Energy-Aware Drone Path Finding with a Fixed-Trajectory Ground Vehicle

JONATHAN DILLER and QI HAN, Colorado School of Mines, USA

Rotary-wing Unmanned Aerial Vehicles (commonly referred to as drones) are versatile autonomous transportation platforms that can be used for a variety of data collection applications including emergency response, environmental monitoring, surveillance, and many others. In this work, we investigate how to plan efficient paths that minimize mission completion time for drone data collection where the drone must rendezvous with a moving ground vehicle (GV) that cannot stop and wait for the drone. Moreover, we address the limited onboard energy storage issue by adapting drone speed. We propose a mixed-integer nonlinear program (MINLP) solution to solve this problem to optimality and provide two heuristics-based alternative solutions (the *k*-TSP and *D*-TSP approaches) that are more computationally tractable. We evaluate these approaches in extensive simulations using real drone characteristics to highlight their trade-offs. Our results show that the *k*-TSP algorithm performs well when data collection points are closer to the GV, averaging within 4.5% of the optimal solution, while the *D*-TSP approach is more versatile, finding solutions in situations where the *k*-TSP algorithm tends to fail. Furthermore, we show that adapting drone speed can improve solution quality by up to 47.1% compared to fixed-speed approaches. In summary, this article serves as an exploratory study in energy-aware planning and scheduling for drones and other autonomous transportation systems.

CCS Concepts: • **Computing methodologies → Planning for deterministic actions**.

Additional Key Words and Phrases: Drone Path Finding; Cooperative Vehicle Routing; Mixed-Integer Nonlinear Program; Adaptive Drone Speed

ACM Reference Format:
Jonathan Diller and Qi Han. 2024. Energy-Aware Drone Path Finding with a Fixed-Trajectory Ground Vehicle. 1, 1 (February 2024), 33 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Rotary-wing Unmanned Aerial Vehicles (i.e., drones) can easily move over rough terrain and water, are low-cost, commercially available, and can be deployed quickly. This makes them well-suited for various data collection applications such as emergency response, environmental monitoring, agriculture, and surveillance. Studying how to deploy drones will better enable them as autonomous systems and facilitate their adoption into society.

A common requirement in these applications is minimizing mission completion time. Intuitively, this implies that we want to find an optimal path for a drone to follow while maximizing the drone's speed. However, drones have limited onboard energy storage and there is a trade-off between speed and energy consumption [54, 66], where maximizing speed does not maximize travel distance due to a higher energy consumption rate (which is further discussed in Section 4). We see a lack of existing research on how to integrate this tradeoff into planning algorithms. Any robust path planning approach should consider the speed-energy consumption trade-off and plan for recharging or battery swaps.

Authors' Contact Information: Jonathan Diller, jdiller@mines.edu; Qi Han, qhan@mines.edu, Colorado School of Mines, Golden, Colorado, USA.

Manuscript submitted to ACM

In many applications, drones often work with GVs which provide battery recharging, battery swap, data offloading, etc. Collaborative drone-GV problems have already been well studied [10, 60, 64]. However, these existing works assume that the GV can stop and wait for the drone to return. While a drone completes its given tasks, the GV deploying the drone may need to continue moving rather than wait at the starting location for the drone to return. One example of this is a large ship on a maritime search and rescue mission, such as those recently seen in the Mediterranean sea where migrants frequently become stranded on failing vessels [30]. The drone can be deployed to inspect minor areas of interest while the main search and rescue ship continues along a preplanned trajectory. Other examples include drones used in urban environments for package delivery or to assist first responders while relying on volunteer vehicles for ride hitching [48]. In such applications, drone path planning algorithms must account for the movement of a GV that is following a preplanned trajectory.

In this work, we seek to answer the research question: *How do we plan energy-aware drone paths and rendezvouses with a non-stopping GV for general data collection applications?* To this end, we consider the trade-offs between drone speed and total travel distance and propose various speed-adapting algorithms. This article expands on our previously published conference paper [16]. For completeness and context, major parts of our previous publication are included in this manuscript. We expand on the previous publication in the following ways.

(1) We remove the assumptions made in our previous work that the drone will be launched at mission start time ($t = 0$) and that subsequent launches happen immediately after swapping batteries on the drone (found in Section 7). Selecting a launch time is now part of the considered problem.
(2) We provide a complete solution to the problem as a whole (in Section 7.1) and prove that the problem is NP-Hard (in Section 3). Our previous publication only provided a complete solution for an underlying Hamiltonian Paths Problem found when fixing the start and stop time of each drone sub-tour and commented on the hardness of the underlying problem.
(3) We expand our literature review to include more recent related work on this topic (in Section 2).
(4) We expand on the simulation evaluations. In addition to more randomly generated inputs (presented in Fig. 6b, 7, and 11b), we also evaluate our solution on non-linear GV trajectories (presented in Fig. 10).

In the next section, we discuss related works and how they compare to our body of research. In Section 3 we formally define the considered problem, termed Minimum-Time while On-the-Move (MT-OTM), then discuss our drone energy model in Section 4. In Section 5 we present our solution framework for the MT-OTM Problem and then further discuss details of our various proposed algorithms in Sections 6 and 7. In Section 8 we provide simulation results and discuss the details of our simulation setup while Section 9 details our field test prototype of the MT-OTM Problem using a physical drone. Finally, we discuss the conclusions of our study and provide closing remarks in Section 10.

2 Related Work

In this section we review recent works in literature with a focus on drone path planning algorithms for mixed drone and GV problems. We also review drone energy models and how they have been applied in drone path finding. For clarity, we refer to unmanned variations of GVs as Unmanned Ground Vehicles (UGVs). The term “GV” may include both manned and unmanned vehicles.

2.1 Multi-Drone Path Finding

Outside of coordinated drone-GV teams, there has been extensive research on path finding for drone swarms with common proposed applications in data collection, package delivery, and supporting wireless infrastructure [1, 2, 9, 57]. These path finding problems are usually presented as variations of the Vehicle Routing Problem (VRP) or the Multi-Traveling Salesman Problem (MTSP) [25, 31, 62]. In the MTSP, we are given a set of salesmen that are deployed from the same depot and a set of stops that must be visited by at least one salesman and are ask to find efficient paths for the salesmen such that all stops are visited [8]. The VRP extends the MTSP by allowing vehicles to start from multiple depots, with variations including capacity constraints, time windows, or pick-ups and deliveries, just to name a few [11].

Common solution approaches include both exact methods and heuristics-based algorithms. Math optimization approaches (such as mixed integer programming) are largely popular for exact methods [25, 31, 47, 62]. Heuristics-based solutions commonly simplify the problem into smaller graph search problems [52, 53] or simplify the problem into the classical Traveling Salesman Problem (TSP) and utilize well-studied heuristics-based solutions, such as the Lin-Kernighan-Helsgaun heuristic [18, 50], genetic algorithms [58], or ant colony optimization [40].

2.2 Mixed Drone & Ground Vehicle Path Finding

Many previous works look at cooperative problems involving drones and GVs where a series of waypoints must be visited. In [60] they consider an application where either a drone or a UGV must be within some distance of a set of waypoints, which is modeled as an orienteering problem, a problem with known algorithms. In [32] they consider drone path planning where a GV can be used to swap-out batteries on the drone but is constrained to a network of streets. They propose treating all drone waypoints as a single TSP, then breaking up the route into sub-tours. This work was further expanded in [3, 33] where they propose a mixed-integer linear programming (MILP) solution. A related scenario is considered in [64, 65], where a UGV can ferry around the drone. The authors model the problem as a Generalized TSP and apply a known solution approach [39]. A similar problem is found in [12], where they first plan GV routes along a road network then plan drone routes using Conflict-Based Search. A similar problem was looked at in [43] and extended in [44], where multiple drones are deployed with a single UGV. The drones must visit mission waypoints while the UGV provides recharging for the drone but is not restricted to a road network. A further extension is found in [17], where the authors use convex optimization to select rendezvous points. They solve this problem by first using a clustering algorithm and a TSP solver to find a path for the UGV then plan drone paths with a set of fixed recharging locations. An iterative solution to this problem setup is found in [59] while [13] looks at how to represent the problem as an agent-based model with behavior trees and finite state machines. A common theme in all of these works is that the GV can stop and wait for the drone to finish flying a sub-tour of waypoints, which often allows the problem to be modeled as a traditional graph theory problem and solved using known techniques. In contrast, our work considers how to plan drone paths when the GV cannot stop and wait for the drone. This means that we must consider the trajectory of the GV as part of the problem.

A related set of drone and GV problems is the drone parcel delivery problem, where a delivery truck, bound to a network of streets, acts as a launching point for a drone to deliver a package in a last-mile delivery system [36]. These scenarios can largely be broken into two catagories: (1) where the drones can only make a single delivery stop per sortie, as seen in [38, 42, 49, 61], and (2) where the drones are allowed to deliver multiple packages per sortie [24, 31, 34, 41]. These problems are usually solved by breaking down the larger problem into smaller ones that can be handled using math programming techniques [10]. Most of the literature on this set of problems allows for one of the vehicles to stop

and wait for the other. However, in applications over water or rugged terrain, the drone cannot land to wait for the GV and hovering is energy intensive.

Another set of related problems looks at selecting rendezvous locations for a drone to be recharged by a GV. In [35], the authors plan where a GV should meet up with a drone on a fixed route by converting the problem into the Generalized TSP and solve it using both integer programming and the Lin-Kernighan-Helsgaun (LKH) heuristic [20]. In [55] both the drone and UGV have established paths and must select rendezvous points that are ideal for both vehicles, which is done using a Markov Decision Process. The work in [55] was further extended in [4] to include multiple UAVs and multiple UGVs. Rather than looking at the trade-off between varying drone speeds, these works looked at the risk of running out of battery and used this metric to determine how often the drone should recharge with the UGV. Their problem setup assumes that the paths for both the UGV and UAV are already determined and is similar to the baseline approach in our work. Although our work also selects rendezvous locations, our scenario differs from the current literature because we consider planning rendezvouses with a non-stopping GV and fit this problem into a larger, combined rendezvous and path planning problem.

For this work, we assume that the drone is capable of landing on the GV after rendezvousing using techniques such as the ones discussed in [5, 68]. Landing on a moving vehicle falls out of the scope of this research.

2.3 Drone Energy Models

Drones have limited onboard energy, motivating the need to model and integrate drone energy consumption into planning algorithms. There have been many proposed methods for modeling energy consumption in drones including: the distance-based method [27, 32], the time-based method [26, 52], discretized approaches [14, 60, 64], and the velocity-based method [28]. The distance-based model assigns a max travel distance and ignores or excludes time spend hovering. The time-based method assigns a total operating time to the drone and usually does not differentiate between time spent hovering versus time spent traveling. Discretized approaches allot an energy budget to the drone and reduce the budget based on actions, such as flying a certain distance or hovering for some period of time. However, the authors in [38] found that these energy modeling approaches are not as accurate as the velocity-based approach. The velocity-based model maps the drone's speed to power consumption based on characteristics specific to the drone. This model was formulated separately in [66] and [28] and validated in [28, 54] through field testing on physical testbeds. Due to its accuracy, we use the velocity-based model in this work.

The velocity-based energy model has been used in various drone path planning problems. In [38], it was applied to the drone parcel delivery problem and in [55] it was used to better model the risk of the drone running out of energy in a rendezvousing problem. In [42], they used this energy model to adapt drone speed as a post-processing step to further improve mission completion time. However, mission completion time is the product of both the drone's speed and the distance that the drone must travel. To directly optimize completion time the energy model should be embedded in the solution formulation. In our work, we add speed adaptation using the velocity-based energy model directly in our mathematical formulation and evaluate speed adaptation against fixed-speed approaches for a multi-waypoint drone problem.

3 Problem Formulation

In this section, we describe the system setup and formally define the MT-OTM problem. In general, in the MT-OTM Problem, we are given a drone that must visit a series of waypoints while being launched from and received by a moving GV. The drone has limited onboard energy and may need to stop several times on the GV to swap out its batteries or

Table 1. Symbols used in the Paper

Symbol	*Description*
Δ_m	Set of m sub-tours
δ_k	Sub-tour k, an ordered set of waypoints
d_k	Distance of sub-tour k
Γ_m	Set of m start-end time tuples
P	Set of waypoints to visit
p_k^d	Depot waypoint for sub-tour k
p_k^t	Terminal waypoint for sub-tour k
Φ_m	Complete drone tour with m sub-tours
S_m	Set of m speeds for sub-tour m
t_b	Time required to swap batteries between sub-tours
τ_k^d	Start time for sub-tour k
τ_k^t	End time for sub-tour k
$\mathcal{X}$	Application space (assumed to be in $\mathcal{R}^2$)

recharge. Our objective is to minimize the time it takes for the drone to visit every waypoint and return to the GV. We focus on the offline, mission planning problem of identifying efficient drone paths for visiting every waypoint in minimal time. Online control for the drone as well as communication and coordination with the GV, although ongoing areas of research, are assumed to function fully as intended and fall out of scope of this work.

Table 1 includes a list of symbols used in our problem formulation. Our problem formulation considers drone energy and battery storage, which is further discussed in detail in Section 4. We acknowledge but leave out various factors in our problem formulation that could affect flight performance such as wind and the energy consumed by making turns. Although we do not consider these and other minor factors that can impact energy consumption, our proposed solution is versatile and can be reapplied with a more comprehensive energy model.

Let $\mathcal{X} \in \mathcal{R}^2$ be a large area with several navigational waypoints that must be visited. A GV that acts as a moving base station moves through $\mathcal{X}$ on a predetermined, fixed route described by $p_b(t)$, a function that returns the GV's position at time t. That is, $p_b(t) \in \mathcal{X}$ for any $t \geq 0$. Without loss of generality, we assume that the origin of the two-dimensional coordinate system for $\mathcal{X}$ is at the GV's initial position at the beginning of the considered time window. Figure 1 shows the general problem setup.

Let P be the set of all waypoints that must be visited by a drone and p_i be the i^{th} waypoint in P. Due to energy constraints, the drone may not be able to visit all of the waypoints in P in a single tour. Let m be the number of sub-tours required to visit every waypoint in P, which is initially unknown. We define the k^{th} sub-tour, denoted as δ_k with path length d_k, as an ordered set of waypoints containing a starting position in $\mathcal{X}$ (a *depot*), an ending position in $\mathcal{X}$ (a *terminal*), and at least one waypoint in P. We denote the depot of sub-tour δ_k as p_k^d and the terminal as p_k^t. Let Δ_m be a set of m sub-tours such that every waypoint in P is visited exactly once and let the set of speeds for each sub-tour in Δ_m be S_m. The time it takes a drone to travel δ_k while moving at speed $s_k \in S_m$ will be

$$t_\delta(\delta_k, s_k) = \frac{d_k}{s_k}. \tag{1}$$

We denote the start time of sub-tour k as τ_k^d and the end time of sub-tour k as τ_k^t. By definition, for any sub-tour k, $\tau_k^t = \tau_k^d + t_\delta(\delta_k, s_k)$. Suppose that it takes t_b seconds to land the drone on the GV and change out the battery between

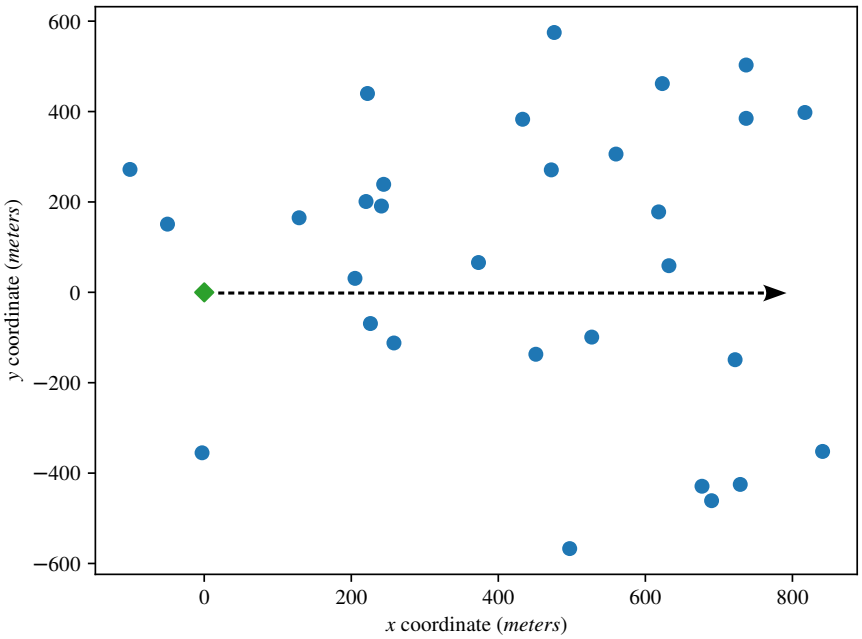


Fig. 1. General problem setup: the blue circles are navigational waypoints that must be visited by the drone, the green diamond is the GV's starting location, and the black, dashed arrow shows the GV's fixed trajectory.

each sub-tour. The earliest that sub-tour $k+1$ can start is $\tau_k^t + t_b$. That is, $\tau_{k+1}^d \geq \tau_k^t + t_b$. We assume that the drone cannot be sent out earlier than the considered time horizon (i.e. $\tau_1^d \geq 0$). Let Γ_m be a set of m $\{\tau_k^d,\ \tau_k^t\}$ tuples, for $k \in \{1 \cdots m\}$, where each $\{\tau_k^d,\ \tau_k^t\}$ tuple is the start and end time for sub-tour k.

We define a complete drone tour as $\Phi_m = \{\Delta_m, S_m, \Gamma_m\}$. We say that Φ_m is *consistent* if the following constraints hold:

$$\tau_1^d \geq 0 \tag{2}$$

$$\tau_{k+1}^d \geq \tau_k^t + t_b, \forall k \in \{1 \cdots m-1\} \tag{3}$$

$$\tau_k^t = \tau_k^d + t_\delta(\delta_k, s_k), \forall k \in \{1 \cdots m\} \tag{4}$$

$$p_k^d = p_b(\tau_k^d), \forall k \in \{1 \cdots m\} \tag{5}$$

$$p_k^t = p_b(\tau_k^t), \forall k \in \{1 \cdots m\} \tag{6}$$

Constraint 2 states that the first sub-tour cannot start earlier than the considered time horizon while Constraint 3 requires any subsequent sub-tour to not start earlier than the end time of the previous sub-tour plus the time required to swap out batteries on the drone. Constraint 4 requires that the end time of any sub-tour is equal to the start time of that sub-tour plus the time required to travel the distance of that sub-tour at the designated speed. Constraint 5 requires that the position of the depot waypoint for sub-tour k be based on the start time of that sub-tour and the GV trajectory function $p_b(t)$ while Constraint 6 holds the same requirements for the terminal waypoint of the sub-tour.

We formally define the MT-OTM problem as:

Definition 1 (MT-OTM Problem). *Given application-space $\mathcal{X}$ with waypoint set P and vehicle position function $p_b(t)$, determine the number of sub-tours m and a corresponding consistent Φ_m such that τ_m^t is minimized and all waypoints in P are visited at least once.*

Solving the MT-OTM Problem is challenging. To show the problem is at least as hard as any problem in the NP-Hard problem space, we first introduce a slightly simplified version of the problem, the Minimum-Time while On-the-Move with a fixed number of sub-tours (fixed-Mt-OTM) problem, defined as:

Definition 2 (fixed-MT-OTM Problem). *Given application-space $\mathcal{X}$ with waypoint set P, vehicle position function $p_b(t)$ and a number of sub-tours m, determine a corresponding consistent Φ_m such that τ_m^t is minimized and all waypoints in P are visited at least once.*

The fixed-MT-OTM Problem can easily be shown to be NP-Hard by a reduction from the TSP, a well known NP-Hard problem [22]. In the classic TSP, we are given n vertices in a fully connected weighted graph and are asked to find a minimum weight cycle in the graph that visits every vertex. Our strategy for reducing TSP to the fixed-MT-OTM Problem is to make the GV stationary.

Theorem 1. *The fixed-MT-OTM problem is NP-Hard.*

Proof. Select an arbitrary vertex in the TSP and set this as the starting location for the GV in the MT-OTM Problem and setting $p_b(t)$ to be this location at all input times. Map the remaining vertices in the TSP to waypoint set P by treating the edge weights as distances between waypoints and setting the drone's energy profile such that it can visit every vertex from the TSP without requiring a battery swap. An optimal solution to this fixed-MT-OTM Problem will also be an optimal solution to the original TSP. This reduction can be performed in linear time in terms of the number of vertices in the TSP.

As we have found a polynomial time reduction from the TSP to the fixed-MT-OTM Problem and it has been well established that TSP is NP-Hard, then the fixed-MT-OTM Problem is at least as hard as any NP-Hard problem. □

The last part of the reduction can be done by assuming the drone has an arbitrarily large and weightless battery. Drone energy models are further discussed in the following section.

We next show that the original MT-OTM Problem is NP-Hard by repeatedly reducing the fixed-MT-OTM Problem to the MT-OTM Problem.

Theorem 2. *The MT-OTM problem is NP-Hard.*

Proof. With $m = 1$, solve the MT-OTM Problem input as a fixed-MT-OTM Problem. Increment the value of m, solve the fixed-MT-OTM Problem with this new value of m, then repeat the process until $m = n$. The value of m and corresponding consistent Φ_m with a minimal value for τ_m^t that was found when solving the fixed-MT-OTM Problem will be the optimal solution to the MT-OTM Problem. This reduction can be performed in polynomial time in terms of the number of waypoints in P.

As we have found a polynomial time reduction from the fixed-MT-OTM Problem to the MT-OTM Problem and we showed that the fixed-MT-OTM Problem is NP-Hard, then the MT-OTM Problem must also be at least as hard as any NP-Hard problem. □

As shown in our hardness proof, solving the MT-OTM Problem could be simplified to finding a single drone sub-tour (i.e. finding an optimal Φ_1) if the drone has enough on-board energy to complete Φ_1 without needing to stop to swap out batteries. However, this will not be the case in more interesting application scenarios and the drone will have to perform multiple sub-tours. This requires us to consider how drones consume energy in our solution.

4 Adaption of Drone Speed

To find a realistic solution to the MT-OTM Problem for non-arbitrary inputs, we need an accurate model for how drones consume energy. In this section, we summarize previous works that looked at how drones consume energy and we discuss the trade-off between maximizing a drone's speed versus maximizing a drone's travel distance.

Previous work has determined the amount of power consumed by a drone at varying speeds [28, 66]. From [66], we see that propulsion power consumption of a rotary-wing drone as a function of speed v can be approximated as:

$$\mathcal{P}(v) \approx C_0 \left(1 + \frac{3v^2}{U_{tip}^2}\right) + \frac{C_i v_0}{v} + \frac{1}{2} d_o \rho s_r A v^3, \tag{7}$$

where C_0 and C_i are constants representing blade profile power and induced power, respectively, U_{tip} represents the tip speed of the drone's propellers, v_0 is what is known as the mean rotor induced velocity while hovering, d_0 is an aircraft-specific drag ratio, s_r is rotor solidity, ρ is the air density and A is the rotor disk area.

Equation 7 is highly dependent on specific aircraft parameters and will change from aircraft to aircraft, but in general, this function has the shape of an upwards-facing parabola as shown in Fig. 2a. The exact values for each parameter in Eq. (7) can be experimentally determined for individual drones, as demonstrated in [54]. Drone payloads will impact the values of these parameters and we assume that these values are known *a priori* for the drone and its payload. For a more detailed summary on drone energy models over varying drone types, we refer the reader to more comprehensive studies on this topic in [66, 67].

Many drone applications depend more on the total distance that a drone can travel as opposed to just the amount of energy consumed. The relationship between speed and the total distance depends on the voltage that a battery supplies

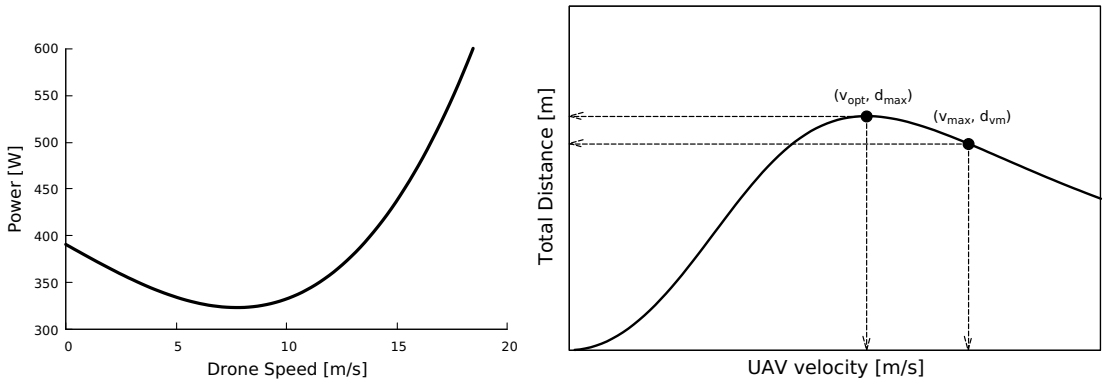


Fig. 2. Graph 2a shows the relationship between a drone's speed and power consumption (Eq. 7) using experimental data in Ref. [54]. Graph 2b shows the relationship between a drone's speed and its max travel distance (Eq. 9).

to the drone and the rate that the battery discharges. We can represent this relationship as:

$$\mathcal{D}(v) = \frac{B_{rate} V_{bat} v}{\mathcal{P}(v)}, \tag{8}$$

where B_{rate} is the rate of battery discharge in amp-seconds and V_{bat} is the voltage of the battery. In practice, V_{bat} should be set lower than what the battery is rated for to ensure a safety buffer for uncertainty.

Equation (8) has the following general form (Fig. 2b):

$$\mathcal{D}(v) = \frac{v}{v^3 + v^2 + \frac{1}{v}}. \tag{9}$$

Intuitively, to minimize total mission time, we will want to maximize the drone's speed. It was found [66] and then verified experimentally [54] that in order to achieve maximum traveling distance, the drone must travel at a lower speed than its maximum possible speed. Let d_{max} be this maximum achievable distance and v_{opt} be the speed that achieves d_{max}. Let v_{max} be the maximum speed that the drone is capable of traveling and d_{vm} be the distance that the drone can travel when moving at v_{max}. As shown in Figure 2b, if $v_{opt} < v_{max}$ then $d_{max} > d_{vm}$.

Inspired by this finding, we formulate a function $v(d)$ that takes a distance and gives us the maximum speed that a drone can travel to achieve this distance.

$$v(d) = \begin{cases} v_{max} & \text{if } d \leq d_{vm} \\ \mathcal{D}^{-1}(d) & \text{if } d_{vm} < d \leq d_{max} \\ infeasible & \text{if } d > d_{max} \end{cases} \tag{10}$$

where $\mathcal{D}^{-1}(d)$ is the inverse of Eq. (8). Note that in the third case (i.e., if $d > d_{max}$) the drone is not capable of actually traveling distance d. This is the adaptive speed we use in our approaches.

If given realistic parameters for a drone, we could determine $\mathcal{D}(v)$. However, $\mathcal{D}^{-1}(d)$ is not easy to work with. Therefore, we instead propose approximating Eq. (8) between d_{vm} and d_{max} as a second order polynomial then finding the inverse of this polynomial to approximate Eq. (10). The inverse of such a polynomial will have the form

$$v(d) = \frac{\sqrt{c_1 + c_2 d}}{c_3} + c_4 \tag{11}$$

where c_1, c_2, c_3 and c_4 will be constants.

In the MT-OTM Problem, we want the drone to visit every waypoint and return to the GV in minimal time. To minimize time, we would like to minimize the distance that the drone must travel while maximizing the speed that the drone travels at. However, if moving faster means that the drone runs out of battery before visiting every waypoint then we may waste time swapping batteries. This creates a tradeoff between maximizing the speed of the drone and maximizing the distance that the drone can travel - potentially avoiding the need to swap out batteries as often. Drone path finding algorithms must be cognizant of this tradeoff and seek to exploit it when battery swaps or charging is required to complete the considered task. In the following section we show how to minimize the distance that the drone must travel and discuss our strategy for creating an energy-aware path planner that exploits the tradeoff between maximizing travel distance and maximizing speed.

5 Our Solution Framework

In this section, we summarize our framework for finding a solution to the MT-OTM problem. Our framework follows the same strategy that we used in Theorem 2 where we fixed the number of sub-tours (m) then solve the fixed-MT-OTM Problem.

Algorithm 1 depicts the process described in Theorem 2 to determine m^o, the optimal number of sub-tours, and a corresponding Φ_m^o, an optimal drone path, for the MT-OTM problem. We first set $m = 1$ (lines 1 and 3), solve for Φ_m using the function *path-planning()* (line 4), then increment m until $m = |P|$ (lines 3 and 8, respectively). On each iteration we check to see if we found a better solution (line 5) and update m^o and Φ_m^o (line 6), as needed. Note that the *path-planning()* function may fail to find a valid solution, particularly when there are a large number of waypoints to visit and m is small. In these scenarios, the condition on line 5 is false and the algorithm continues.

The runtime of Algorithm 1 greatly depends on the runtime of the *path-planning()* function. Assume that the *path-planning()* function runs in $O(\psi)$. The loop that starts at line 2 will run $|P| = n$ times, making the runtime of Algorithm 1 $O(n\psi)$. In practice, we can reduce the number of times we call the *path-planning()* function by terminating the algorithm when we stop seeing an improvement after incrementing m. However, this modification will not improve the worst-case run-time and provides no guarantee of optimality.

If we can write a function *path-planning()* that finds an optimal solution to the fixed-MT-OTM Problem, then Algorithm 1 will find an optimal solution and is an exact algorithm for the MT-OTM problem. In Section 6, we simplify the fixed-MT-OTM Problem by assuming that the first sub-tour starts at the beginning of the considered time horizon and that every consecutive sub-tour begins as early as possible. This assumption allows us to design both a MINLP and a heuristics-based solution for the *path-planning()* function. By building on the algorithms presented in Section 6, we remove our simplifying assumption on sub-tour start times in Section 7 and formulate an exact solution to the fixed-MT-OTM Problem and propose a flexible heuristics-based solution.

6 Fixed-Start MT-OTM Solutions

To solve the MT-OTM problem, Algorithm 1 breaks the problem into repeated instances of the fixed-MT-OTM problem which are solved using a suitable implementation of the function *path-planning()*. In this section we propose a method for implementing *path-planning()* that assumes the first drone sub-tour begins at the beginning of the considered time horizon and that all consecutive sub-tours will start as early as possible. That is, we assume $\tau_1^d = 0$ and $\tau_{k+1}^d = \tau_k^t + t_b$ for $m \geq k \geq 1$, where τ_k^d and τ_k^t are the start and end times of the k^{th} sub-tour, respectively. We term this the *fixed-start*

Algorithm 1 *MT-OTM Solver*

Input: P : set of waypoints to visit, $p_b(t)$: GV position function
Output: m^o : optimal number of sub-tours, Φ_m^o : optimal complete drone tour

1: $m^o \leftarrow 0,\ \Phi_m^o \leftarrow \emptyset,\ m \leftarrow 0$
2: **Do:**
3: $\quad m \leftarrow m + 1$
4: $\quad \Phi_m \leftarrow\ path\text{-}planning(P, p_b(t), m)$
5: $\quad$ **if** $\Phi_m.\tau_m^t < \Phi_m^o.\tau_m^t$ **then**
6: $\quad\quad \Phi_m^o \leftarrow \Phi_m, m^o \leftarrow m$
7: $\quad$ **end if**
8: **While** $m < |P|$
9: **return** m^o, Φ_m^o

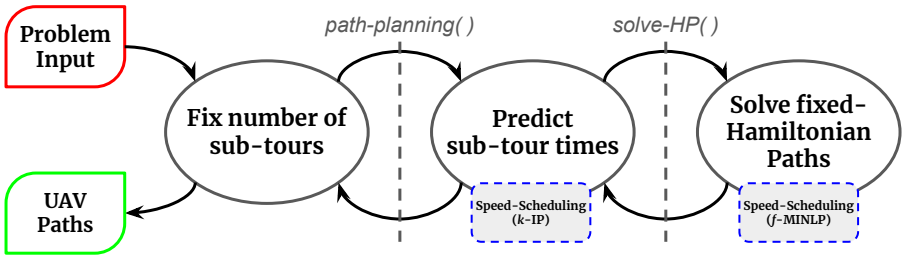


Fig. 3. A flow graph for solving the MT-OTM Problem using the fixed-start assumption. We initially fix the number of sub-tours, then iteratively predict sub-tour times and find fixed-Hamiltonian paths until the solution is consistent in the *path-planning()* function. We then increase the number of sub-tours and rerun the *path-planning()* function.

assumption. The motivation for using the fixed-start assumption is that it simplifies the *path-planning()* function and can outperform generalized approaches that remove the assumption (as shown in Section 8).

Figure 3 shows the flow of our proposed *path-planning* function based on the fixed-start assumption while Algorithm 2 provides more details on our implementation. We start by guessing at the total time, t, that it will take to visit the waypoints in P based on $p_b(t)$ and m. Vector A represents the time required for each of the m sub-tours. In the while loop, we set our guess for the total time and sub-tour times to t' and A', respectively. We then form the graph G'_m using P, $p_b(t)$, m, and A in function *form-graph*(), which determines the locations of all p^d_k and p^t_k (the depots and terminals of each sub-tour) using A. This graph can be described mathematically as $G'_m = (V_m, E_m)$, where $V_m = P \cup \{p^d_1, \cdots, p^d_m\} \cup \{p^t_1, \cdots, p^t_m\}$ and

$$\begin{aligned} E_m = &\{(p_i, p_j) \mid 1 \le i, j \le n \text{ and } i \ne j\} \\ &\cup \{(p^d_k, p_i) \mid 1 \le k \le m \text{ and } 1 \le i \le n\} \\ &\cup \{(p_i, p^t_k) \mid 1 \le i \le n \text{ and } 1 \le k \le m\}. \end{aligned}$$

Once we find G'_m, the problem becomes solving an underlying Hamiltonian Path problem (function *solve-HP()*). We set $\mathbf{\Delta}$ and S as the found set of sub-tours and set of assigned speeds, respectively, then update the value of t based on the actual total mission time to run all sub-tours in $\mathbf{\Delta}$ given S. We update A using the function *sub-tour-times*(), which updates each sub-tour k's entry in A based on the time required to travel sub-tour δ_k while moving at speed s_k.

The loop continues until one of the following conditions is met:

(1) The predicted mission time, t, is within some epsilon of the actual time, t', and every entry of our predicted vector A' is within some epsilon of the actual vector A, or
(2) Some iteration limit has been met.

We then return the last found graph G' and the corresponding set of sub-tours $\mathbf{\Delta}$. If an iteration time-out condition occurs, we propose updating G' based on t and A without changing the found set of tours $\mathbf{\Delta}$.

To make an initial guess for t, we find the minimum spanning forest of m trees on $P \cup \{p^d_1\}$, determine the minimum time that a UAV needs to fly the total distance of the forest while making $m-1$ stops to swap batteries. We can use this time to predict where the base station will be located for each of the $m-1$ stops. Using this intermediate guess, we can create a G'_m, find a new minimum forest of m trees in G'_m then use the distance of this new forest to get a guess on the time required to complete the search.

Algorithm 2 *path-planning*

Input: P : set of waypoints to visit, $p_b(t)$: GV position function, m : number of sub-tours
Output: Φ_m : complete drone tour with m sub-tours

$t \leftarrow$ *guess-time*$(P, p_b(t), m)$
$A \leftarrow \{\frac{t}{m}, \cdots, \frac{t}{m}\}$
Do:
  $t' \leftarrow t, A' \leftarrow A$
  $G'_m \leftarrow$ *form-graph*$(P, p_b(t), m, A')$
  $\Delta, S \leftarrow$ *solve-HP*(G'_m)
  $t \leftarrow t(\Delta, S)$
  $A \leftarrow$ *sub-tour-times*(Δ, S)
While $\left(\left|t - t'\right| \geq \epsilon_t$ **or** $\left|max_i(A - A')\right| \geq \epsilon_A\right)$ **and** iterations < iteration-limit
return Φ_m

Our fixed-start solution framework reduces the MT-OTM problem into an underlying fixed multi-depot, multi-terminal Hamiltonian paths problem (fixed-MdMtHPP). This is a special case of the more general MdMtHPP, which is formally defined as: "Given m salesmen that start from distinct depots, m terminals and n destinations, the problem is to choose paths for each of the salesmen so that (1) each salesman starts at his respective depot, visits at least one destination and reaches any one of the terminals not visited by other salesmen, (2) each destination is visited exactly once, and (3) the cost of the paths is minimum among all possible paths for the salesmen" [6]. fixed-MdMtHPP differs from the traditional MdMtHPP in that the depots and terminals are fixed. Each tour that starts at some depot (p_k^d) must end at a specific terminal (the corresponding p_k^t).

MdMtHPP is NP-Hard ([6]). Fixing the matching between depots and terminals does not make the problem easier to solve.

Theorem 3. *The fixed multi-depot, multi-terminal Hamiltonian path problem (fixed-MdMtHPP) is NP-Hard.*

Before beginning the proof for Theorem 3, we would like to remind the reader of the fixed destination multi-salesmen, multi-depot TSP (MmTSP). In MmTSP, there is a given set of depots with one or more salesmen and a set of destination vertices that must be visited by exactly one salesman. In fixed-destination MmTSP, each salesman must end their tour at the vertex they started at, which is known to be NP-Hard.

Proof. We form a reduction from fixed-destination MmTSP to fixed-MdMtHPP as follows. Let G be the graph for the fixed-destination MmTSP problem. Form a new graph G' by taking the set of destination vertices and depots from G. If any depot has more than one salesman that starts at it, then form a new depot at this same location and assign the additional salesman to this depot. For every salesman, create a terminal vertex that lies on top of the salesman's starting vertex. Solve the fixed-MdMtHPP that we have just formed. The routes from our solution to fixed-MdMtHPP will also be a solution to the fixed-destination MmTSP.

This reduction can be performed in linear time in terms of the number of vertices in G. As we have found a polynomial time reduction from fixed-destination MmTSP to fixed-MdMtHPP, and we know that fixed-destination MmTSP is NP-Hard, then fixed-MdMtHPP must also be NP-Hard. □

We present a MINLP that provides an exact solution to the fixed-MdMtHPP in the following subsection. However, as we have just shown, the fixed-MdMtHPP is NP-Hard and MINLP are not tractable for larger inputs. To address this limitation of our MINLP solution, we present a heuristics-based solution in Section 6.2.

6.1 MINLP Formulation for the fixed-MdMtHPP

In this subsection we formulate a MINLP to solve the underlying fixed-MdMtHPP for our drone path finding problem. We term this approach the *fixed*-MINLP, or f-MINLP. We also show how a simplified version of the formulation can be used to solve the general case of the fixed-MdMtHPP.

Our f-MINLP formulation minimizes mission completion time by jointly minimizing the distance of each sub-tour and maximizing the speed that the drone travels on each sub-tour. Due to the relationship between travel distance and speed discussed in Section 4, we cannot simply use a fixed drone speed but must adapt the speed based on distance, which is why we optimize these two jointly. We use a variation of the Miller-Tucker-Zemlin formulation [21, 37] for the capacitated VRP because it provides us with the flexibility to force fixed depots and terminals. Although our fixed-start assumption prevents us from making guarantees on the optimality of a solution to the MT-OTM problem, our f-MINLP can find an optimal solution to the underlying fixed-MdMtHPP.

In general, we use capital letters for sets, capital letters with subscripts as decision variables, and lowercase letters as constants. We refer the reader to Table 2 for a list of symbols used in our formulation. For the set of waypoints $i, j \in P$ and sub-tour k in the set of tours K, let E_{ijk} be a binary decision variable that determines if edge $(i,\ j)$ is included in tour k. We denote the Euclidean distance between i and j as d_{ij}. We use binary variables D^d_{ki} and D^t_{jk} to connect each depot k to some waypoint i, and some waypoint j to some terminal k, respectively. Let L_k and S_k be continuous variables for the total distance that the UAV must fly on sub-tour k and the constant speed of the UAV on k, respectively. To prevent cycles within each sub-tour, we use the integer variable U_i to give ordering assignments to each waypoint.

Our f-MINLP formulation of the problem is as follows.

$$\min \sum_{k \in K} \frac{L_k}{S_k} \tag{12}$$

subject to:

$$L_k = \sum_{i \in P} \sum_{j \in P} d_{ij} E_{ijk} + \sum_{i \in P} d_{ki} D^d_{ki} + \sum_{j \in P} d_{jk} D^t_{jk}, \qquad \forall_{k \in K} \tag{13}$$

$$S_k \leq \frac{\sqrt{c_1 - c_2 L_k}}{c_3} + c_4, \qquad \forall_{k \in K} \tag{14}$$

$$1 - \mathcal{M}(1 - E_{ijk}) \leq U_j - U_i \leq 1 + \mathcal{M}(1 - E_{ijk}), \qquad \forall_{i,j \in P}, \forall_{k \in K} \tag{15}$$

$$1 \leq U_i - U^d_k D^d_{ki} \leq 1 + \mathcal{M}(1 - D^d_{ki}), \qquad \forall_{i \in P} \forall_{k \in K} \tag{16}$$

$$\sum_{j \in P} E_{j,i,k} + D^d_{ki} = \sum_{j \in P} E_{i,j,k} + D^t_{ki}, \qquad \forall_{i \in P} \forall_{k \in K} \tag{17}$$

$$\sum_{j \in P} \sum_{k \in K} E_{j,i,k} + \sum_{k \in K} D^d_{ki} + \sum_{j \in P} \sum_{k \in K} E_{i,j,k} + \sum_{k \in K} D^t_{ki} = 2, \qquad \forall_{i \in P} \tag{18}$$

$$\sum_{i \in P} D^d_{ki} = 1, \qquad \forall_{k \in K} \tag{19}$$

$$\sum_{i \in P} D^t_{ki} = 1, \qquad \forall_{k \in K} \tag{20}$$

$$S_k \leq v_{max}, \qquad \forall_{k \in K} \tag{21}$$

$$L_k \leq d_{max}, \qquad \forall_{k \in K} \tag{22}$$

Table 2. MINLP Formulation Symbols

Symbol	*Type*	*Description*
D^d_{ki}	Binary Variable	1 if waypoint i is the first stop on sub-tour k, 0 otherwise
D^t_{jk}	Binary Variable	1 if waypoint j is the last stop on sub-tour k, 0 otherwise
E_{ijk}	Binary Variable	1 if edge (i, j) is in sub-tour k, 0 otherwise
K	Set	Set of sub-tours, where $k \in K$ is a sub-tour
L_k	Continuous Variable	Total distance of sub-tour k
L^t_{jk}	Continuous Variable	Distance from waypoint j to terminal of sub-tour k
L^d_{ki}	Continuous Variable	Distance from depot of sub-tour k to waypoint i
P	Set	Set of waypoints to visit, where $i, j \in P$ are waypoints
S_k	Continuous Variable	Assigned drone speed for sub-tour k
T_k	Continuous Variable	Start (T^d_k) and terminal (T^t_k) times for sub-tour k
X_k	Continuous Variable	x coordinate for depot (X^d_k) and terminal (X^t_k) of sub-tour k
Y_k	Continuous Variable	y coordinate for depot (Y^d_k) and terminal (Y^t_k) of sub-tour k

Our objective function (12) is minimizing the total time required for the UAV to travel all sub-tours. Note that we can remove possible domain violations in the objective using an additional auxiliary variable, A_k, to separate L_k and S_k as $L_k = A_k S_k$ then minimizing A_k. We forego adding additional variables here for brevity. Constraint (13) forces variable L_k to equal the distance of sub-tour k. Constraint (14) adapts a sub-tour speed based on the distance of sub-tour k and is derived from Eq. (11).

Constraint (15) enforces a tight numbering scheme for consecutive waypoints in each sub-tour where $\mathcal{M}$ is some sufficiently large number which will be further discussed shortly. Constraint (16) forces the waypoint after depot k to be assigned a sequence number of $U^d_k + 1$, where U^d_k is an implied, fixed sequence number assigned to depot k that remains constant. For each waypoint i on sub-tour k, we want to assign a sequence number to U_i that is within a designated range to force waypoints that are on the same sub-tour to be numbered together. We define the bounds of this numbering range for sub-tour k using implied depot and terminal sequence numbers U^d_k and U^t_k, respectively. If l_m is the maximum number of waypoints that can be assigned to a sub-tour then we want to have l_m sequence numbers available between U^d_k and U^t_k. When given n waypoints to visit on m sub-tours with at least one waypoint on each sub-tour, by the pigeon hole principle, $l_m = n - m + 1$. For the first sub-tour, if we set $U^d_1 = 0$ then U^t_1 must be $l_m + 1$. This makes $U^d_2 = l_m + 2$. Following this trend, we find that for any sub-tour k, $U^d_k = (l_m + 2)(k - 1)$ and $U^t_k = k(l_m + 1) + (k - 1)$. In constraints (15) and (16) we want a value for $\mathcal{M}$ that is large enough to allow for all feasible sequence number assignments for each U_i. To keep a tight bound on our constraints, we set $\mathcal{M} = U^t_m = ml_m + 2m - 1$.

Constraint (17) ensures that the in-degree is equal to the out-degree of each waypoint while constraint (18) forces each waypoint to have a degree of two. Constraints (19) and (20) force every depot and terminal to be used, respectively. Finally, constraints (21) and (22) bound the maximum allowable speed of the UAV and maximum allowable distance of any sub-tour, respectively.

The numbering scheme described above is what allows us to fix the depots with their corresponding terminals. We are also ensuring that each sub-tour contains at least one waypoint by not defining an edge from depot k to terminal k in our formulation.

We can modify our f-MINLP formulation to solve the general case of fixed-MdMtHPP by making the objective function to be

$$\min \sum_{k \in K} L_k \tag{23}$$

and removing constraints (14), (21) and (22). The general fixed-MdMtHPP formulation avoids the additional complexity of variable multiplication seen in our f-MINLP formulation but does not adapt UAV speed and is not directly optimizing mission completion time.

Our f-MINLP formulation provides an exact solution to the fixed-MdMtHPP created by fixing the drone's launch and receive locations. However, the fixed-MdMtHPP is an NP-Hard problem and MINLPs in general take a long time to solve to completion. Therefore, we do not expect our f-MINLP formulation to be able to solve large problem inputs in a reasonable amount of time. In the following subsection, we address this limitation by introducing a heuristics-based algorithm that can handle large problem inputs.

6.2 Heuristic Solution for the fixed-MdMtHPP

Mixed-Integer Non-Linear Programs are often very hard to solve, even for commercial solvers on high-performance computers, so in this subsection we propose a more tractable approach that combines a heuristics-based k-means clustering algorithm and a TSP solver. The general concept is to partition the waypoints into m groups then form sub-tours by solving the TSP. After finding waypoint grouping and ordering each sub-tour, we schedule speeds for the sub-tour by finding the distance of the sub-tour and using Eq. (10).

To partition the waypoints, we use Lloyd's algorithm to form m k-means clusters [29]. We use the centroid of each p_k^d and p_k^t pair as the initial cluster centroids. We then limit the number of iterations that the algorithm runs which prevents the cluster centroids from migrating too far away from their corresponding p_k^d and p_k^t pairs. Once we have put each waypoint into a cluster, we combine the clusters with the corresponding p_k^d and p_k^t. We then solve a TSP on the resulting graph and force the edge connecting p_k^d and p_k^t to be part of the solution.

There are several well-studied approaches to solving the TSP. In this work we use both an Integer Program (IP) and the Lin-Kernighan-Helsgaun (LKH) heuristic [20]. We use the IP formulation found in [45] that uses sub-cycle cuts to enforce closed tours. To avoid an exponential number of sub-cycle cuts, we treat these as lazy constraints where the constraint is only added to the solver when a found solution would break the constraint, a feature available in many commercially available solvers such as Gurobi. To get the LKH solver to force edge connecting p_k^d and p_k^t to be part of the solution we set this edge equal to 0 and multiply all other edges going out of each of these vertices by a constant. We term the IP version of this solution the k-IP approach and the LKH solution the k-TSP approach. Although neither the k-IP nor the k-TSP algorithms provide theoretical guarantees on solution quality, we should expect both algorithms to run faster than solving the f-MINLP formulation.

The f-MINLP formulation and the k-IP and k-TSP algorithms all make the fixed-start assumption, where the drone is launched at the beginning of the considered time horizon and all subsequent sub-tours start as early as possible. The results of our numerical simulations in Section 8 show that this assumption is valid when all waypoints are located close to the GV at the start of the considered time horizon but can cause these algorithms to fail to find a solution when the waypoints are further away. To address this problem, we next consider more flexible solutions that remove the fixed-start assumption.

7 Generalized MT-OTM Solutions

Algorithm 1 can find a solution to the MT-OTM problem if given function *path-planning()* - which solves instances of the fixed-MT-OTM problem. In the previous section we discussed how to implement *path-planning()* by assuming that the first sub-tour should start at the beginning of the considered time horizon and that every consecutive sub-tour should start as early as possible (the "fixed-start assumption"). In this section, we remove the fixed-start assumption and propose more generalized solutions.

We first look at a MINLP that provides an exact solution to the MT-OTM problem then propose a heuristic-based solution that is more tractable. These approaches solve the fixed-MT-OTM problem directly rather than solving for a fixed-MdMtHPP as presented in the previous section.

7.1 MINLP Formulation for the MT-OTM Problem

In this subsection we formulate a new MINLP to solve the generalized MT-OTM Problem. Our formulation minimizes mission completion time (the end time of the final sub-tour, τ_m^t) by jointly minimizing the distance of each sub-tour and maximizing the speed that the drone travels on each sub-tour. Additionally, our formulation for the generalized problem dynamically selects p_k^d and p_k^t, the starting and end location of sub-tour k. We term this solution *dynamic*-MINLP, or *D*-MINLP.

Our *D*-MINLP formulation builds on the f-MINLP formulation presented in Section 6.1 with a few additional variables. Let L_{ki}^d be a continuous variable that tracks the distance from the depot of sub-tour k to waypoint i while continuous variable L_{jk}^t tracks the distance from waypoint j to the sub-tour's terminal. Let T_k^d and T_k^t be continuous variables for the start and end time of sub-tour k, respectively. Continuous variables X_k^d and Y_k^d are the x and y coordinates, respectively, of the depot for sub-tour k and variables X_k^t and Y_k^t are the coordinates for the terminal of the sub-tour. Similarly to the f-MINLP formulation, the *D*-MINLP formulation uses the S_k variable and Eq. (10) to schedule speeds as part of the optimization problem.

Our *D*-MINLP formulation of the problem is as follows:

$$\min T_m^t \tag{24}$$

subject to Eq. (13), (14) (15), (16), (17), (18), (19), (20), (21), (22), and:

$$L_{ki}^d = \sqrt{(X_k^d - x_i)^2 + (Y_k^d - y_i)^2}, \qquad \forall_{k\in K}\forall_{i\in P} \tag{25}$$

$$L_{jk}^t = \sqrt{(X_k^t - x_j)^2 + (Y_k^t - y_j)^2}, \qquad \forall_{j\in P}\forall_{k\in K} \tag{26}$$

$$(X_k^d, Y_k^d) = p_d(T_k^d), \qquad \forall_{k\in K} \tag{27}$$

$$(X_k^t, Y_k^t) = p_d(T_k^t), \qquad \forall_{k\in K} \tag{28}$$

$$T_1^d \geq 0 \tag{29}$$

$$T_k^d \geq T_{k-1}^t + t_b, \qquad \forall_{k\in K, k\geq 2} \tag{30}$$

$$T_k^t = T_k^d + \frac{L_k}{S_k}, \qquad \forall_{k\in K} \tag{31}$$

Our objective function (24) minimizes mission completion time by minimizing T_m^t, the end time of the final sub-tour. Constraint (25) sets variable L_{ki}^d based on the distance from depot k to waypoint i while constraint (26) sets variable L_{ki}^d

to be the distance from waypoint j to terminal k. Constraints (27) and (28) ensure that the location of each depot and terminal, respectively, is based on $p_d(t)$, the function for the GV's trajectory discussed in Section 3. Constraints (29) and (30) bound the start time of sub-tour k based on the considered time window (for the former) and the previous sub-tour's end time (for the latter). Constraint (31) sets the sub-tour's end time.

Our *D*-MINLP formulation finds an optimal solution to the fixed-MT-OTM Problem and makes Algorithm 1 an exact algorithm to the larger MT-OTM Problem when used as function *path-planning()*. However, even on the strongest commercial solvers, MINLPs are not easy to solve to optimality and we have already shown that the MT-OTM Problem is NP-Hard in Section 3. In the following section we propose a heuristics-based method for solving the fixed-MT-OTM Problem that is more tractable than our *D*-MINLP formulation.

7.2 Heuristic Solutions for the fixed-MT-OTM Problem

Our heuristics-based solution for solving the fixed-MT-OTM Problem builds on the concepts presented in Section 6.2. We use a clustering algorithm to form m clusters that serve as our m sub-tours. We pick start and end locations for each sub-tour then treat the clusters with start and end locations as a TSP and iteratively solve this TSP and re-selecting start and end locations until the solution becomes consistent and stops changing from iteration to iteration. We term this approach the *Dynamic*-TSP, or *D*-TSP. This solution requires $p_b^{-1}(x_c, y_c)$, the inverse of the GV's trajectory function that gives you the time that the GV will pass through (x_c, y_c), or a method for approximating the inverse of $p_b(t)$.

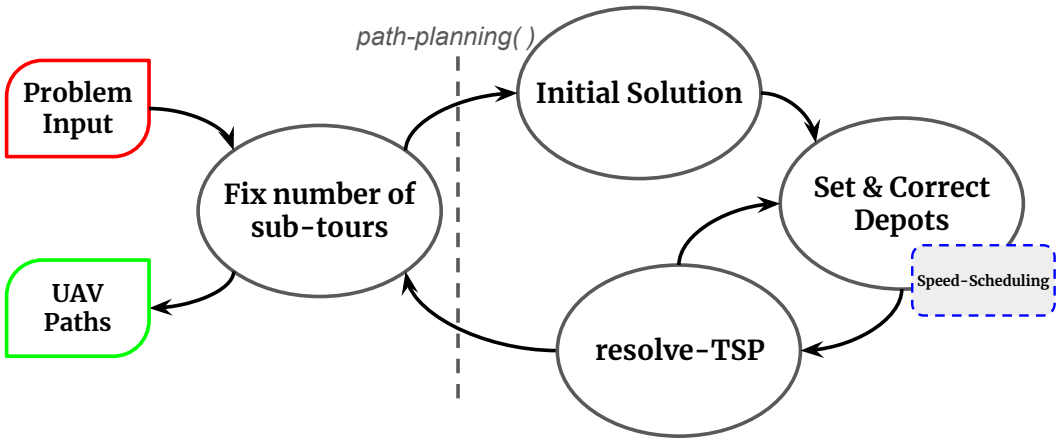


Fig. 4. A flow graph for solving the MT-OTM Problem directly. We initially fix the number of sub-tours, run the *path-planning()* function, and then increase the number of sub-tours. The *path-planning()* function finds an initial solution by clustering together waypoints then iteratively solves a TSP on each cluster and adjusts depot and terminal locations until the solution is consistent.

Figure 4 shows the general flow for our *D*-TSP algorithm for directly solving the MT-OTM Problem while Algorithm 3 lists the details of our implementation. Starting on line 1, we use a k-means cluster algorithms to form m sets of waypoints (P_m). Using the function *create-depots()* on line 2, we create m pairs of depots and terminals (stored in set Γ_m). For each cluster, the function finds the closest point along the GV's trajectory to the cluster's centroid. Suppose this position is (x_c, y_c). The function then queries $p_b^{-1}(x_c, y_c)$ to determine the time that the GV will be at this location. Suppose this time is t_c. The *create-depots()* function then estimates the time required for a drone to visit all waypoints in the cluster using a minimum spanning tree and assuming the drone moves at v_{max}. Suppose this time is t'_c. The

function then places the depot for the cluster at the location of the GV's position at $t_c - \frac{t'_c}{2}$ and the terminal at $t_c + \frac{t'_c}{2}$. The algorithm then treats the pairs of depots and terminals in Γ_m and waypoint clusters in P_m as a set of TSP, which is solved using the LKH heuristic on line 3.

Algorithm 3 *D*-TSP

Input: P : set of waypoints to visit, $p_b(t)$: GV position function, m : number of sub-tours
Output: Φ_m : complete drone tour with m sub-tours

1: $P_m \leftarrow k\text{-}means(P, m)$
2: $\Gamma_m \leftarrow create\text{-}depots(P_m)$
3: $\Delta_m \leftarrow solve\text{-}TSP(P_m, \Gamma_m),\ \Delta'_m \leftarrow \emptyset$
4: **while** $\Delta_m \neq \Delta'_m$ **do**
5: $\quad \Delta'_m \leftarrow \Delta_m,\ \Gamma'_m \leftarrow \emptyset$
6: $\quad$ **while** $\Gamma_m \neq \Gamma'_m$ **do**
7: $\quad\quad \Gamma'_m \leftarrow \Gamma_m$
8: $\quad\quad S_m, \Gamma_m \leftarrow set\text{-}depots(\Delta_m)$
9: $\quad\quad \Gamma_m \leftarrow correct\text{-}depots(\Gamma_m)$
10: $\quad$ **end while**
11: $\quad \Delta_m \leftarrow solve\text{-}TSP(P_m, \Gamma_m)$
12: **end while**
13: **return** $\Phi_m \leftarrow \{\Delta_m, S_m, \Gamma_m\}$

On line 8, the function *set-depots()* sets sub-tour speeds based on the distance of the sub-tour and Eq. (10) then corrects the position of the depots and terminals so the solution meets problem constraints (4), (5), and (6). This logic is based on centering the depot and terminal around the mid-point time t_c that was discussed earlier. This function also attempts to move the start time of a sub-tour to an earlier time if the distance of the sub-tour is less than d_{vm}, the distance that the drone can travel if moving at v_{max}. The idea behind moving up the start time of a sub-tour is that if the tour can start earlier then it can potentially end earlier and reduce the time required to visit all waypoints.

Both the *create-depots()* and *set-depots()* functions ignore that a consecutive sub-tour cannot start earlier than the end time of the previous sub-tour plus the time required to swap out batteries (constraint 3). Function *correct-depots()* on line 9 fixes constraint 3 violations by solving the following convex optimization problem:

$$\min \sum_{k=1}^{m} (T_k^d - t_k^d)^2 \tag{32}$$

subject to:

$$T_{k-1}^d + \hat{t}_k + t_b \leq T_k^d, \qquad 2 \leq k \leq m \tag{33}$$

where T_k^d is a continuous variable for the start time and t_k^d is a constant that represents the desired start time for sub-tour k. Constant $\hat{t}_k$ is the time duration of sub-tour k and t_b is the time required to swap out batteries on the drone at the GV.

After finding a consistent solution that meets constraints (2), (3), (4), (5), and (6), we run the TSP solver again on line 11 to verify that the sub-tour ordering cannot be improved based on how we moved the depots and terminals. The while-loop on line 4 runs until the sub-tours stop changing between iterations and the while-loop on line 6 runs until

the location of the depots and terminals stops changing. In practice, these while-loops should have a time-out criteria to avoid an infinite loop should the algorithm never converge onto a consistent solution.

Algorithm 3 can be performed in polynomial time. To ensure that the k-means clustering algorithm converges in a timely manner we can add a time-out condition based on $n = |P|$. The *create-depots()* function finds a minimum spanning tree, which can be done in $O(n^2 log(n))$. The LKH solver is approximated to run in $O(n^{2.2})$ [19]. The function *set-depots()* can be done in $O(n)$. Convex optimization problems can be solved to optimality in cubic time based on the number of decision variables [63], which means function *correct-depots()* can run in $O(m^3)$. Suppose we limit both while-loops to time-out after a constant number of iterations, then the inner loop at line 6 will have a runtime of $O(n)$ while the outer loop at line 4 will be dominated by the runtime of the LKH solver at $O(n^{2.2})$. This makes the run-time of Algorithm 3 dependent on the LKH solver at $O(n^{2.2})$. If we use Algorithm 3 as function *path-planning()* from Algorithm 1, then the runtime complexity of the D-TSP approach will be $O(n^{3.2})$.

Similar to the k-IP and k-TSP algorithms, the D-TSP algorithm provides no theoretical guarantees on solution quality. However, we should expect the D-TSP algorithm to compute solutions to the general MT-OTM problem much faster than solving the D-MINLP formulation.

8 Simulation Evaluation

In this section we discuss our evaluation of our framework for solving the MT-OTM Problem in simulation using parameters from previous field testing and commercially available hardware. We start this section with a discussion on our evaluation setup. We then look at our proposed methods for the MT-OTM Problem where the start points of each sub-tour are fixed and then remove this assumption and evaluate our generalized solutions. We conclude this section with a review of run times for all considered approaches.

8.1 Simulation Setup

Our simulations were conducted on a machine with an Intel 3.4 GHz 16-Core CPU and 64 GiB of RAM. We use Gurobi Optimizer version 10.0.3 for our optimization solver. Our solution framework is implemented in C++ and provided as open-source[1].

8.1.1 Realistic Energy Model from Field Tests. In Section 4 we presented $\mathcal{P}(v)$, a theoretical equation that represents the power consumed by a drone based on speed v. $\mathcal{P}(v)$ was verified through field testing in [54]. For their specific drone, $\mathcal{P}(v)$ can be approximated as

$$\mathcal{P}(v) = 0.07v^3 + 0.0391v^2 - 13.196v + 390.95 \tag{34}$$

If we equip the drone with a commercially available LiPo battery rated at 2,200 mAh, 12.6 volt, then we can plug Eq. (34) into Eq. (8) and get

$$d(v) = \frac{99,792v}{0.07v^3 + 0.0391v^2 - 13.196v + 390.95} \tag{35}$$

We can use Eq. (35) to determine d_{max}, the maximum distance that the drone can travel. By approximating the curve of Eq. (35) between d_{vm} and d_{max} as a polynomial, we can find an approximation for $\mathcal{D}^{-1}$ between v_{opt} and v_{max} in the form of Eq. (8).

[1] github.com/pervasive-computing-systems-group/MT-OTM-Solver

8.1.2 Metrics and Baseline Approaches. In all simulations, our primary metric was the total time required to visit all waypoints and then return to the GV. This includes the time required to swap out batteries between consecutive sub-tours, incentivizing each approach to reduce the number of sub-tours required to visit all waypoints. We also looked at the percentage of inputs where each approach was able to find a solution and the computation time for each approach.

For approaches that assume a fixed start time we compared against a baseline adapted from the heuristics-based approach found in [33]. We refer to this approach as *tour-splitting*, which is further discussed below. When evaluating our generalized MT-OTM Problem solutions we compared against the approaches that assume a fixed start time and as well as a dynamic implementation of the *tour-splitting* algorithm.

The *tour-splitting* (TS) approach finds a TSP tour on the entire set of waypoints using the LKH heuristic [20] then has the drone follow this tour until it runs out of energy. We chose this approach because it avoids repeatedly solving for Hamiltonian paths, allows us to adapt drone speeds to sub-tour distances, and has been proposed for similar problems in recent literature [3, 4, 33]. Additionally, this approach gives us an example of simplifying the problem into a variation of the traditional TSP and then using a TSP-based heuristic to solve the problem.

This approach as proposed in [33] finds a cycle while in the MT-OTM Problem we wish for the drone to generally follow the path of the GV. This requires us to slightly adapt the original approach instead of using it directly. To adapt this approach, we find a minimum distance Hamiltonian path using the LKH heuristic that starts from the waypoint closest to the vehicle's starting point and ends where we predict the GV to be at the end of the entire data-collection mission, as described in Section 5. We then divide this total path into m roughly equal segments. For each segment, we form a sub-tour by determining where the GV will be at for the beginning of the sub-tour based on the previous sub-tour. We then iteratively approximate the maximum possible drone speed allowed for the sub-tour using Eq. (10) and determine a corresponding sub-tour terminating location until we settle on a consistent solution. The approach is plugged into Algorithm 1 as the *path-planning()* function.

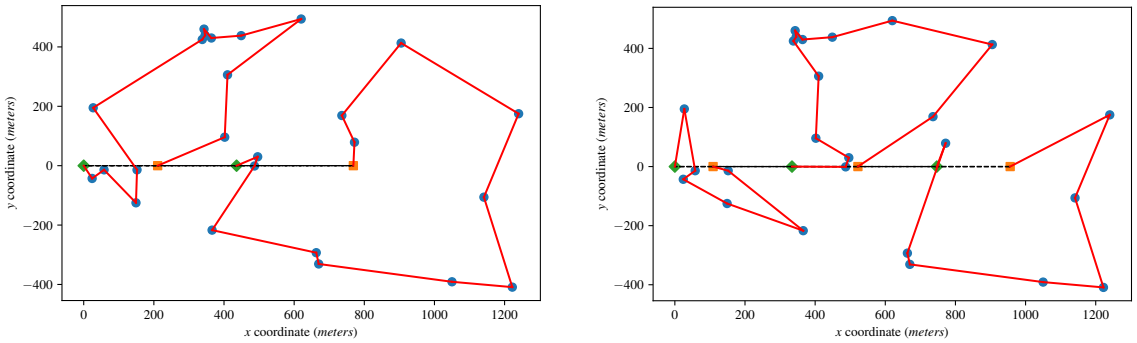
The basic TS approach holds the fixed-start assumption that was discussed in Section 6. To compare against our dynamic solutions (D-MINLP and D-TSP), we further modify the TS approach to allow it to dynamically set sub-tour start and end locations. The adaptive TS (a-TS) selects depot positions by finding the closest point along the GV's trajectory to the beginning of each tour that does not break problem constraints (2), (3), (4), (5), and (6).

8.2 Comparison of Fixed-Start Approaches

We ran all three fixed-start approaches on randomized graphs with n ranging from 5 up to 80 at increments of 5. We generated 50 graphs at each increment. To keep the results comparable across each input graph, we had the GV move at a fixed speed of 2.5 m/s along the x-axis.

Figure 5a shows an example output of the f-MINLP approach and Figure 5b shows the output of the k-IP approach on a randomly generated graph with 25 waypoints and a linear GV trajectory. The f-MINLP approach found a superior solution with a mission completion time of 307.5 s while the k-IP approach found a solution with a completion time of 382.4 s. However, the k-IP only took 0.038 s to compute this solution while using the f-MINLP formulation took 130.9 s to compute a solution. This tradeoff is further demonstrated in Section 8.2.

Figure 6a (*top*) shows how the number of waypoints affects the average mission completion time. The error bars show the standard deviation for each approach. We stopped the f-MINLP at $n = 30$ due to long computation times, which are further discussed below. The TS and k-IP approaches were not always able to find a feasible solution and their failure rates are also documented below. On average, the f-MINLP and the k-IP approaches provide a 23.8% and 14.5%

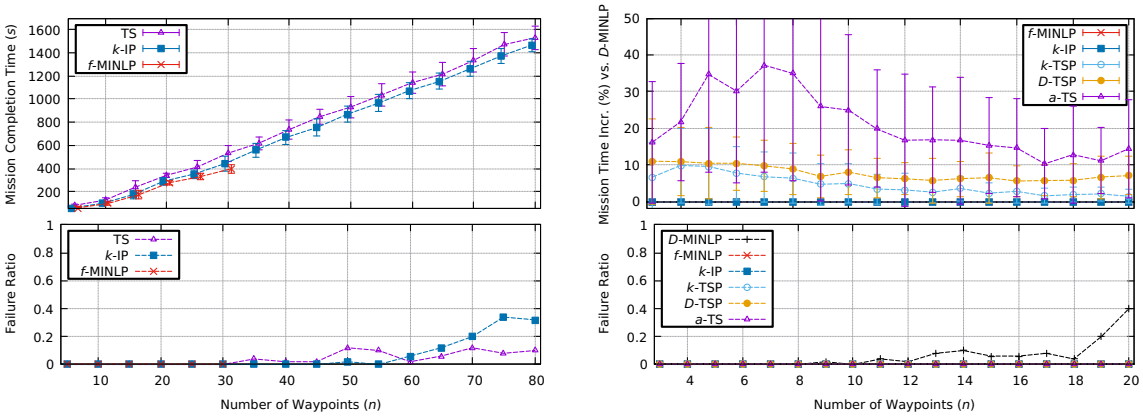


(a) Drone paths generated using f-MINLP (b) Drone paths generated using k-IP

Fig. 5. Sample drone paths generated using our approaches. The blue circles are the navigational waypoints, the dashed, black line is the GV's trajectory, the red lines are the drone's path where the green diamonds are drone launch points and the orange squares are drone receiving points.

improvement over the TS approach, respectively. The k-IP's performance is not as good as the f-MINLP solution but only averages a 3.8% increase in mission completion time over the f-MINLP approach and provides a nice alternative to the f-MINLP in larger sized problems. We note that the results in Figure 6a were first presented in our conference publication [16].

Figure 6a (*bottom*) shows the ratio of graphs where each approach failed to find a solution. The k-IP approach failed to find a valid solution for 6.6% of the graphs while the TS approach failed at 4.3% of the graphs. This suggests



(a) f-MINLP and k-IP vs. baseline (b) Comparison against f-MINLP optimal solution

Fig. 6. Simulation results for fixed-start solutions (*a*) and dynamic solutions (*b*). The top plot of (*a*) shows the impact of the number of waypoints on mission completion time for the f-MINLP, k-IP, and TS approaches. The top plot of (*b*) shows the impact of the number of waypoints on mission completion time for the f-MINLP, k-IP, k-TSP, D-TSP, and adaptive TS approaches when compared against an optimal solution found using the D-MINLP approach. The bottom plots for both (*a*) and (*b*) show the ratio of graphs where each approach failed to find a solution.

that although k-IP outperforms TS in mission completion time, TS may be able to find more solutions than k-IP. The f-MINLP approach was able to find a solution of all graphs up to the cut-off point ($n = 30$).

8.3 Comparison of Dynamic Approaches

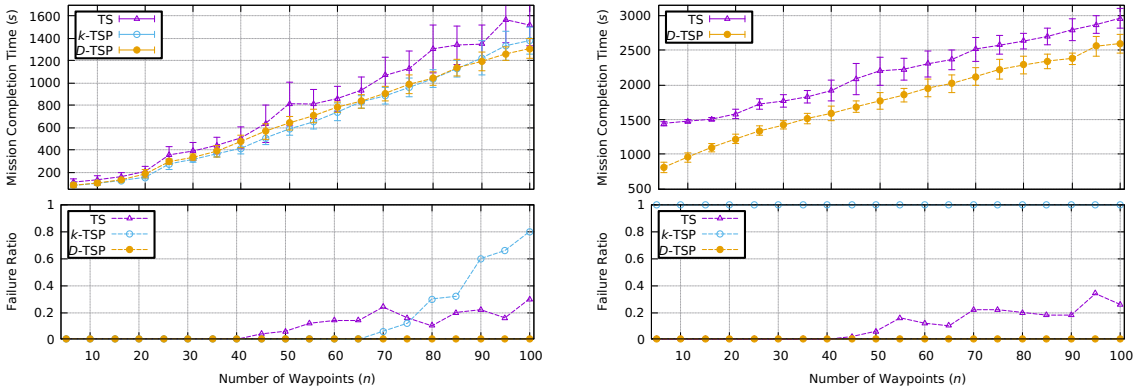
In this section we evaluate our proposed generalized solutions for the MT-OTM Problem that remove the fixed-start time assumption. We ran each approach on multiple randomized sets of graphs with varying values for n. In addition to linear GV trajectories, we also compared our various solutions on instances where the GV moves along a sinusoidal path. The details for each set of inputs is further described below.

8.3.1 Mission Completion Time Compared to Optimal Solution. For our generalized approaches we first ran the D-MINLP, f-MINLP, k-IP, k-TSP, D-TSP and adaptive TS on a test set where the GV moved along a linear trajectory at 2.5 m/s. To avoid letting the solver get stuck on hard inputs, we set the D-MINLP solution to time-out at 1,200 s. The input set had n ranging from 3 up to 20 at increments of 1. We generated 50 graphs at each increment. We stopped at 20 because this is when the D-MINLP began to struggle to find a solution in a reasonable amount of time.

Figure 6b (*top*) shows the percent increase in mission completion time for each method when compared against the exact D-MINLP approach. The error bars show the standard deviation. The f-MINLP and k-IP approaches found a comparable solution to the D-MINLP for all inputs. On average, the k-TSP and D-TSP approaches found a solution that was within 4.5% and 7.7% of the optimal solution, respectively, while the adaptive TS approach averaged 20.8% increase in completion time compared to the optimal.

Figure 6b (*bottom*) shows the ratio of graphs where each approach failed to find a solution. All approaches except for the D-MINLP were able to find a solution on this data set. We set a timeout of 1,200 s for the D-MINLP and all failures were from the solver timing out.

8.3.2 Larger Inputs with Linear Trajectory Gound Vehicle. We then ran the k-TSP, D-TSP and adaptive TS on larger inputs with a linear GV trajectory. We chose to use the k-TSP approach here instead of the k-IP because we wanted to compare the D-TSP approach against another polynomial time solution. The GV moved at 1.5 m/s and n ranged from 5



(a) Near-start linear data set

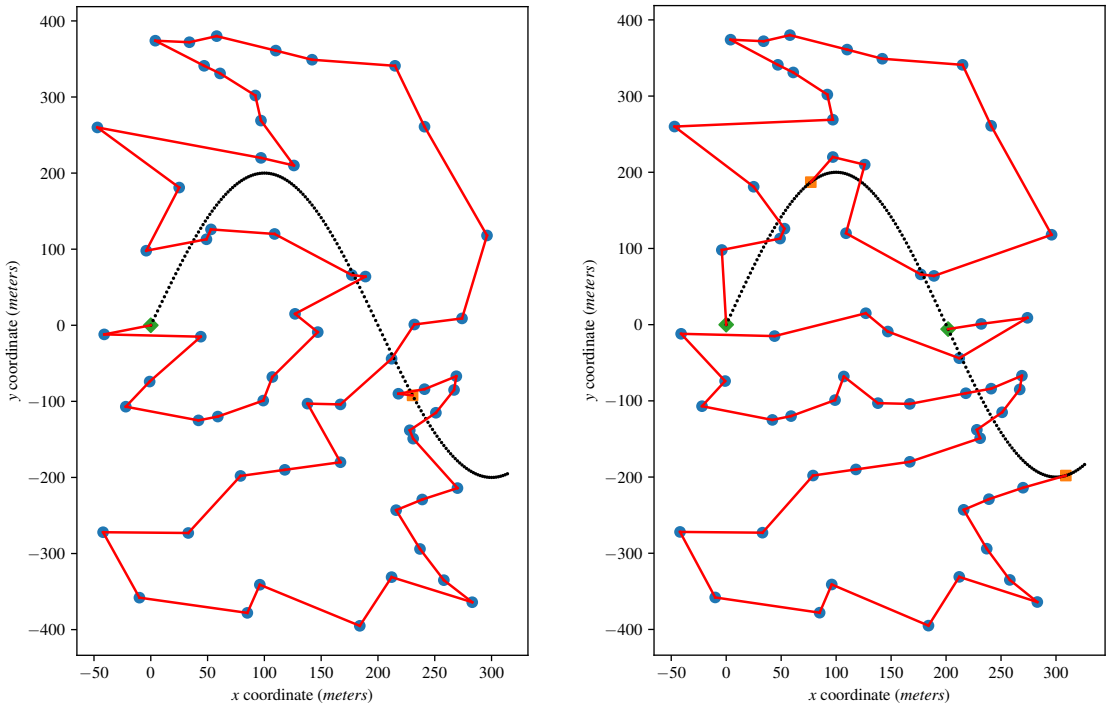
(b) Far-start linear data set

Fig. 7. Simulation results for dynamic solutions on inputs with linear GV trajectories. The top plots show the impact of the number of waypoints on mission completion time. The bottom plots show the ratio of graphs where each approach failed to find a solution.

up to 100 in increments of 5 with 50 randomly generated plots at each increment. We generated two data sets with the setup. The first has the waypoints beginning close to the GV's position at the beginning of the considered time window, which we refer to as the "*near-start*" linear data set. The second has the GV start 2 kilometers before the waypoints, which we refer to as the "*far-start*" linear data set.

Figure 7 shows the results of these two test scenarios with Figure 7a showing the near-start results and Figure 7b showing the far-start results. On the near-start linear data set, the k-TSP and D-TSP approaches performed nearly the same with the former slightly outperforming the latter. When compared against adaptive TS, the k-TSP and D-TSP approaches improved completion time by an average of 18.3% and 14.8%, respectively. As shown in the bottom of Figure 7a, the k-TSP approach struggled to find a solution as n grew above 70. We note that Figure 7a is only showing results for inputs where all three algorithms were able to find a solution.

Figure 7b shows the results for the far-start linear data set. The k-TSP approach failed to find any solution on these inputs because the GV started at a distance from the first waypoints and the fixed-start assumption caused the drone to launch too early. However, both the D-TSP and adaptive TS approaches were able to solve inputs from this data set with the former finding a solution to all given inputs. The D-TSP improved completion time by an average of 19.2% when compared to the adaptive TS.



(a) Drone path generated using k-TSP

(b) Drone path generated using D-TSP

Fig. 8. Sample drone paths for a GV moving along a sinusoidal trajectory. The blue circles are the navigational waypoints, the dashed, black line is the GV's trajectory (which moves from left to right), the red lines are the drone's path where the green diamonds are drone launch points and the orange squares are drone receiving points.

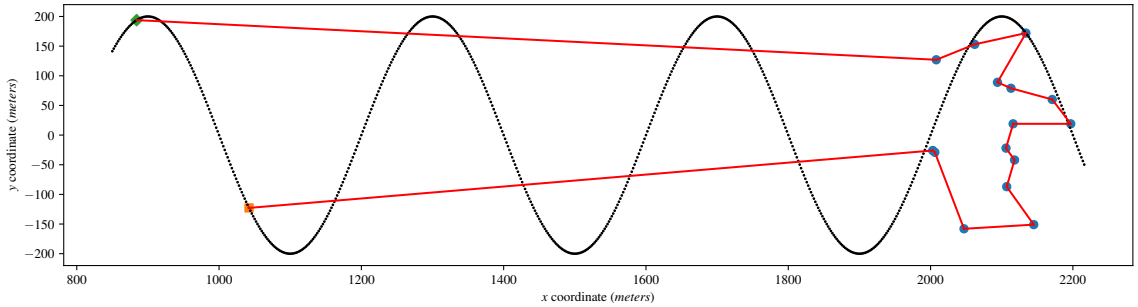
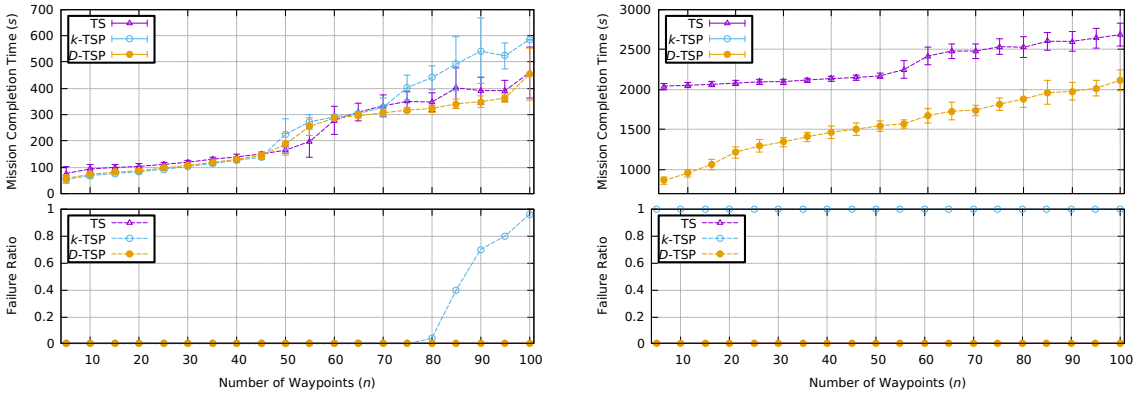


Fig. 9. Drone path generated using *D*-TSP for a GV moving along a sinusoidal trajectory. The blue circles are the navigational waypoints, the dashed black line is the GV's trajectory (which moves from left to right and begins at position (0, 0)), the red lines are the drone's path where the green diamonds are drone launch points and the orange squares are drone receiving points.

8.3.3 Larger Inputs with Sinusoidal Trajectory Gound Vehicle. We next ran the *k*-TSP, *D*-TSP and adaptive TS test data sets where the GV moves along a sinusoidal trajectory. The GV moved in the positive x direction at 1 m/s while the vehicle's y-coordinate was defined as $y(t) = 200sin(\frac{2\pi t}{400})$. We again generated two data sets with this setup; the near-start sinusoidal data set where the waypoints begin close to the GV's starting position and the far-start sinusoidal data set where the GV starts 2 kilometers before the waypoints.

Figure 8a shows an example output of the *k*-TSP approach and Figure 8b shows an example output of the *D*-TSP approach on a graph with 60 waypoints from the near-start sinusoidal data set. The *k*-TSP approach found a superior solution with a single sub-tour and mission completion time of 230.5 s while the *D*-TSP approach found a solution with two sub-tours and a 308.9 s completion time. Figure 9 shows the solution found using the *D*-TSP approach on graph with 15 waypoints from the far-start sinusoidal data set. In this example, at the beginning of the time window ($t = 0$)



(a) Near-start sinusoidal data set (b) Far-start sinusoidal data set

Fig. 10. Simulation results for dynamic solutions on inputs with sinusoidal GV trajectories. The top plots show the impact of the number of waypoints on mission completion time. The bottom plots show the ratio of graphs where each approach failed to find a solution.

the GV began at position (0, 0). The *D*-TSP approach found a solution that begins at $t = 884.2$ and ends at $t = 1042.1$ while the *k*-TSP approach was unable to find any solution.

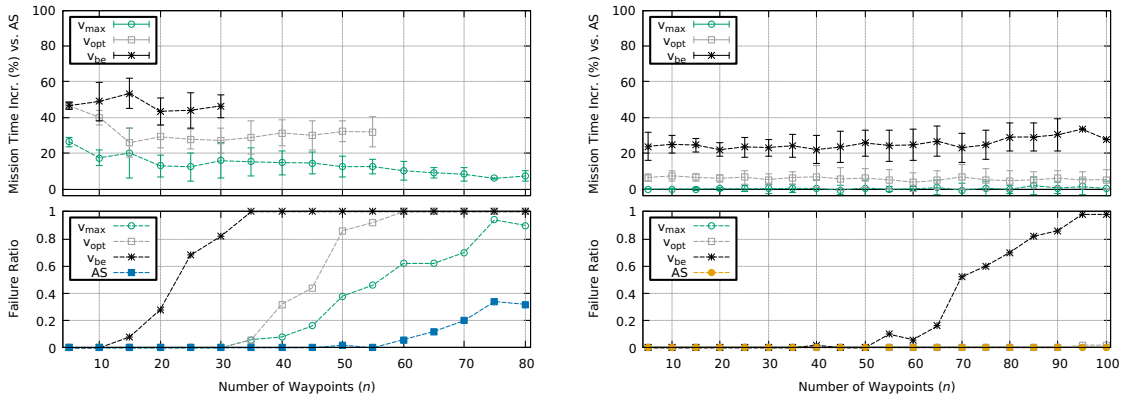
Figure 10 shows the results of these two test scenarios with Figure 10a showing the near-start results and Figure 10b showing the far-start results. On the near-start sinusoidal data set, the *k*-TSP struggled to compete against both the *D*-TSP and adaptive TS approaches, with the *D*-TSP improving mission completion time by an average of 7.8% compared to *k*-TSP. The *D*-TSP also improved mission completion time by an average of 7.0% when compared against the TS approach. Similar to the larger linear data set, the *k*-TSP also struggled to find solutions as n grew above 80.

On the far-start sinusoidal data set, the *k*-TSP approach again failed to find a solution for all inputs because of the assumption that the first sub-tour should start at the beginning of the considered time horizon. Both the *D*-TSP and adaptive TS approaches were able to find solutions to all inputs with the *D*-TSP improving mission completion time by an average of 33.3% when compared against the adaptive TS approach.

8.4 Impact of Speed-Scheduling

We also evaluated how our approach to adaptive speed affected mission completion time. We compared both the *k*-IP and *D*-TSP approaches using adaptive speed (AS) against using these same algorithms when the drone's speed is fixed at v_{max}, v_{opt}, and the speed that minimizes energy consumption (termed best endurance, or v_{be}). We chose these different speed settings because they have all been proposed for drone path planning problems in recent literature [38, 41, 55].

Figure 11a shows the results from using the *k*-IP algorithm on the same input set that was used to generate the data shown in Figure 6a. In this data set, the GV moves along a linear trajectory and starts close to the waypoints. On average, the adaptive speed approach improved the mission completion time by 11.9%, 31.9%, and 47.1% compared against fixing the velocity at v_{max}, v_{opt}, and v_{be}, respectively. With speed fixed at v_{max}, v_{opt}, and v_{be} the solver only found solutions for 69.3%, 52.5%, and 25.9% of the graphs, respectively, while using an adaptive speed approach found a solution for 93.4% of the inputs. In fact, when fixing speed at v_{be} no graphs were solved with $n \geq 35$.



(a) Impact of speed scheduling for *k*-IP

(b) Impact of speed scheduling for *D*-TSP

Fig. 11. Simulation results showing the impact of using adaptive speed in path finding approaches. The top plot of (*a*) shows the percent improvement that speed scheduling yields with the *k*-IP approach when compared to fixing the speed while (*b*) shows this impact using the *D*-TSP approach. The bottom plots for both (*a*) and (*b*) show the ratio of graphs where each approach failed to find a solution.

Figure 11b shows the results of running the D-TSP approach on the far-start linear data set. Adaptive speed had negligible impact when compared to fixing speed at v_{max} while when compared to fixing the speed at v_{opt} adaptive speed only improved mission completion time by an average of 5.7% across all inputs. When compared against v_{be}, using adaptive speed improved the results by 25.5%. Using v_{be} again made it hard for our solution framework to find solutions as the number of waypoints grew but the D-TSP approach was able to find solutions for all inputs when using speed scheduling and when fixing speed at v_{max} and v_{opt}.

8.5 Computational Efficiency

Table 3 shows the average and 95% confidence interval on computation time for randomized graph inputs on the various approaches. The f-MINLP results are from the data set used for Figure 6a and D-MINLP results come from the data set used for Figure 6b, while the rest are from the near-start linear data set discussed in Section 8.3.2. This data only includes the results from graphs that each approach was able to find a feasible solution on.

The table shows that the computational time for both the D-MINLP and the f-MINLP increases exponentially with the number of waypoints, with the f-MINLP able to solve slightly larger inputs. As expected, the a-TS approach averaged the lowest computation time. Surprisingly, the k-IP approach was the second fastest despite having no guarantee to run in polynomial time, which we credit to the speed of the Gurobi solver. Out of the solutions that are iteratively running the LKH solver, the D-TSP approach was slightly faster than the k-TSP. We believe this is because the D-TSP algorithm converges on a solution faster than the k-TSP does.

Table 3. Average computation time (in *seconds*) and 95% confidence interval (95% CI) for the D-MINLP (D-MP), f-MINLP (f-MP), k-IP, k-TSP, D-TSP and adaptive tour-splitting (a-TS) approaches with varying number of waypoints (n).

	$n=$	5	10	15	20	30	40	50	60	70	80
d-MP	Avg.	0.24	77.1	79.3	**162.9**	-	-	-	-	-	-
	95% CI	0.039	59.7	53.0	127.8	-	-	-	-	-	-
f-MP	Avg.	0.08	0.73	14.4	135	**5139**	-	-	-	-	-
	95% CI	0.01	0.15	7.64	48.34	3215	-	-	-	-	-
k-IP	Avg.	0.02	0.02	0.03	0.05	0.08	0.13	0.19	0.23	0.30	**0.37**
	95% CI	0.002	0.002	0.004	0.006	0.010	0.011	0.017	0.027	0.032	0.033
k-TSP	Avg.	0.02	0.03	0.03	0.04	0.13	0.19	0.40	0.51	0.64	0.73
	95% CI	0.003	0.007	0.005	0.009	0.022	0.024	0.049	0.068	0.072	0.097
D-TSP	Avg.	0.06	0.07	0.08	0.09	0.16	0.18	0.29	0.40	0.45	**0.54**
	95% CI	0.005	0.005	0.007	0.008	0.015	0.018	0.026	0.038	0.047	0.045
a-TS	Avg.	0.003	0.005	0.006	0.008	0.018	0.04	0.05	0.06	0.08	0.12
	95% CI	0.0	0.001	0.002	0.002	0.003	0.005	0.004	0.010	0.015	0.022

8.6 Summary of Findings

Our numerical simulations demonstrate various tradeoffs between our different considered approaches. We will first summarize the performance of the fixed-start algorithms introduced in Section 6 then look at when the fixed-start assumption should and should not be used. We will then summarize our findings on the generalized algorithms introduced in Section 7 and remark on the impact of speed scheduling for drone path finding.

Among the fixed-start solutions to the MT-OTM problem, the f-MINLP outperforms the other approaches in solution quality but at the cost of computation time, taking too long to reasonably solve inputs with 30 or more waypoints. The

k-IP algorithm out performs the k-TSP and the fixed-start TSP-based baseline approach (TS) in both solution quality and in computation time. A downside to both the k-IP and k-TSP approaches is that they occasionally fail to find a solution in scenarios where the fixed-start assumption should otherwise work (a short-fall not as common in the TS approach). In our evaluation, we observed that the k-IP and k-TSP approaches struggle on inputs with sub-regions with a high concentration of waypoints while the rest of the graph is sparse. On these graphs, the clustering algorithm forms a single cluster with too many waypoints while the rest of the clusters are fairly small creating many short sub-tours with one very long tour that cannot be completed in a single flight. Computation time should also be considered when deploying the k-IP algorithm. Although our implementation of the k-IP algorithm has a reasonable computation time (averaging 0.37 seconds for inputs with 80 waypoints), this approach is still solving an instance of the TSP to optimality. On more limited hardware and with less powerful optimization tools, we expect the k-IP solution to struggle and recommend the k-TSP algorithm as an alternative.

Our results show that the fixed-start assumption is valid when the waypoints are located near the GV at the beginning of the considered time horizon. In fact, on smaller inputs where the waypoints were all located close to the GV, the f-MINLP and k-IP algorithms found comparable solutions to the D-MINLP algorithm (an exact method for the general MT-OTM problem) and the k-TSP algorithm averaged solutions that were within 4.5% of the optimal. These results show us that, in scenarios where the waypoints are located close to the GV's starting position, the fixed-start assumption is valid and the k-IP and k-TSP algorithms should be preferred. However, the fixed-start assumption begins to fail for inputs where the waypoints were located much further away from the GV's starting position. In such scenarios, the fixed-start algorithms (k-IP and k-TSP) fail to find valid solutions.

Among the algorithms that remove the fixed-start assumption, we found that the D-TSP provides a nice balance between performance, flexibility, and computation time. The algorithm performed slightly worse than the fixed-start solutions (k-IP and D-TSP) on smaller inputs where the waypoints were all located close to the GV. However, when it is not clear when to launch the drone or when the waypoints are spread across a large area, the D-TSP algorithm's versatility helps it find valid solutions when the fixed-start assumption fails. The D-MINLP performs well as an exact algorithm when the number of waypoints is less than 20 but becomes too slow to solve after this point.

Although case dependent, our numerical simulation results also show that speed scheduling can have a major impact on performance. We found that adapting the drones speed did not perform any different from fixing the speed at the drone's max speed (v_{max}) when the waypoints are located further away from the GV. We believe that this is because the D-TSP algorithm is able to stretch each sub-tour out when the waypoints are far enough away from the GV's starting location, which removes the benefit of speed scheduling. However, on inputs where the waypoints are close to the GV's starting location, speed scheduling within the k-IP algorithm not only improved solution quality but also made it easier for the algorithm to find valid solutions. Among the fixed-speed approaches, fixing the drone's speed at v_{max} was the next best option in this scenario but still struggled to consistently find valid solutions at the same rate as our variable-speed approach. We also found that moving at the speed that minimizes energy consumption (v_{be}) performs particularly poor in both near-start and far-start scenarios. We hypothesize that on more structured path finding problems, such as the VRP where the base station is static and vehicles must swap out batteries, speed scheduling will improve performance in the same way that it improved the performance of the k-IP approach and recommend incorporating it into path finding algorithms when planning autonomous behavior for drones.

9 Problem Scenario Prototypes

In this section we demonstrate various prototype scenarios to demonstrate how the MT-OTM Problem could be applied in different environments and on physical hardware. We first give an example of the MT-OTM Problem in an urban environment. We then discuss our field test on a physical drone to show the practicality of our considered problem scenario.

9.1 Urban Environment Simulation

To demonstrate how our MT-OTM solution framework can be used in real-world scenarios, we apply it to an example in an urban environment where the drone must visit a set of waypoints in a city while the GV follows a set route on city streets, i.e., not a straight line as in the previous simulation settings. We selected 20 waypoints for the drone to visit in the urban environment. We selected these points using Google Earth, then converted the GPS coordinates into relative distances and generated a graph for our MT-OTM solution framework to solve. The GV follows a series of city streets, moving at a constant 3 $\frac{m}{s}$.

Figure 12 shows the drone path generated using the f-MINLP approach. It took the f-MINLP approach 43.1 seconds to find this solution. We argue that this demonstrates that for most common scenarios the number of waypoints will be small enough that the f-MINLP approach can be used to find superior solutions. The solution found using the f-MINLP has a mission completion time of 355.8 seconds while the k-IP finds a solution with 373.1 seconds.

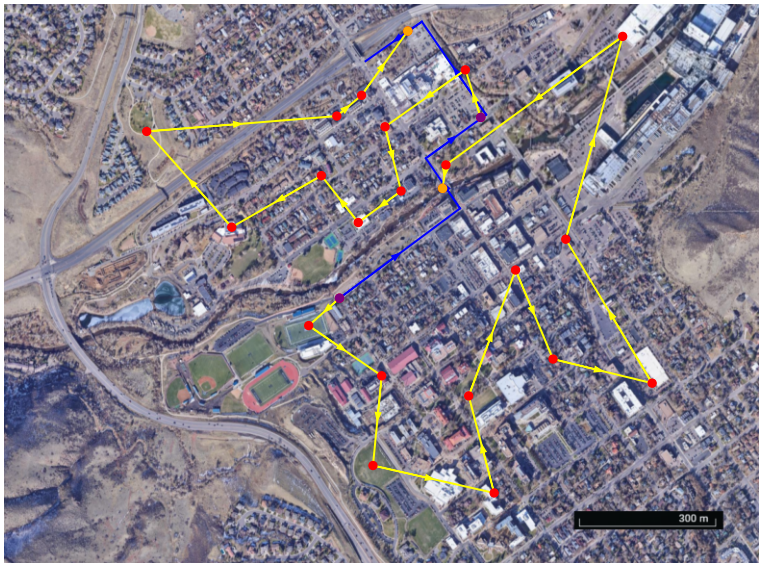
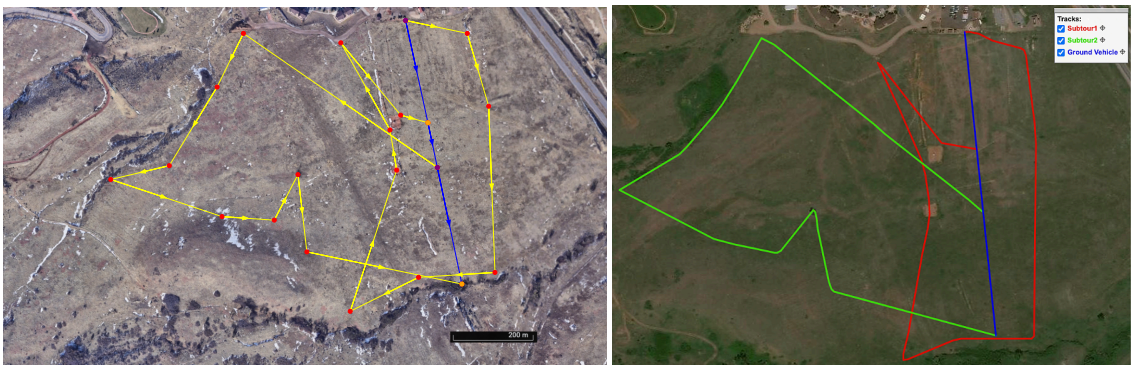


Fig. 12. Drone path generated using f-MINLP approach for a case study in an urban environment. The red circles are drone waypoints, the purple circles are drone launch locations, and the orange circles are drone landing points. The yellow line shows the path of the drone and the blue line shows the path of the GV.



(a) Drone paths generated using k-IP (b) GPS trace of drone path from deployment

Fig. 13. Field prototype results. Image (a) shows the drone path generated using k-IP approach and image (b) drone's GPS trace from our field experiment.

9.2 Field Test

To further validate our solution framework we created a field prototype of the MT-OTM Problem using our own physical drone testbed [15]. We selected 18 waypoints in an empty field using Google Earth with the GV moving in a straight line across the field at 2.5 $\frac{m}{s}$. For simplicity, we landed the drone manually at the end of each sub-tour and substituted the GV by walking the path of the vehicle on foot. Because the drones in [15] can travel up to 8 km, we shortened the max flying distance to 1.7 km and set a max velocity of 11 $\frac{m}{s}$ to scale down the physical prototype.

Figure 13a shows the drone path found using the k-IP approach. Figure 13b shows the GPS trace of the drone while following the found paths. The result on the physical prototype demonstrates that our approach works well when the MT-OTM problem is applied in real world scenarios.

10 Conclusions and Discussion

In this work, we formulated the Minimum-Time while On-The-Move (MT-OTM) Problem and presented several algorithms that solve this problem. This work builds on our previous conference proceedings [16].

We first looked at how to solve the MT-OTM Problem when assuming that the first drone sub-tour must start at the beginning of the considered time window and that all consecutive sub-tours must start as early as possible. This assumption allowed us to simplify the problem into an underlying fixed multi-depot, multi-terminal Hamiltonian paths problem (fixed-MdMtHPP). We developed two approaches for solving fixed-MdMtHPP, a MINLP (termed f-MINLP) that optimizes drone speeds and a k-means clustering algorithm paired with a TSP solver (the k-IP and k-TSP approaches).

We then looked at how to solve the MT-OTM Problem with the fixed-start assumption removed. We modified our f-MINLP formulation so that it selects the start and end location of each sub-tour (the D-MINLP solution), which provides an admissible algorithm to the MT-OTM Problem. We also provided a dynamic version of our k-TSP approach (the D-TSP approach) that dynamically sets sub-tour start and end locations and runs in polynomial time.

The main findings from this larger body of work are:

(1) We presented a framework for drone path finding that allows a drone to be launched from a GV moving along a fixed trajectory.

(2) Through numerical simulations, we found that the *D*-MINLP solution can solve smaller problem inputs to optimality but begins to struggle as the number of waypoints grows.
(3) We showed that our *k*-IP approach is ideal when the user knows when to start the time horizon but is not as flexible for more general inputs.
(4) We found that the *D*-TSP approach at times does not perform as well as the *k*-IP approach, but is much more flexible than the *k*-IP and was able to find a solution to a more diverse range of inputs.
(5) We discovered a tradeoff between maximizing drone speed and maximizing the distance that a drone can travel.
(6) We showed that drone speed adaptation can reduce mission completion time in well-structured problems and should be considered when planning paths for drones.

We believe that the tradeoff between speed and travel distance, leading to our speed scheduling approach within our algorithms, should be of particular interest to the research community and argue that our results support this position. Speed scheduling not only led to better quality solutions, but also allowed our algorithm to find solutions on inputs where following a fixed-speed approach failed to find a solution. Beyond speed scheduling, we also believe that our proposed solution framework provides a template for stakeholders to build off of when implementing planning algorithms for autonomous drones.

One possible application of this research is maritime search and rescue, where drones can be quickly launched from large ships to search for survivors after a disaster [30]. Another possible application is to further integrate our problem setup into a smart city environment [51]. The drone and GV could utilize static infrastructure to aide with communication [56], navigation [7], and battery charging [46]. In a smart city scenario, the drone could be delivering packages, monitoring traffic conditions, or aiding first responders [23] while the GV performs some other task within the city but allows the drone to ride hitch [48].

For future work, we plan to expand the MT-OTM problem to include multiple drones and multiple GVs. Variations of this multi-vehicle extension could include limiting the number of drones that can be at a GV, using the GV to ferry around drones, and considering repeated patrolling scenarios over long time horizons. We made the assumption that the drone does not need to stop and hover at each waypoint to collect data in this work (e.g. taking images while passing through a waypoint). Future work could expand on this application setup by requiring the drone to hover at each waypoint for some predetermined time period. Other expansions to the current work include addressing communication delays and online coordination between the drone and the GV, further investigating how to apply our general framework for package delivery applications, and application-specific integration of the drone and GV into smart cities. Possible solutions to these open research questions include reinforcement learning and adaptive algorithms for online decision making.

More work is also needed to further validate this energy profile on a physical testbed and to study the probability distribution associated with our theoretical model. We believe that a probabilistic model for drone energy could be used to further improve the performance and accuracy of drone path finding algorithms by combining our speed scheduling approach with works such as [55] and [4] where they consider the risk of the drone running out of energy before completing its assigned task. Future work should also investigate if the tradeoff between maximizing speed and maximizing travel distance exists for other autonomous transportation systems, such as GVs, and determine how speed scheduling for these vehicle types impacts path finding.

Declarations

This work was funded in part by the Army Research Laboratory DCIST W911NF-17-2-0181.

References

[1] Mohamed Abdelkader, Samet Güler, Hassan Jaleel, and Jeff S Shamma. 2021. Aerial swarms: Recent applications and challenges. *Current robotics reports* 2 (2021), 309–320.

[2] Bander Alzahrani, Omar Sami Oubbati, Ahmed Barnawi, Mohammed Atiquzzaman, and Daniyal Alghazzawi. 2020. UAV assistance paradigm: State-of-the-art in applications and challenges. *Journal of Network and Computer Applications* 166 (2020), 102706.

[3] Divansh Arora, Parikshit Maini, Pedro Pinacho-Davidson, and Christian Blum. 2019. Route Planning for Cooperative Air-Ground Robots with Fuel Constraints: An Approach Based on CMSA. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 207–214. https://doi.org/10.1145/3321707.3321820

[4] Ahmad Bilal Asghar, Guangyao Shi, Nare Karapetyan, James Humann, Jean-Paul Reddinger, James Dotterweich, and Pratap Tokekar. 2023. Risk-aware recharging rendezvous for a collaborative team of uavs and ugvs. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 5544–5550.

[5] Tomas Baca, Petr Stepan, Vojtech Spurny, Daniel Hert, Robert Penicka, Martin Saska, Justin Thomas, Giuseppe Loianno, and Vijay Kumar. 2019. Autonomous landing on a moving vehicle with an unmanned aerial vehicle. *Journal of Field Robotics* 36, 5 (2019), 874–891.

[6] Jungyun Bae and Sivakumar Rathinam. 2012. Approximation algorithms for multiple terminal, hamiltonian path problems. *Optimization Letters* 6, 1 (2012), 69–85.

[7] Ahmed Bahabry, Xiangpeng Wan, Hakim Ghazzai, Gregg Vesonder, and Yehia Massoud. 2019. Collision-free navigation and efficient scheduling for fleet of multi-rotor drones in smart city. In *62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 552–555.

[8] Tolga Bektas. 2006. The multiple traveling salesman problem: an overview of formulations and solution procedures. *omega* 34, 3 (2006), 209–219.

[9] Tarek Bouzid, Noureddine Chaib, Mohamed Lahcen Bensaad, and Omar Sami Oubbati. 2023. 5G network slicing with unmanned aerial vehicles: Taxonomy, survey, and future directions. *Transactions on Emerging Telecommunications Technologies* 34, 3 (2023), e4721.

[10] Nils Boysen, Stefan Fedtke, and Stefan Schwerdfeger. 2021. Last-mile delivery concepts: a survey from an operational research perspective. *Or Spectrum* 43 (2021), 1–58.

[11] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. 2016. The vehicle routing problem: State of the art classification and review. *Computers & industrial engineering* 99 (2016), 300–313.

[12] Shushman Choudhury, Kiril Solovey, Mykel Kochenderfer, and Marco Pavone. 2022. Coordinated Multi-Agent Pathfinding for Drones and Trucks over Road Networks. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 272–280.

[13] Kenny Chour, Jean-Paul Reddinger, James Dotterweich, Marshal Childers, James Humann, Sivakumar Rathinam, and Swaroop Darbha. 2023. An agent-based modeling framework for the multi-UAV rendezvous recharging problem. *Robotics and Autonomous Systems* 166 (2023), 104442.

[14] Jonathan Diller, Peter Hall, and Qi Han. 2023. Holistic path planning for multi-drone data collection. In *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. IEEE, 222–226.

[15] Jonathan Diller, Peter Hall, Corey Schanker, Kristen Ung, Philip Belous, Peter Russell, and Qi Han. 2022. ICCSwarm: A Framework for Integrated Communication and Control in UAV Swarms. In *Proceedings of the Eighth Workshop on Micro Aerial Vehicle Networks, Systems, and Applications* (Portland, OR, USA) *(DroNet '22)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3539493.3539579

[16] Jonathan Diller and Qi Han. 2023. Energy-aware UAV Path Planning with Adaptive Speed. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 923–931.

[17] Jonathan Diller, Qi Han, Robert Byers, James Dotterweich, and James Humann. 2025. Hitchhiker's Guide to Patrolling: Path-Finding for Energy-Sharing Drone-UGV Teams. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[18] Jinyu Fu, Guanghui Sun, Jianxing Liu, Weiran Yao, and Ligang Wu. 2023. On hierarchical multi-UAV dubins traveling salesman problem paths in a complex obstacle environment. *Transactions on Cybernetics* 54, 1 (2023), 123–135.

[19] Keld Helsgaun. 2000. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European journal of operational research* 126, 1 (2000), 106–130.

[20] Keld Helsgaun. 2017. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. *Roskilde* (2017), 24–50.

[21] Imdat Kara, Gilbert Laporte, and Tolga Bektas. 2004. A note on the lifted Miller–Tucker–Zemlin subtour elimination constraints for the capacitated vehicle routing problem. *European Journal of Operational Research* 158, 3 (2004), 793–795.

[22] Richard M. Karp. 1972. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*. 85–103.

[23] Muhammad Asghar Khan, Bilal Ahmed Alvi, Alamgir Safi, and Inam Ullah Khan. 2018. Drones for good in smart cities: A review. In *Proc. Int. Conf. Elect., Electron., Comput., Commun., Mech. Comput.(EECCMC)*. 1–6.

[24] Patchara Kitjacharoenchai, Byung-Cheol Min, and Seokcheon Lee. 2020. Two echelon vehicle routing problem with drones in last mile delivery. *International Journal of Production Economics* 225 (2020), 107598. https://doi.org/10.1016/j.ijpe.2019.107598

[25] Patchara Kitjacharoenchai, Mario Ventresca, Mohammad Moshref-Javadi, Seokcheon Lee, Jose MA Tanchoco, and Patrick A Brunese. 2019. Multiple traveling salesman problem with drones: Mathematical model and heuristic approach. *Computers & Industrial Engineering* 129 (2019), 14–30.

[26] Songhua Li, Minming Li, Lingjie Duan, and Victor Lee. 2021. Online ride-hitching in UAV travelling. In *International Computing and Combinatorics Conference*. Springer, 565–576.

[27] Chi Harold Liu, Chengzhe Piao, and Jian Tang. 2020. Energy-Efficient UAV Crowdsensing with Multiple Charging Stations by Deep Learning. In *IEEE Conference on Computer Communications*. 199–208. https://doi.org/10.1109/INFOCOM41043.2020.9155535

[28] Zhilong Liu, Raja Sengupta, and Alex Kurzhanskiy. 2017. A power consumption model for multi-rotor small unmanned aircraft systems. In *International Conference on Unmanned Aircraft Systems (ICUAS)*. 310–315. https://doi.org/10.1109/ICUAS.2017.7991310

[29] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[30] Vincenzo Lomonaco, Angelo Trotta, Marta Ziosi, Juan de Dios Yáñez Ávila, and Natalia Díaz-Rodríguez. 2018. Intelligent Drone Swarm for Search and Rescue Operations at Sea. In *Workshop on AI for Good, NeurIPS 2018 (Neural Information Processing Systems)*.

[31] Zhihao Luo, Mark Poon, Zhenzhen Zhang, Zhong Liu, and Andrew Lim. 2021. The Multi-visit Traveling Salesman Problem with Multi-Drones. *Transportation Research Part C: Emerging Technologies* 128 (2021), 103172. https://doi.org/10.1016/j.trc.2021.103172

[32] Parikshit Maini and P. B. Sujit. 2015. On cooperation between a fuel constrained UAV and a refueling UGV for large scale mapping applications. In *International Conference on Unmanned Aircraft Systems (ICUAS)*. 1370–1377. https://doi.org/10.1109/ICUAS.2015.7152432

[33] Parikshit Maini, Kaarthik Sundar, Mandeep Singh, Sivakumar Rathinam, and PB Sujit. 2019. Cooperative aerial–ground vehicle route planning with fuel constraints for coverage applications. *Transactions on Aerospace and Electronic Systems* 55, 6 (2019), 3016–3028.

[34] M Amine Masmoudi, Simona Mancini, Roberto Baldacci, and Yong-Hong Kuo. 2022. Vehicle routing problems with drones equipped with multi-package payload compartments. *Transportation Research Part E: Logistics and Transportation Review* 164 (2022), 102757.

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

[36] Neil Mathew, Stephen L Smith, and Steven L Waslander. 2015. Planning paths for package delivery in heterogeneous multirobot teams. *Transactions on Automation Science and Engineering* 12, 4 (2015), 1298–1308.

[37] Clair E Miller, Albert W Tucker, and Richard A Zemlin. 1960. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)* 7, 4 (1960), 326–329.

[38] Chase C Murray and Ritwik Raj. 2020. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies* 110 (2020), 368–398.

[39] Charles E Noon and James C Bean. 1993. An efficient transformation of the generalized traveling salesman problem. *INFOR: Information Systems and Operational Research* 31, 1 (1993), 39–44.

[40] Sara Pérez-Carabaza, Jürgen Scherer, Bernhard Rinner, José A López-Orozco, and Eva Besada-Portas. 2019. UAV trajectory optimization for Minimum Time Search with communication constraints and collision avoidance. *Engineering Applications of Artificial Intelligence* 85 (2019), 357–371.

[41] Stefan Poikonen and Bruce Golden. 2020. Multi-visit drone routing problem. *Computers & Operations Research* 113 (2020), 104802.

[42] Ritwik Raj and Chase Murray. 2020. The multiple flying sidekicks traveling salesman problem with variable drone speeds. *Transportation Research Part C: Emerging Technologies* 120 (2020), 102813.

[43] Subramanian Ramasamy, Jean-Paul F Reddinger, James M Dotterweich, Marshal A Childers, and Pranav A Bhounsule. 2021. Cooperative route planning of multiple fuel-constrained Unmanned Aerial Vehicles with recharging on an Unmanned Ground Vehicle. In *International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 155–164.

[44] Subramanian Ramasamy, Jean-Paul F Reddinger, James M Dotterweich, Marshal A Childers, and Pranav A Bhounsule. 2022. Coordinated route planning of multiple fuel-constrained unmanned aerial systems with recharging on an unmanned ground vehicle for mission coverage. *Journal of Intelligent & Robotic Systems* 106, 1 (2022), 30.

[45] Ronald L Rardin. 1998. *Optimization in operations research*. Prentice Hall, Upper Saddle River, NJ.

[46] Roberto G Ribeiro, Luciano P Cota, Thiago AM Euzébio, Jaime A Ramírez, and Frederico G Guimarães. 2021. Unmanned-aerial-vehicle routing problem with mobile charging stations for assisting search and rescue missions in postdisaster scenarios. *Transactions on Systems, Man, and Cybernetics: Systems* 52, 11 (2021), 6682–6696.

[47] Baiq Zuyyina Hilyatur Rozaliya, I-Lin Wang, and Ahmad Muklason. 2022. Multi-UAV routing for maximum surveillance data collection with idleness and latency constraints. *Procedia Computer Science* 197 (2022), 264–272.

[48] Lihua Ruan, Lingjie Duan, and Jianwei Huang. 2021. Optimal uav hitching on ground vehicles. In *Global Communications Conference (GLOBECOM)*. IEEE, 01–06.

[49] Mohamed R Salama and Sharan Srinivas. 2022. Collaborative truck multi-drone routing and scheduling problem: Package delivery with flexible launch and recovery sites. *Transportation Research Part E: Logistics and Transportation Review* 164 (2022), 102788.

[50] Raïssa G Mbiadou Saleu, Laurent Deroussi, Dominique Feillet, Nathalie Grangeon, and Alain Quilliot. 2022. The parallel drone scheduling problem with multiple drones and vehicles. *European Journal of Operational Research* 300, 2 (2022), 571–589.

[51] Ruben Sánchez-Corcuera, Adrián Nuñez-Marcos, Jesus Sesma-Solance, Aritz Bilbao-Jayo, Rubén Mulero, Unai Zulaika, Gorka Azkune, and Aitor Almeida. 2019. Smart cities survey: Technologies, application domains and challenges for the cities of the future. *International Journal of Distributed Sensor Networks* 15, 6 (2019), 1550147719853984.

[52] Jürgen Scherer and Bernhard Rinner. 2017. Short and full horizon motion planning for persistent multi-UAV surveillance with energy and communication constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 230–235. https://doi.org/10.1109/IROS.2017.8202162

[53] Jürgen Scherer and Bernhard Rinner. 2020. Multi-UAV surveillance with minimum information idleness and latency constraints. *Robotics and Automation Letters* 5, 3 (2020), 4812–4819.

[54] Feng Shan, Junzhou Luo, Runqun Xiong, Wenjia Wu, and Jiashuo Li. 2020. Looking before Crossing: An Optimal Algorithm to Minimize UAV Energy by Speed Scheduling with a Practical Flight Energy Model. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 1758–1767. https://doi.org/10.1109/INFOCOM41043.2020.9155376

[55] Guangyao Shi, Nare Karapetyan, Ahmad Bilal Asghar, Jean-Paul Reddinger, James Dotterweich, James Humann, and Pratap Tokekar. 2022. Risk-aware UAV-UGV Rendezvous with Chance-Constrained Markov Decision Process. In *IEEE 61st Conference on Decision and Control (CDC)*. 180–187. https://doi.org/10.1109/CDC51059.2022.9993358

[56] Weisen Shi, Haibo Zhou, Junling Li, Wenchao Xu, Ning Zhang, and Xuemin Shen. 2018. Drone assisted vehicular networks: Architecture, challenges and opportunities. *IEEE Network* 32, 3 (2018), 130–137.

[57] Georgy Skorobogatov, Cristina Barrado, and Esther Salamí. 2020. Multiple UAV systems: A survey. *Unmanned Systems* 8, 02 (2020), 149–169.

[58] Yuan Tang, Rui Zhou, Guibin Sun, Bin Di, and Rongling Xiong. 2020. A novel cooperative path planning for multirobot persistent coverage in complex environments. *Sensors Journal* 20, 8 (2020), 4485–4495.

[59] Neelanga Thelasingha, A Agung Julius, James Humann, Jean-Paul Reddinger, James Dotterweich, and Marshal Childers. 2024. Iterative Planning for Multi-Agent Systems: An Application in Energy-Aware UAV-UGV Cooperative Task Site Assignments. *Transactions on Automation Science and Engineering* (2024).

[60] Pratap Tokekar, Joshua Vander Hook, David Mulla, and Volkan Isler. 2016. Sensor planning for a symbiotic UAV and UGV system for precision agriculture. *Transactions on Robotics* 32, 6 (2016), 1498–1511.

[61] Yong Wang, Zheng Wang, Xiangpei Hu, Guiqin Xue, and Xiangyang Guan. 2022. Truck–drone hybrid routing problem with time-dependent road travel time. *Transportation Research Part C: Emerging Technologies* 144 (2022), 103901.

[62] Zheng Wang and Jiuh-Biing Sheu. 2019. Vehicle routing problem with drones. *Transportation research part B: methodological* 122 (2019), 350–364.

[63] Yinyu Ye and Edison Tse. 1989. An extension of Karmarkar's projective algorithm for convex quadratic programming. *Mathematical programming* 44 (1989), 157–179.

[64] Kevin Yu, Ashish Kumar Budhiraja, Spencer Buebel, and Pratap Tokekar. 2019. Algorithms and experiments on routing of unmanned aerial vehicles with mobile recharging stations. *Journal of Field Robotics* 36, 3 (2019), 602–616.

[65] Kevin Yu, Ashish Kumar Budhiraja, and Pratap Tokekar. 2018. Algorithms for routing of unmanned aerial vehicles with mobile recharging stations. In *international conference on robotics and automation (ICRA)*. IEEE, 5720–5725.

[66] Yong Zeng, Jie Xu, and Rui Zhang. 2019. Energy minimization for wireless communication with rotary-wing UAV. *IEEE Transactions on Wireless Communications* 18, 4 (2019), 2329–2345.

[67] Juan Zhang, James F Campbell, Donald C Sweeney II, and Andrea C Hupman. 2021. Energy consumption models for delivery drones: A comparison and assessment. *Transportation Research Part D: Transport and Environment* 90 (2021), 102668.

[68] Yufei Zhang, Zhong Wu, and Tong Wei. 2024. Precise Landing on Moving Platform for Quadrotor UAV via Extended Disturbance Observer. *Transactions on Intelligent Vehicles* (2024).