $n = 12$. We ran 10 trials with random input sites. We plot the average value along with the maximum and minimum value.

Due to limitations in *concorde*, we were not able to run larger instances, but GLNS can run larger instances, as shown in Figure 6. We show the effect of incrementing n from 20 to 50 and m from 50 to 150 in Figure 6.

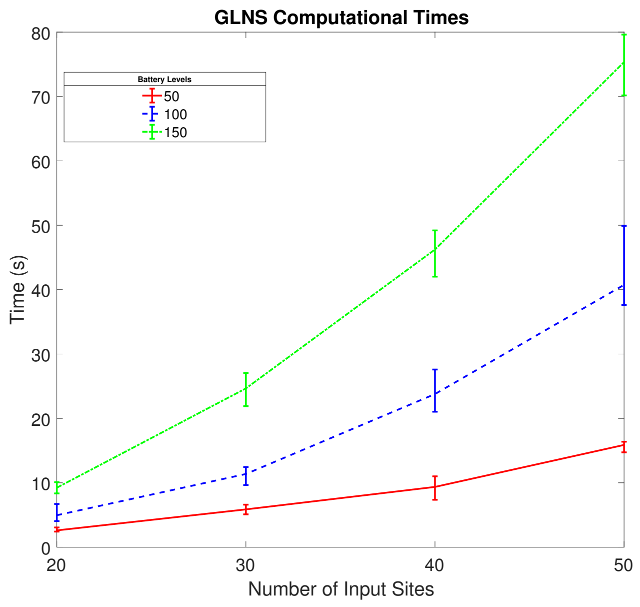


Figure 6: The computational time of GLNS for larger instances. This was done for 5 random instances with the average, max, and min plotted as well. These problem instances could not be solved by concorde.

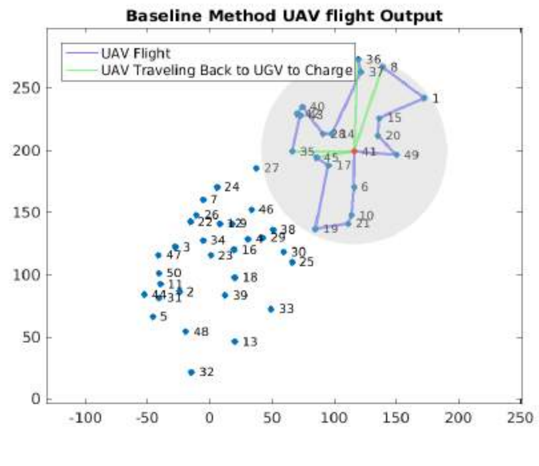
We ran 5 random instances and plot the average of the 5 instances with the maximum and minimum value. The simulations show that GLNS is able to solve larger sized instances in reasonable amount of time.

5.3 Comparison With Baseline Algorithm for SMCS

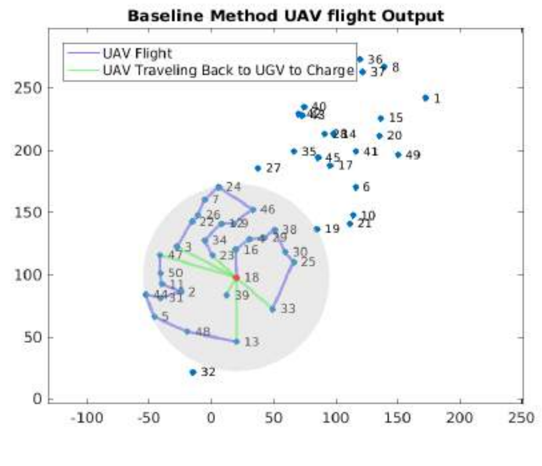
We compare our proposed GTSP-based algorithm with the method presented by Maini and Sujit [26]. We refer to the algorithm presented in [26] as the baseline method. There are some minor differences in the two problem formulations. Our method proposed overcomes many of the limitations in the baseline method. In the baseline method, the UGV is restricted to navigate only on a given road network which is assumed to be a tree. The UGV must remain stationary while recharging the UAV in the baseline method whereas we allow for the UGV to recharge the UAV while carrying it to the next deploying site. Furthermore, it is implicitly assumed that the UAV's battery is fully recharged every time it lands on the UGV whereas we provide as output the charging schedule as well. Our method keeps track of the UAV's battery level, thereby allowing it to recharge at any level (not necessarily when fully depleted) and potentially not recharge to 100%, if it is not needed. As a result, our method produces paths that visit the set of input sites in lesser time. We present the empirical results next.

Figure 7 shows a sample output from the baseline method. The method starts by discretizing the edges of the UGV roadmap. A disk of radius $B/2$ is placed at each discretized point, where B is the energy budget (assuming unit rate of discharge and unit speed). A hitting set problem is then solved to find the minimum number of disks required to cover all the sites that need to be visited by the UAV. The UAV then visits all the sites within each chosen disk, while the UGV waits at the center of the disks. The UAV may return to the center multiple times, if the tour within a disk is longer than B. The UGV's path is found using depth-first search over the roadmap to visit all the chosen disk centers.

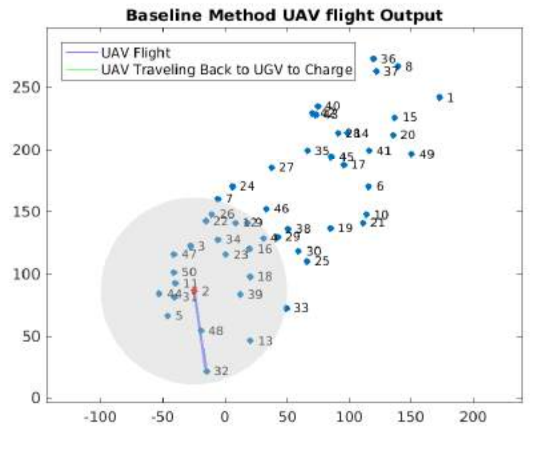
Since our algorithm does not take a UGV roadmap as input, we create such a roadmap by constructing a minimum spanning tree of all the input sites. We use concorde to solve for the UAV tours within each disk and add detours to visit the center if the tour exceeds the budget B (green edges in Figure 7). We find the minimum hitting set by greedily choosing the disk that contains the most uncovered sites and continuing until all sites are covered. In addition to using depth-first search for finding the UGV path, we also find a



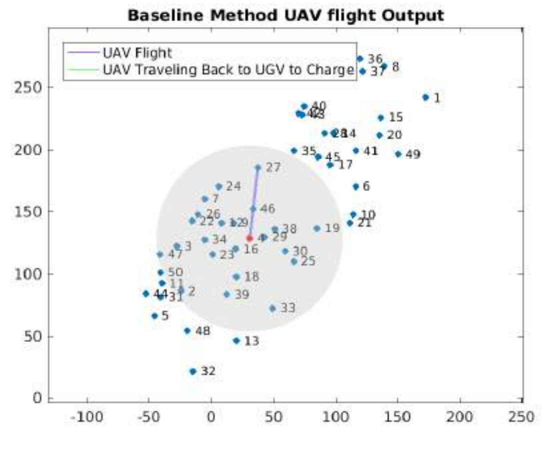
(a) First UAV flight.



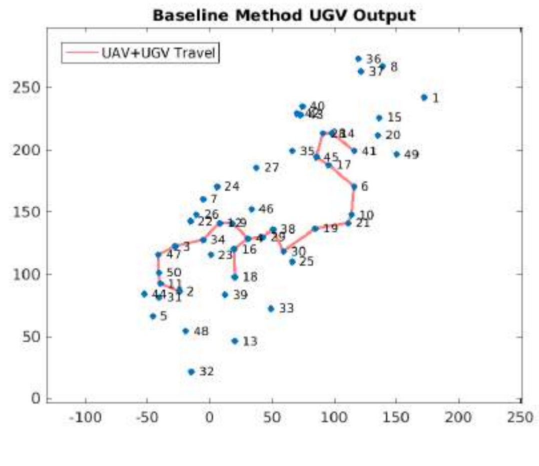
(b) Second UAV flight.



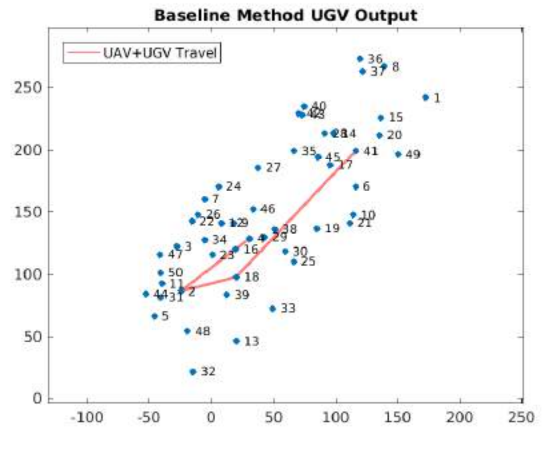
(c) Third UAV flight.



(d) Fourth UAV flight.

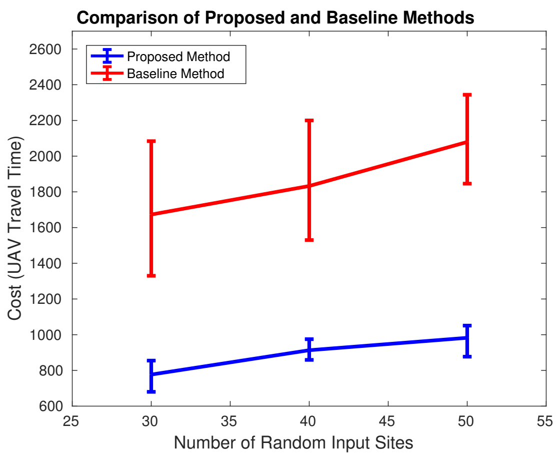


(e) UGV path using DFS.

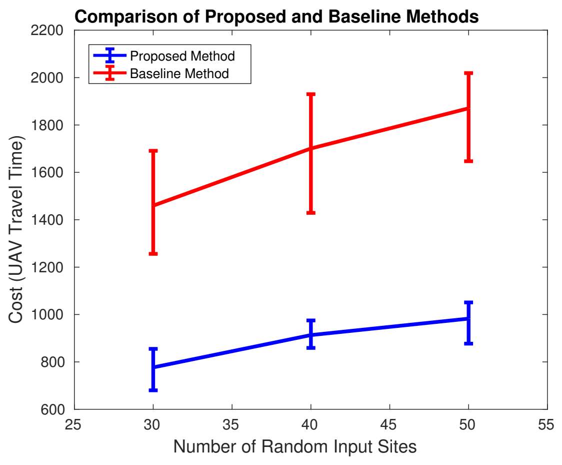


(f) UGV path using TSP.

Figure 7: Representative output of the baseline method [26]. The number of sites randomly chosen were 50 with 100 battery levels and 150 meter budget (assuming unit rate of discharge and unit speed).



(a) Depth First Search method.



(b) TSP method.

Figure 8: Comparison between the method proposed in this paper and the baseline method [26]. We also consider a variant of the baseline method that uses a TSP solver to calculate the UGV path instead of depth first search. All graphs show mean and variance of 10 random trials. We kept the number of battery levels the same at 100 and allow an energy budget of 150 meters.

TSP tour of the chosen disk centers which results in shorter paths. Nevertheless, the baseline method still results in longer tours as compared to our method.

Figure 8 shows the cost of tours generated by the baseline method (with DFS and TSP) and our method. The figures show the total cost of the UAV tour which includes the UAV flight time as well as the time it would take to return to the UGV, land, recharge, take-off, and then return to the path. The sites are generated randomly in an area of 300×110 meters. The energy budget is set to $B = 150$. The baseline method returns path that take almost twice as long as our method. We attribute this to the fact that we allow the UGV to recharge while carrying the UAV and not always recharge to 100% or recharge only when completely depleted.

5.4 Number of UGVs required in MMCS

For Problem 3, we study the effects of a slower UGV on the number of UGVs required to service the same set of sites. The ILP was solved using the Integer Programming toolbox in MATLAB. Figure 9 shows the minimum number of UGVs necessary to service a single UAV as a function of the relative speeds. We used the same $m = 50$ data set that was used to create Figure 6. As expected, the ILP solution uses only one UGV when the UAV and UGV have equal speeds. As the relative speed of the UGV decreases, more UGVs are required on an average. The actual trend depends on the specific configuration of sites.

6 Field Experiments

We evaluated the proposed algorithm through proof-of-concept experiments using a custom UAV and a Clearpath Husky UGV (Figure 1). The UAV is capable of flying fully autonomous missions aided by GPS. The UAV is based on a DJI Flame Wheel F450 frame and uses the APM firmware running on a Pixhawk autopilot. The onboard computer (Nvidia Jetson TX1) interfaces with the autopilot using the mavros package [3] of the Robotic Operating System [37]. The Jetson TX1 runs the high-level mission algorithm and sends the list of waypoints to the autopilot.

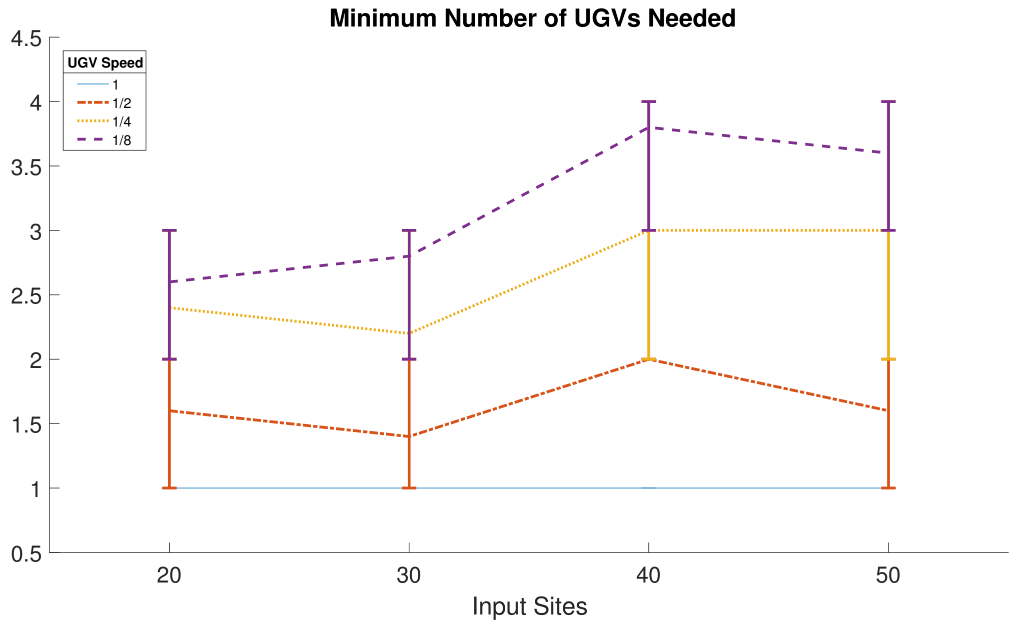


Figure 9: Minimum number of UGVs necessary to service an UAV with multiple UGV speeds. The UGV speed in this figure is in meters/second. The input was the same 5 random trails that were used for $m = 50$ data from Figure 6.

The mission to be executed by the UAV and UGV is computed offline. The mission is decomposed into subpaths; each subpath consists of a single take-off, visit one or more waypoints, and land sequence. While the UAV is executing the subpath, the UGV directly navigates to the landing site. Once at the final site on the subpath, the UAV lands on the UGV using a precision landing method that is implemented via IR-Lock (Figure 10a).

IR-Lock [2] is an off-the-shelf system integrated in APM firmware. It gives a landing accuracy of up to 30cm in ideal conditions. Our experiments suggest that the UAV is capable of landing directly on top of the Husky (59cm x 73cm) in normal conditions. In windy conditions, we mounted a larger platform of size 91cm x 122cm (Figure 10c) on top of the Husky. The UAV was repeatedly successful in landing on the paltform.

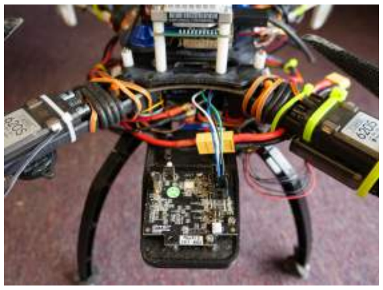
The IR-Lock sensor has 3.6mm lens giving horizontal and vertical viewing angles of approximately 60° and 35°. We use the MarkOne Beacon as our IR emitter, which has a detection range of 15m+ and a beam angle of 65°. During our experiments, the UAV flew at an altitude of 10m. Therefore, the UAV can detect the UGV within a 17m horizontal and 7m vertical footprint.

Once the UAV lands on the UGV, the UGV travel to the take-off site on the next subpath. We manually "cold-swapped" the battery of the UAV everytime it landed on the UGV. This part of the system can be completely automated using existing solutions such as those presented in [32, 5].

This next take-off site could be a new site due to TYPE II edge or the same site due to TYPE III edge. The UAV then detects if it is at the next take-off point by comparing its current GPS position with the GPS coordinates of the next take-off point. This process loops until the UAV reaches its final site whereupon the UAV would autonomously land on the ground. A video of the system in action can be seen in the multimedia submission.

Figure 11a shows the 50 sites used for the proof-of-concept experiments of the SMCS problem. The sites were randomly generated in an area of 300×110 meters. The path was found using the GLNS solver with $m = 100$ battery levels, $t_{TO} = 4$, $t_L = 4$, $r = 0$, and $t_{UGV} = t_{UAV}$. For this particular path, a single

(a) IR-Lock camera mounted on the UAV, viewed from the bottom.


(b) IR-Lock camera mounted on the UAV, viewed from the top.

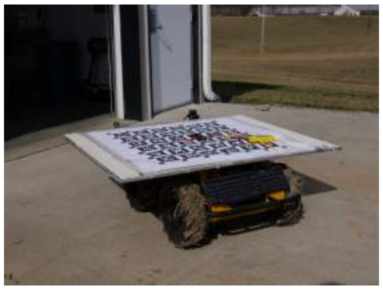
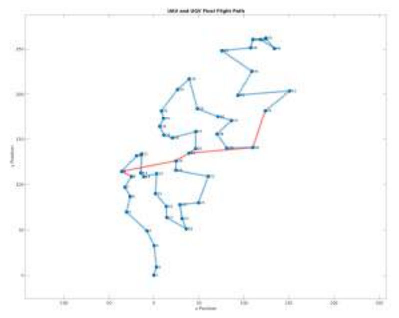

(c) IR-Beacon mounted on top of the landing platform on the UGV.

Figure 10: The IR-Lock system consists of a camera mounted on the UAV and an IR beacon mounted on the UGV. The system enables precision landing of the UAV on the UGV.

UGV was sufficient even though it is slower than the UAV. Regardless, our implementation will make the UAV hover over the landing site until the UGV reaches it. In the experiments, the UGV always reached the landing site before the UAV. The outputs for the UAV tour and UGV tour are shown in Figure 11a. Figure 11b shows the output data from the Pixhawk flight controller.

The total time for the experiment was approximately 23 minutes. A full version and a condensed version of a video showing the experiment is included in the multimedia submission. We conducted multiple sets of experiments with similar results (additional experiments can be found in [50]).


(a) Final MATLAB plot of UAV and UGV path.

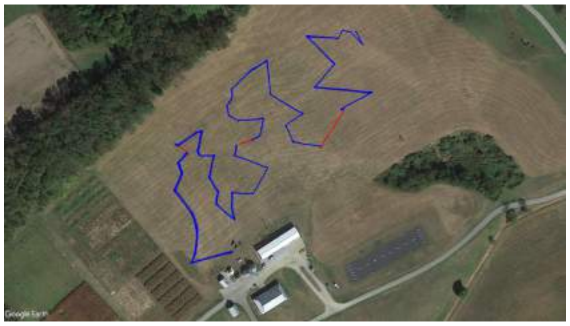

(b) GPS trace of the UAV.

Figure 11: Experiments were performed at Kentland Farms in Blacksburg VA. The input was 50 sites in a 300×110 meters area with 100 battery levels. $t_{TO} = 4$, $t_L = 4$, $r = 0$, and $t_{UGV} = t_{UAV}$. For the experiment the UAV autonomously took off, traversed sites, and landed. The blue subpaths are UAV-only, the red subpaths are UAV+UGV edges. The UGV operated autonomously between landing and take-off sites.

7 Conclusion and Future work

In this paper, we present an algorithm for routing a battery-limited UAV and a mobile recharging station to visit a set of sites of interest. We specifically consider scenarios where the UGV can recharge the UAV while being ferried to the next deployment location. We look at different combinations of input parameters that help dictate different methods to recharge in Section 5. We evaluated our assumptions through proof-of-concept experiments lasting about 30 minutes and rigorous simulations.

In this paper, we considered the case where the UAVs can be recharged only at a site or along the edge between two sites. Specifically, we assume that the UAV can take-off and land only at a site. This can be relaxed by adding additional sites that need not be visited by the UAV except for recharging. These sites will belong to a new cluster. Since GTSP requires each cluster to be visited at least once, we will also add a dummy vertex in this new cluster with zero cost edges to other clusters.

For the case where the UGVs are slower than the UAV, we decouple the problem by first finding a UAV path (assuming UGV is as fast a UAV) and then finding the minimum number of UGVs required to service the UAV path. This decoupled approach may not necessarily yield the minimum number of UGVs. It may be possible to directly optimize the UAV path and the number of UGVs required simultaneously. This is a possible direction for immediate improvement.

There are several avenues for future work. The algorithm can directly handle cases where the motion of the UGVs is restricted to specific regions, e.g., road networks. In this case, the recharging edges (Type II and Type III) will exist only for those sites that are accessible by the UGV. This can also be extended to multiple UAVs by using the techniques from [27, 48] which show how to use single-robot GTSP formulation for finding multi-robot tours. The longer-term future work is to design algorithms to handle multiple UAVs and UGVs as well as stochastic energy consumption models. Longer term future works would be to consider scenarios where the locations of all sites is not known *a priori* and requests for aerial inspection arrive over time. Finally, instead of inspecting sites, the objective may also include inspecting a set of regions (e.g., inspecting stressed regions in a farm [10]).

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 156624 and NIFA grant 2015-67021-23857.

References

[1] Coverage path planning under the energy constraint. In *Proceedings of the International Conference on Robotics and Automation (ICRA).*

[2] Ir-lock — infrared tracking systems for drones and robot automation. `https://irlock.com/`. (Accessed on 02/26/2018).

[3] mavros - ros wiki. `http://wiki.ros.org/mavros`. (Accessed on 02/26/2018).

[4] Shaimaa Ahmed, Amr Mohamed, Khaled Harras, Mohamed Kholief, and Saleh Mesbah. Energy efficient path planning techniques for uav-based systems with space discretization. In *Wireless Communications and Networking Conference (WCNC), 2016 IEEE*, pages 1–6. IEEE, 2016.

[5] Anonymous. Skysense charging pad. (Web), December 2016. http://www.skysense.co/charging-pad/.

[6] David Applegate, ROBERT Bixby, Vasek Chvatal, and William Cook. Concorde tsp solver. *URL http://www. tsp. gatech. edu/concorde*, 2006.

[7] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM (JACM)*, 45(5):753–782, 1998.

[8] Avrim Blum, Shuchi Chawla, David R Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. *SIAM Journal on Computing*, 37(2):653–670, 2007.

[9] Francesco Cocchioni, Adriano Mancini, and Sauro Longhi. Autonomous navigation, landing and recharge of a quadrotor using artificial vision. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 418–429. IEEE, 2014.

[10] Jnaneshwar Das, Gareth Cross, Chao Qu, Anurag Makineni, Pratap Tokekar, Yash Mulgaonkar, and Vijay Kumar. Devices, systems, and methods for automated monitoring enabling precision agriculture. In *Proceedings of IEEE Conference on Automation Science and Engineering*, pages 462–469. IEEE, 2015.

[11] Jason Derenick, Nathan Michael, and Vijay Kumar. Energy-aware coverage control with docking for robot teams. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3667–3672. IEEE, 2011.

[12] Carmelo Di Franco and Giorgio Buttazzo. Energy-aware coverage path planning of uavs. In *Autonomous Robot Systems and Competitions (ICARSC), 2015 IEEE International Conference on*, pages 111–117. IEEE, 2015.

[13] John A Dougherty. *Laser-Guided Autonomous Landing of a Quadrotor UAV on an Inclined Surface*. PhD thesis, The George Washington University, 2014.

[14] M. Dunbabin and L. Marques. Robots for environmental monitoring: Significant advancements and applications. *IEEE Robotics and Automation Magazine*, 19(1):24 –39, Mar 2012.

[15] Jeremy C Goldin. *Perching using a quadrotor with onboard sensing*. Utah State University, 2011.

[16] Bruno Herissé, Tarek Hamel, Robert Mahony, and François-Xavier Russotto. Landing a vtol unmanned aerial vehicle on a moving platform using optical flow. *IEEE Transactions on robotics*, 28(1):77–89, 2012.

[17] Saghar Hosseini, Ran Dai, and Mehran Mesbahi. Optimal path planning and power allocation for a long endurance solar-powered uav. In *2013 American Control Conference*, pages 2588–2593. IEEE, 2013.

[18] F Paulo Kemper, Koji AO Suzuki, and James R Morrison. Uav consumable replenishment: design concepts for automated service stations. *Journal of Intelligent & Robotic Systems*, 61(1):369–397, 2011.

[19] Jonghoe Kim, Byung Duk Song, and James R Morrison. On the scheduling of systems of uavs and fuel service stations for long-term mission fulfillment. *Journal of Intelligent & Robotic Systems*, pages 1–13, 2013.

[20] Weiwei Kong, Dianle Zhou, Daibing Zhang, and Jianwei Zhang. Vision-based autonomous landing system for unmanned aerial vehicle: A survey. In *Multisensor Fusion and Information Integration for Intelligent Systems (MFI), 2014 International Conference on*, pages 1–8. IEEE, 2014.

[21] Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research*, 59(3):345–358, 1992.

[22] Gilbert Laporte, Ardavan Asef-Vaziri, and Chelliah Sriskandarajah. Some applications of the generalized travelling salesman problem. *Journal of the Operational Research Society*, 47(12):1461–1467, 1996.

[23] Daewon Lee, Tyler Ryan, and H Jin Kim. Autonomous landing of a vtol uav on a moving platform using image-based visual servoing. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 971–976. IEEE, 2012.

[24] Lantao Liu and Nathan Michael. Energy-aware aerial vehicle deployment via bipartite graph matching. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 189–194. IEEE, 2014.

[25] Peter Liu, Albert Y Chen, Yin-Nan Huang, J Han, J Lai, S Kang, T Wu, M Wen, and M Tsai. A review of rotorcraft unmanned aerial vehicle (uav) developments and applications in civil engineering. *Smart Struct. Syst*, 13(6):1065–1094, 2014.

[26] Parikshit Maini and PB Sujit. On cooperation between a fuel constrained uav and a refueling ugv for large scale mapping applications. In *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, pages 1370–1377. IEEE, 2015.

[27] Neil Mathew, Stephen L Smith, and Steven L Waslander. Multirobot rendezvous planning for recharging in persistent tasks. *IEEE Transactions on Robotics*, 31(1):128–142, 2015.

[28] Nathan Michael, Ethan Stump, and Kartik Mohta. Persistent surveillance with a team of mavs. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2708–2714. IEEE, 2011.

[29] Derek Mitchell, Ellen A Cappo, and Nathan Michael. Persistent robot formation flight via online substitution. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4810–4815. IEEE, 2016.

[30] Fabio Morbidi, Roel Cano, and David Lara. Minimum-energy path generation for a quadrotor uav. In *IEEE International Conference on Robotics and Automation*, 2016.

[31] Scott Morton, Ruben D'Sa, and Nikolaos Papanikolopoulos. Solar powered uav: Design and experiments. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 2460–2466. IEEE, 2015.

[32] Yash Mulgaonkar and Vijay Kumar. Autonomous charging to enable long-endurance missions for small aerial robots. In *SPIE Defense+ Security*, pages 90831S–90831S. International Society for Optics and Photonics, 2014.

[33] Charles E Noon and James C Bean. An efficient transformation of the generalized traveling salesman problem. *INFOR*, 31(1):39, 1993.

[34] So-Ryeok Oh, Kaustubh Pathak, Sunil Kumar Agrawal, Hemanshu Roy Pota, and Matt Garrett. Autonomous helicopter landing on a moving platform using a tether. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3960–3965. IEEE, 2005.

[35] Tolga Ozaslan, Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. Inspection of penstocks and featureless tunnel-like environments using micro uavs. In *International Conference on Field and Service Robotics*, 2013.

[36] Fabio Pasqualetti, Antonio Franchi, and Francesco Bullo. On optimal cooperative patrolling. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 7153–7158. IEEE, 2010.

[37] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[38] Eduard Semsch, Michal Jakob, Dušan Pavlicek, and Michal Pechoucek. Autonomous uav surveillance in complex urban environments. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*, pages 82–85. IEEE Computer Society, 2009.

[39] Hazim Shakhatreh, Abdallah Khreishah, Jacob Chakareski, Haythem Bany Salameh, and Issa Khalil. On the continuous coverage problem for a swarm of uavs. In *Sarnoff Symposium, 2016 IEEE 37th*, pages 130–135. IEEE, 2016.

[40] Aydin Sipahioglu, Gokhan Kirlik, Osman Parlaktuna, and Ahmet Yazici. Energy constrained multi-robot sensor-based coverage path planning using capacitated arc routing approach. *Robotics and Autonomous Systems*, 58(5):529–538, 2010.

[41] Ryan N Smith, Mac Schwager, Stephen L Smith, Burton H Jones, Daniela Rus, and Gaurav S Sukhatme. Persistent ocean monitoring with underwater gliders: Adapting sampling resolution. *Journal of Field Robotics*, 28(5):714–741, 2011.

[42] S. L. Smith and F. Imeson. GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem. *Computers & Operations Research*, 87:1–19, 2017.

[43] Kaarthik Sundar, Saravanan Venkatachalam, and Sivakumar Rathinam. Formulations and algorithms for the multiple depot, fuel-constrained, multiple vehicle routing problem. In *American Control Conference (ACC), 2016*, pages 6489–6494. IEEE, 2016.

[44] Koji AO Suzuki, Paulo Kemper Filho, and James R Morrison. Automatic battery replacement system for uavs: Analysis and design. *Journal of Intelligent & Robotic Systems*, 65(1):563–586, 2012.

[45] Kurt A Swieringa, Clarence B Hanson, Johnhenri R Richardson, Jonathan D White, Zahid Hasan, Elizabeth Qian, and Anouck Girard. Autonomous battery swapping system for small-scale helicopters. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3335–3340. IEEE, 2010.

[46] Cornelius A Thiels, Johnathon M Aho, Scott P Zietlow, and Donald H Jenkins. Use of unmanned aerial vehicles for medical product transport. *Air medical journal*, 34(2):104–108, 2015.

[47] Pratap Tokekar, Ashish Kumar Budhiraja, and Vijay Kumar. Visibility-based persistent monitoring with robot teams. *IEEE Transactions on Robotics*, 2016. In Submission.

[48] Pratap Tokekar, Joshua Vander Hook, David Mulla, and Volkan Isler. Sensor planning for a symbiotic UAV and UGV system for precision agriculture. *IEEE Transactions on Robotics*, 2016. To Appear.

[49] Tuna Toksoz, Joshua Redding, Matthew Michini, Bernard Michini, Jonathan P How, Matthew Vavrina, and John Vian. Automated battery swap and recharge to enable persistent uav missions. In *AIAA Infotech@ Aerospace Conference*, 2011.

[50] Kevin Yu, Ashish Kumar Budhiraja, and Pratap Tokekar. Algorithms and experiments on routing of unmanned aerial vehicles with mobile recharging stations. IEEE International Conference on Robotics and Automation, 2018. To appear.