Multi-robot Rendezvous Planning for Recharging in Persistent Tasks

Neil Mathew Stephen L. Smith *Member, IEEE,* Steven L. Waslander *Member, IEEE*

***Abstract*—This paper addresses a multi-robot scheduling problem in which autonomous unmanned aerial vehicles (UAVs) must be recharged during a long term mission. The proposal is to introduce a separate team of dedicated charging robots that the UAVs can dock with in order to recharge. The goal is to schedule and plan minimum cost paths for charging robots such that they rendezvous with and replenish the UAVs, as needed, during the mission. The approach is to discretize the 3D UAV flight trajectories into sets of projected charging points on the ground, thus allowing the problem to be abstracted onto a partitioned graph. Solutions consist of charging robot paths that collectively charge each of the UAVs. The problem is solved by first formulating the rendezvous planning problem to recharge each UAV once using both, an Integer Linear Program and a transformation to the Travelling Salesman Problem. The methods are then leveraged to plan recurring rendezvous' over longer horizons using fixed horizon and receding horizon strategies. Simulation results using realistic vehicle and battery models demonstrate the feasibility and robustness of the proposed approach.**

***Index Terms*—Path planning, Multiple mobile robot systems, Scheduling and coordination.**

I. Introduction

COORDINATED teams of autonomous robots are often proposed as a means to continually monitor dynamic environments in applications such as air quality sampling [2], border security [3] or visual inspections of power plants and pipe-lines [4]. Such *persistent surveillance* tasks generally require the robots to continuously traverse the environment in trajectories designed to optimize certain performance criteria such as quality or frequency of sensor measurements taken in the region [5], [6], [7]. In this work we focus on the case of a team of multi-rotor UAVs monitoring an environment, in a scenario such as power-line inspection.

The challenge with using aerial robots in persistent tasks is that mission durations generally exceed the run time of the robots, and in order to maintain continuous operation they must be periodically replenished by recharging stations or automated battery swap systems that have been demonstrated in [8] and [9].

As described by Vaughan et al. in [10], in nonstationary tasks such as surveillance, the location of the docking station has a significant impact on the task performance of the team, since the optimal docking location may vary over the mission.

A preliminary version of this work appeared as [1].

This research is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

N. Mathew and S. L. Waslander are with the Department of Mechanical and Mechatronics Engineering and S. L. Smith is with the Department of Electrical and Computer Engineering, all at the University of Waterloo, Waterloo ON, N2L 3G1 Canada (nmathew@uwaterloo.ca; stephen.smith@uwaterloo.ca; stevenw@uwaterloo.ca)

Thus, we present a cooperative replenishment strategy for a team of *working robots* (UAVs) performing a surveillance task, using one or more mobile *charging robots*. Each charging robot is equipped with a payload of batteries and automated battery swap systems and the goal is to design routes that optimally charge each working robot.

The working robots are not required to modify their three-dimensional surveillance trajectories for rendezvous, so as to minimize hindrances to the mission objectives caused by the recharging schedule. We assume that charging robots possess sufficient energy resources and need not be refuelled themselves within the planning horizon.

A. Related Work

The problem of persistent coverage and surveillance with mobile robots, has been investigated in a variety of scenarios in existing literature, such as a centroidal Voronoi tessellation-based controller for static coverage [5], optimal velocity controllers for surveillance along precomputed paths, [6], and path planning to periodically visit a set of discrete interest points with varying frequencies of observation [11].

Such persistent surveillance tasks by definition will exceed the range capabilities of any inspection robot, and therefore naturally require the inclusion of recharging in their formulations. Derenick et al. [12] propose a modification to [5] that introduces a combined coverage and energy dependent control law to drive each robot toward a fixed docking station as their energy levels become critical. Their work considers only the static coverage case and there is no notion of charge scheduling as each agent is assigned a dedicated static charging station.

Contrary to [12], the notion of mobile charging stations has been studied in literature in the contexts of long term missions performed by UAVs [13], [14], [15], satellites [16] or general robotic agents [17]. In [17], Litus et al. consider the problem of finding a set of meeting points for a set of static working robots and a single charging robot in a Euclidean plane using a discrete and continuous optimization approach. In [13], [14], [15], the authors formulate recharge scheduling with multiple working agent and a single replenishment agent as combinatorial optimization problems and solve them using methods such as integer program formulations, dynamic programming and heuristic search algorithms.

In our work we will consider a heterogeneous UAV-UGV team consisting of a team of (potentially heterogeneous) UAV working robots with varying trajectories and a team of homogeneous UGV recharging robots similar to the multi-robot teams described in [18], [19], [20]. For such heterogeneous multi-robot teams, Rathinam et al. explore optimal

path planning for UAVs using variants of the Travelling Salesman Problem (TSP) and the Generalized Travelling Salesman Problem (GTSP) [21], [22], [23], [24] which are well studied problems in operations research literature and can be solved using a number of exact, approximate or heuristic algorithms.

In contrast to this literature, we define a recharge scheduling problem for a scenario where a team of UAVs is required to persistently monitor an environment in trajectories that are known within a planning horizon. We discretize the UAV trajectories into sets of charging locations along the UAV trajectory where the UAVs can dock with a charging robot, recharge using an automated battery swap system and take-off to return to their respective trajectories. These charging locations along the UAV trajectories correspond to charging point on the ground to which the charging robots must travel in order to execute a recharge procedure, and we seek the optimal selection and ordering of recharge procedures to service the fleet of aerial vehicles.

We cast the problem as a Generalized Travelling Salesman Problem (GTSP) for a single charging robot and a Multiple Generalized Travelling Salesman Problem (MGTSP) for multiple charging robots. We propose solutions based on both integer linear programs and a transformation to the Travelling Salesman Problem. The TSP is a well studied problem with a number of established exact, approximate and heuristic solution algorithms in operations research literature. One of the best known algorithms is the Lin-Kernighan heuristic [25] implemented as the Concorde LinKern TSP solver and an adaptation proposed by Helsgaun [26] implemented as the Lin-Kernighan-Helsgaun (LKH) TSP solver. While these heuristics do not have proven guarantees on sub-optimality, they have been empirically shown to produce solutions within 2% of the optimal [27].

In this work we use the Noon-Bean Transformation [28] to cast the GTSP as an Asymmetric Travelling Salesman Problem (ATSP) and solve it using the LKH solver. In the case of the MGTSP, we present a modification to the Noon-Bean transform to the multiple route computation case, drawing from operations research literature on graph transformations for the MTSP [29].

Another body of research that inspired this work is existing literature on the existence of heterochromatic paths in k-vertex and k-edge coloured graphs [30], [31]. To the best of our knowledge, the work that refers to this problem in the context of coloring resides primarily in the discrete mathematics community, and deals with determining graph properties that ensure the existence of long heterochromatic (or monochromatic) paths in the graph.The work that refers to the problem as a GTSP resides primarily in the operations research community, and focuses more on practical (heuristic) solution techniques. We found the later work more useful for solving the charging problem, and thus attached the name GTSP to our work.

B. Contributions

Our approach, in this work, is to position cooperative recharge scheduling in the space of graph-based optimal path planning problems. We develop our algorithms in two stages, first a rendezvous schedule to recharge each working robot once (*single charge cycle*), and second, an extension of the proposed methods to recurring recharges over indefinite planning horizons (*recurring charge cycles*). A preliminary version of this work appeared in [1]. The contributions of this work are four-fold.

(i) We formulate the cooperative recharge scheduling problem as a Multiple Generalized Travelling Salesman Problem (MGTSP) on a partitioned directed acyclic graph (DAG).
(ii) We present two solutions, first, an optimal Integer Linear Program method and second, a polynomial transformation to the Travelling Salesman Problem (TSP) followed by the application of TSP heuristic solvers in existing literature.
(iii) We explore potential failure modes for an offline optimal path planner, investigate the robustness of the schedule and propose some online strategies to mitigate failures arising from modelling errors, and stochasticity in the environment.
(iv) Based on the developed recharge scheduling framework, we extend the solutions to recurring recharges over longer planning horizons using receding horizon and fixed horizon strategies.

The organization of this paper is as follows. Section II introduces the key definitions and nomenclature that are referred to throughout the paper. Section III formulates the single charge cycle problem as an NP-hard path planning problem on a partitioned directed acylic graph. Section IV and V then present the two solution methods employed to compute paths based on an optimal ILP and TSP transformation respectively. Section VI examines the extension of the proposed algorithms to longer planning horizons. Finally, Section VII-C proposes methods to strengthen the robustness of the plan to stochasticity in the system and environment. Section VII presents simulation results to benchmark the performance of optimal and heuristic solutions.

II. Definitions and Nomenclature

A *graph* G is represented by (V, E, c), where V is the set of *vertices*, E is the set of *edges* and $c : E \to \mathbb{R}$ is a function that assigns a cost to each edge in E. In an undirected graph, each edge $e \in E$ is a set of vertices $\{v_i, v_j\}$. In a directed graph each edge is an ordered pair of vertices (v_i, v_j) and is assigned a direction from v_i to v_j. A *partitioned graph* is a graph G with a partition of its vertex set into R mutually exclusive subsets $(V_1, \ldots, V_R)$ such that $\cup_i V_i = V$.

A *path* in a graph G, is a subgraph denoted by $P = (\{v_1, \ldots, v_{k+1}\}, \{e_1, \ldots, e_k\})$ such that $v_i \neq v_j$ for all $i \neq j$, and $e_i = (v_i, v_{i+1})$ for each $i \in \{1, \ldots, k\}$. The set V_P represents the set of vertices in P and by definition $V_P \subseteq V$. Similarly a *tour* or *cycle* T is a closed path in the graph such that $v_1 = v_{k+1}$. Finally, a *directed acyclic graph* (DAG) is a directed graph in which no subset of edges forms a directed cycle. With this we can define the following key problems.

Problem II.1 (The Hamiltonian Path/Tour Problem [32])**.** Given a graph G, does there exist a path P that visits every vertex in G exactly once. Similarly, the Hamiltonian Tour Problem requires a closed path, T, that satisfies the same properties.

Problem II.2 (Travelling Salesman Problem (TSP) [33])**.** Given a graph G, find a Hamiltonian tour T such that total cost of $\sum_{e\in E_T} c(e)$ is minimized, where E_T is the set of edges in T. A symmetric TSP is computed on an undirected graph. Similarly, an asymmetric TSP is obtained on a directed graph.

Problem II.3 (Generalized Travelling Salesman Problem (GTSP) [28])**.** Given a partitioned graph G, find a tour T, that visits a single vertex in every vertex set exactly once, such that the total cost $\sum_{e\in E_T} c(e)$ of T is minimized, where E_T is the set of edges in T.

Finally, we define the extension of the GTSP to multiple robots.

Problem II.4 (Multiple Generalized Traveling Salesman Problem (MGTSP) [34])**.** Given a partitioned graph G, find a collection of paths which collectively visit each vertex set exactly once, with minimum total cost.

III. The Single Charge Cycle Problem

Given a team of working robots conducting a persistent task, the goal of this section is to compute an optimal schedule and path plan for the team of charging robots to rendezvous with every working robot along its trajectory exactly once.

A. Motion Planning For Charging Robots

Consider an environment, $\mathcal{E} \subset \mathbb{R}^3$, which contains R working robots, denoted by the set $\mathcal{R} = \{1, \dots, R\}$. Each working robot, indexed by $r \in \mathcal{R}$, is described by its motion along an independent known trajectory, $P_r(t) \in \mathcal{E}$ within a planning horizon $t \in [0, T_r]$ determined by its battery depletion model and a recharging time window $[\underline{T}_r, \overline{T}_r] \subseteq [0, T_r]$.

The environment also contains M charging robots, denoted by the set $\mathcal{M} = \{1, \dots, M\}$. The charging robots are constrained to a two-dimensional manifold $\bar{\mathcal{E}} \subset \mathcal{E}$ in the case of ground robots, although the method could easily be extended to aerial recharging vehicles operating throughout the environment. Each charging robot, indexed by $m \in \mathcal{M}$, is described by its initial position $p_m(0) \in \bar{\mathcal{E}}$ and its maximum speed, v. We assume that all charging robots have the same maximum speed. The problem is to find optimal paths for the charging robots, $p_m(t)$ in $\bar{\mathcal{E}}$ (where $|\dot{p}_m(t)| \leq v$) such that for each $r \in \mathcal{R}$, there exists a charging robot $m \in \mathcal{M}$ and a time $t_r \in [\underline{T}_r, \overline{T}_r]$ for which $p_m(t_r) = p_r(t_r)$, where p_r denotes the vertical projection of the point $P_r(t_r)$ onto the ground manifold, $\bar{\mathcal{E}}$.

This constraint states that the team of charging robots must rendezvous at least once with each working robot at a feasible charging point before the working robot is completely discharged. Figure 1 illustrates the problem statement with a team of four working robots following a single path, along with two charging robots.

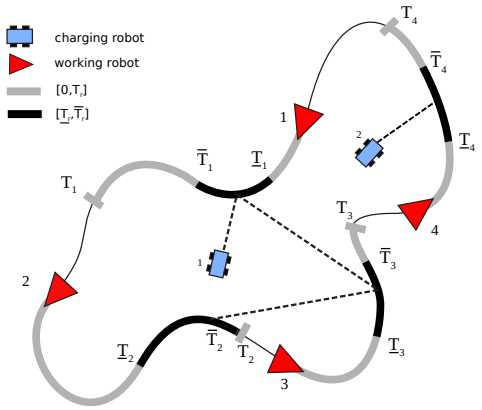


Fig. 1: Four working robots (red triangles) traveling along one path. For each working robot r, $[0, T_r]$ is denoted by a bold grey line and $[\underline{T}_r, \overline{T}_r]$, by a bold black line. The two blue charging robots must meet all working robots on their paths within their charging windows to guarantee persistent operation.

The continuous-time problem, as stated, requires an optimization over the space of all charging robot trajectories [35]. Hence, discretizing the formulation converts the problem into a more tractable form and allows the application of graph-based linear programming techniques to obtain a solution.

B. Discrete Graph Representation

For each working robot r, given that the trajectory $P_r(\cdot)$ is known over the planning horizon, we can discretize its charging time window to generate a set of K_r charging times $\tau_r = \{t_{r,1}, \dots, t_{r,K_r}\} \subseteq [\underline{T}_r, \overline{T}_r]$ at which it can be recharged along its trajectory. The set of charging points on the ground manifold that result are defined as,

$$C_r = \{(p_r(t), t) \,|\, t \in \tau_r\}.$$

Each charging point $(p_r(t_{r,i}), t_{r,i})$ is described by its time of occurrence, $t_{r,i}$, and the projection of the working robots position, $p_r(t_{r,i})$.

A charging robot, subject to its speed constraints, will attempt to charge a working robot by arriving at one of its charging points $p_r(t_{r,i}), t_{r,i}) \in C_r$ at a time $t \leq t_{r,i}$ and staying there until time $t_{r,i}$ such that $p_m(t_{r,i}) = p_r(t_{r,i})$. This definition satisfies the previously stated condition for a rendezvous in continuous time. Note that for the sake of simplicity, the formulation assumes instantaneous charge, but it can be extended directly to the case of nonzero charging durations at each charging point, as discussed in Remark III.1 .

The discrete problem is one of finding paths for the charging robots that visit one charging point in each set C_r. We can encode every possible charging path in a partitioned directed graph G, defined as follows.

a) Vertices: The vertices, are defined by $R + 1$ disjoint vertex sets, $V_0, V_1, \dots, V_R$. The set V_0 is the set of initial locations of the charging robots. Each vertex in set V_r, for $r \in \mathcal{R}$ corresponds to a charging point in C_r, the set of all charging points for robot r. The complete vertex set is then $V = V_0 \cup V_1 \cup \cdots \cup V_R$.

b) Edges: An edge (v_i, v_j) is added to E, where $v_i \in V_{r_1}$ and $v_j \in V_{r_2}$ for some $r_1, r_2 \in \mathcal{R}$ with $r_1 \neq r_2$, to E if there exists a feasible traversable path from charging point $(p_{r_1}(t_{r_1,i}), t_{r_1,i})$ to $(p_{r_2}(t_{r_2,j}), t_{r_2,j})$. That is, if

$$\frac{\|p_{r_2}(t_{r_2,j}) - p_{r_1}(t_{r_1,i})\|}{v} \leq t_{r_2,j} - t_{r_1,i}. \quad (1)$$

c) Edge Costs: Each edge $e = (v_i, v_j) \in E$ is associated with a non-negative cost $c(e)$ that can be chosen based on the objective of the optimization such as minimizing total distance travelled by charging robots or total makespan of the recharging process.

Remark III.1 (Nonzero Charging Durations). For simplicity of presentation we have assumed that charging occurs instantaneously. Thus, if a charging robot performs a rendezvous with a working robot at charging point $(p_r(t_{r,i}), t_{r,i})$, it can leave that charging point at time $t_{r,i}$. We can extend this formulation to charging points described as triples $(p_r(t_{r,i}), t_{r,i}, \Delta t_{r,i})$, where $\Delta t_{r,i}$ is the time required for working robot r to descend to the ground, charge at the i^{th} charging point, and reascend to resume its trajectory. In this case the charging robot can leave the charging point at time $t_{r,i} + \Delta t_{r,i}$. The condition to add an edge in equation 1 then changes to

$$\frac{\|p_{r_2}(t_{r_2,j}) - p_{r_1}(t_{r_1,i})\|}{v} \leq t_{r_2,j} - (t_{r_1,i} + \Delta t_{r_1,i}). \quad (2)$$

Using the autonomous battery swap mechanisms developed in [8] and [9], and a consistent vertical speed for the UAVs, it is possible to deterministically define $\Delta t_{r,i}$ for most environments conducive to surveillance with quadrotor UAVs. The ROS Gazebo simulation that accompanies this paper justifies the use of $\Delta t_{r_1,i}$ in the optimal recharge formulation. •

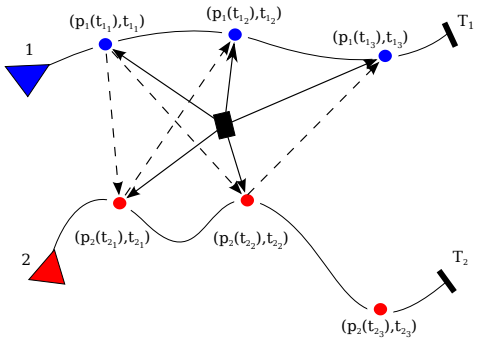
As a simple illustrative example, Figure 2(a) shows two working robots r_1 (blue) and r_2 (red) following arbitrary trajectories and one charging robot m_1 in an environment $\mathcal{E} \subset \mathbb{R}^2$. Each robot path is discretized into three charging points and graph G is constructed on them based on the feasibility conditions.

In addition to the vertex partition, a special property of G is that there are no edges between vertices of the same vertex set, thus making the graph *multipartite* in nature. Further, since the edges represent rendezvous conditions between pairs of time-stamped locations and all edges are directed towards vertices increasing in time, it is impossible for G to contain any directed cycles. Hence, by definition G is partitioned directed acyclic graph (DAG).

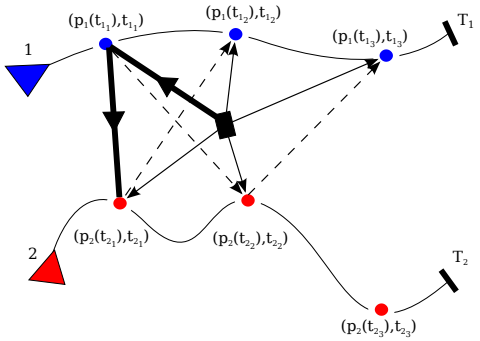
C. Optimization on a Partitioned DAG

Given a partitioned DAG G, the goal is to find an optimal path or set of paths, starting at the m initial locations of charging robots, that collectively visit each vertex set once, as shown in Figure 2(b). To characterize the complexity of our problem, it will be helpful to state this as a decision problem.

Problem III.2 (The One-in-a-set DAG Path Problem)**.** Consider a partitioned DAG G and a partition $(V_0, V_1, \ldots, V_R)$ of V where $V_0 = \{v_m | m \in \mathcal{M}\}$. Does there exist a set of paths $P = \{P_1, \ldots, P_M\}$ in G, where $P_m \in P$ starts at $v_m \in V_0$, such that $|V_i \cap V_P| = 1$ for all $i \in \mathcal{R}$?



(a) Sampled UAV trajectories and roadmap graph for the charging robot.



(b) Optimal Recharge Path Solution

Fig. 2: Building a traversal graph for two working robots and one charging robot. The resulting graph is a directed acyclic graph with vertex partitions.

We will say that the partitioned DAG G contains One-in-a-set path(s) if and only if the answer to the corresponding decision problem is yes.

The One-in-a-set Path problem has been proved to be NP-hard for the case of undirected, complete, or general directed graphs, because they contain, as special cases, the undirected and directed TSP problems, respectively, which are both NP-hard. Unlike these TSP problems, the One-in-a-set DAG Path problem consists of a path through a directed acyclic graph, which is not trivially provable as NP-hard given that the *longest path problem* for directed acyclic graphs is solvable in polynomial time using dynamic programming [36]. However, in the following section we prove that the One-in-a-set DAG Path problem is in fact an NP-hard problem.

Remark III.3 (Maximizing the Number of Robots Charged). For a given number of charging robots and a given discretization of working robot paths, there may not exist recharging paths to charge all robots. In this case, a reasonable approach is to determine a set of charging paths such that the number of working robots that are successfully charged is maximized.

This can be accommodated is follows. For working robot $r \in \mathcal{R}$, we add one extra vertex z_r to the set V_r. A charging robot visits this additional vertex only if the corresponding working robot cannot be charged. We fix $C > 0$ to be larger than the longest edge in the graph and add the following directed edges each with weight C:

(i) edges from each vertex in the original graph to each z_r,
(ii) edges between each pair z_{r_1} and z_{r_2}, where $r_1, r_2 \in \mathcal{R}$

with $r_1 \neq r_2$.

A solution will traverse through the original DAG for as many charges as possible, and will then switch over to the z_r vertices when it cannot charge any more working robots. The remaining working robots will be visited using their vertex z_r, implying that they are not charged and must land safely. Since their are no edges from a z_r back to the original DAG vertices, this modification does not generate any cycles.

Notice that if there exists a feasible solution to the original problem, then no z_r vertices will be visited and the problem is equivalent to Problem III.2. If not all working robots can be charged, then a One-in-a-set DAG solution to this problem will minimize the number of uncharged robots. •

D. Hardness of the Discrete Problem

We will prove NP-hardness of the One-in-a-set DAG Path problem by using a reduction from the NP-Complete Hamiltonian path problem [37].

Theorem III.4 (NP-Completeness of Problem III.2)**.** *The One-in-a-set DAG Path problem is NP-Complete.*

Proof. Consider an instance of the Hamiltonian path problem defined on graph G'. We will give a polynomial transformation of G into a partitioned DAG, G, that is a valid input to the One-in-a-set DAG Path decision problem.

Given the undirected graph G', we need to create a DAG $G = (V, E)$ along with the vertex partition $(V_0, V_1, \ldots, V_R)$. Our approach will be to encode every possible Hamiltonian path order in G'. The One-in-a-set DAG decision problem will then have a yes answer if and only if the graph G' contains a Hamiltonian path.

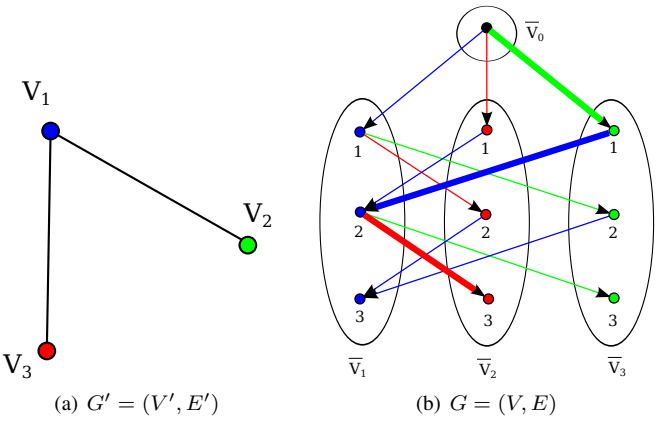
Let $V' = (v_1, \ldots, v_R)$ and for each $r \in \{1, \ldots, R\}$, let V_R be given by R copies of v_r, which we will denote by $V_r := (v_{r,1}, \ldots, v_{r,R})$. The jth copy of v_r will correspond to all paths in G' that have v_r as their jth vertex. Finally, we create a (dummy) vertex V_0 and define $V = V_0 \cup V_1 \cup \cdots \cup V_R$.

Now, we define the edges E as follows. We begin by adding an the edge $(v_0, v_{r,1})$ to E for each $r \in \{1, \ldots, R\}$. Then for any two sets V_i and V_j and for $k \in \{1, \ldots, R-1\}$ we add the edge $(v_{i,k}, v_{j,k+1})$ if and only if $(v_i, v_j) \in E'$. Figure 3 illustrates this reduction and shows that a feasible path is found in the DAG. It is clear that a feasible solution to the described One-in-a-set DAG Path problem yields a feasible solution to the Hamiltonian path problem.

This defines the input G to the One-in-a-set DAG decision problem. It is easy to see that G is acyclic since it has a topological sort: Define the partial ordering as $v_{i,k} \leq v_{j,\ell}$ if and only if $k \leq \ell$ and note that there is an edge from $v_{i,k}$ to $v_{j,\ell}$ only if $\ell = k + 1$. Also, note that G has $R^2 + 1$ vertices.

Finally, we just need to show that G' contains a Hamiltonian path if and only if G contains a One-in-a-set path. Suppose G contains a Hamiltonian path $P = v_{r_1} v_{r_2} \cdots v_{r_R}$, where $(v_{i_j}, v_{i_{j+1}}) \in E$ for each $j \in \{1, \ldots, R-1\}$. Then, the path $\overline{P} = \overline{V}_0, v_{r_1,1} v_{r_2,2} \cdots v_{r_R,R}$ is a One-in-a-set path in $\overline{G}$ since each edge $(v_{r_j,j},\ v_{r_{j+1},j+1})$ is in $\overline{E}$.

Conversely, suppose that G contains a One-in-a-set path P. By the definition of the edges E, the path must be of the form $v_0, v_{r_1,1} v_{r_2,2} \cdots v_{r_R,R}$. This implies that $(v_{r_j}, v_{r_{j+1}}) \in E$ for each $j \in \{1, \ldots, R-1\}$ and thus $P' = v_{r_1} \cdots v_{r_R}$ is a Hamiltonian path in G'. □



(a) $G' = (V', E')$ (b) $G = (V, E)$

Fig. 3: A reduction of the Hamiltonian Path Problem to the One-in-a-set DAG Problem. Each color in graph G' represents an individual vertex. Each vertex color in graph G' corresponds to a unique vertex set in graph G

NP-Completeness of the One-in-a-set DAG decision problem implies that our recharging optimization problem is NP-hard. In what follows we present our approach to the problem from the bottom up. We first formulate the ILP for the single charging robot case and use it to characterize the structure of the optimization. We then extend the problem to include multiple charging robots and investigate algorithmic alternatives to generate near optimal solutions.

IV. Solution 1: Integer Linear Programming

The One-in-a-set DAG Path problem can be stated as an integer linear program (ILP) and optimally solved for smaller instances of the problem. For ease of presentation we first formulate the ILP for a single charging robot path in a partitioned DAG.

Given the partitioned graph G with vertex sets $(V_0, V_1, \ldots, V_R)$, we define a decision variable, $x_{ij} \in \{0, 1\}$ with $x_{ij} = 1$ if, in the resulting path, a visit to vertex v_i is followed by a visit to vertex v_j, where $i \in V_{r_1}$, $j \in V_{r_2}$ and $r_1 \neq r_2$, $r_1, r_2 \in \mathcal{R}$. The cost of the edge traversal x_{ij} is denoted by c_{ij}, and is defined as follows. For the edge $e = (v_i, v_j)$ (with associated decision variable x_{ij}) we define.

$$c_{ij} = \begin{cases} c(e), & \text{if } e \in E, \\ \infty, & \text{if } e \notin E. \end{cases} \tag{3}$$

The start vertex is denoted by index d. The solution path must end at the dummy vertex denoted with index f. The single charging robot ILP is now defined as follows.

$$\min \quad \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{4}$$

subject to

$$\sum_{j \in V \setminus V_0} x_{dj} = 1 \tag{5}$$

$$\sum_{i \in V \setminus V_0} x_{if} = 1 \tag{6}$$

$$\sum_{j \in V_r} \sum_{i \in V} x_{ij} = 1 \qquad \forall r \in \mathcal{R} \tag{7}$$

$$\sum_{i \in V_r} \sum_{j \in V} x_{ij} = 1 \qquad \forall r \in \mathcal{R} \tag{8}$$

$$\sum_{i,j \in V} (x_{ik} - x_{kj}) = 0 \qquad \forall k \in V \setminus V_0 \tag{9}$$

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in V \tag{10}$$

The objective function (4) seeks to minimize the total path cost defined as the travel distance of the charging robot. Constraint (5) and (6) guarantee that the tour starts at the start vertex and ends at the finish vertex. Constraint (7) and (8) ensure that each vertex set is visited only once. Constraint (9) is a flow constraint to guarantee that the entering and exiting edge for each vertex set are both incident on the same vertex in the group. Finally, Constraint (10) specifies binary constraints on the decision variables x_{ij}.

The total number of constraints in this single charging robot formulation is $(2+2R+N)$, where $N = |V|$. The maximum number of binary decision variables on the edges of a complete graph is $N(N-1)$. However, given the multipartite graph G, a decision variable x_{ij} exists only if v_i and v_j belong to different vertex sets.

Note that the number of variables and constraints grows with the number of vertices in the graph G. For a given environment and configuration of working and charging robots, the size of G is determined by the length of the charging window $[\underline{T}_r, \overline{T}_r] \subseteq [0, T_r]$, and the density of charging locations along each robot path.

A. ILP Properties

The optimal solution to the ILP provides a minimum cost path that passes through each vertex set of a DAG exactly once. We observe from the formulation that the problem can be modelled as the Generalized Travelling Salesman Problem (GTSP) [28] as stated in Problem II.3.

Despite structural similarities to the GTSP, the ILP formulated for the One-in-a-set DAG Path problem introduces a significantly smaller constraint set than TSP and GTSP routing problems. This is due to the directed acyclic graph structure. In particular, the TSP requires subtour elimination constraints to ensure that the resulting tour does not contain disjoint subtours in the solution. The most efficient way to eliminate subtours is to introduce an additional N variables u_i, and $O(N^2)$ constraints [38] of the form $u_i - u_j + N x_{ij} \leq N-1$ for all $1 \leq i \neq j \leq N$, where x_{ij} is the decision variable on the edge $(v_i, v_j) \in E$.

The lack of directed cycles in a DAG eliminates the need for sub-tour elimination constraints in our formulation. Further, the multipartite nature of the graph reduces the number of binary decision variables in the integer program. Due to this reduction in the number of variables and constraints relative to a TSP mixed-integer program, we can solve larger problems with relatively lesser computational effort. In practice, we observed that problems with an order of magnitude increase in the number of vertices could be solved in comparable time.

Nevertheless, given the NP-hardness of the problem, optimally solving the ILP will not remain computationally tractable with increasing problem complexity and Section V describes a heuristic approach to compute near-optimal solutions.

B. ILP Formulation for Multiple Charging Robots

The integer linear program in Section IV can be easily extended to the multiple charging robot problem, using a three-index flow formulation. We highlight the differences here and refer a reader to [1] for more details.

In the extended formulation each charging robot m is represented by an independent route p_m and hence the binary decision variables on edges are defined as $x_{ijm} \in \{0,1\}$ with $x_{ijm} = 1$ if, in route p_m, the vertex v_j is visited after vertex v_i, where $i \in V_{r_1}, j \in V_{r_2}, r_1 \neq r_2$ and $r_1, r_2 \in \mathcal{R}$. The new objective function is

$$\min \quad \sum_{m=1}^{M} \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ijm}.$$

Notice that this objective seeks to minimize the total path cost of all charging vehicles. The number of constraints in this formulation is $(2M + 2R + N)$ and the number of decision variables is upper bounded by $MN(N-1)$.

V. Solution 2: Graph Transformations

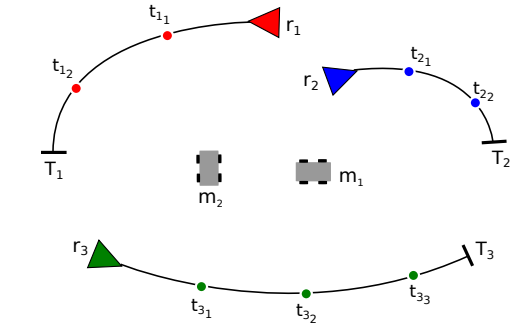
In this section we present an alternative to the ILP formulation, which leverages the high quality heuristic solvers available for the Traveling Salesman Problem. In Section V-A, we implement a modification to the Noon-Bean transformation to transform the multiple charging robot problem into an instance of the TSP that is consistent with similar approaches in optimal path planning literature [21].

Remark V.1 (Noon-Bean Transformation). The Noon-Bean transformation [28] is a graph transformation algorithm used to convert an instance of a GTSP, defined on graph G^0 into a TSP problem instance on graph G^1 such that the optimal TSP solution to the transformed problem will always correspond to the optimal solution to the GTSP. •

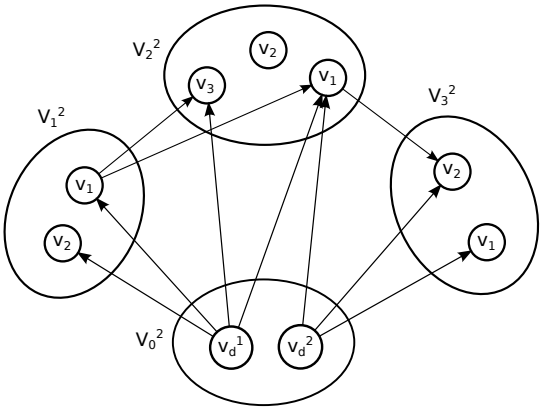
Remark V.2 (Runtime of Reduction to TSP). Given a GTSP instance consisting of N vertices divided into R mutually exclusive sets, the Noon-Bean transformation generates a TSP on N vertices in $O(N^2)$ time [28]. To solve the resulting TSP instance, one can use the Lin-Kernighan heuristic, which is empirically found to generate solutions in $O(N^{2.2})$ time [39]. The resulting GTSP solution can then be reconstructed in $O(N)$ time. •

A. Multi-robot Noon-Bean Transformation

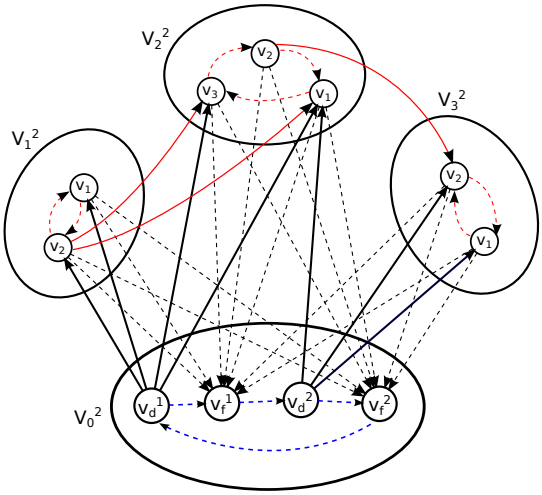
In this section we describe an extension to the Noon-Bean method to transform the MGTSP into a TSP. The



(a) A sample problem consisting of three working robots and two charging robots.



(b) Problem (P2) corresponding to the scenario in Figure 4(a).



(c) Problem (P3), generated using the modified Noon-Bean algorithm. Red edges represent the Noon-Bean transformation and blue edges represent new additions in the modified algorithm.

Fig. 4: The modified transformation for the scenario in Figure 4(a) using both charging robots m_1 and m_2.

tranformation, inspired by a similar approach by Rathinam et al. [21], ensures that the optimal solution to the TSP can be used to construct the optimal solution to the MGTSP. The algorithm is implemented to solve the One-in-a-set DAG path problem for multiple charging robots.

We begin by stating the One-in-a-set DAG Path problem as an MGTSP and call this Problem (P2). See Figure 4(b) as an example.

Problem (P2) is an instance of an MGTSP, defined over a partitioned DAG, G^2, with a partition of its vertices V^2 into $R+1$ sets, $(V_0^2, V_1^2, \ldots, V_R^2)$. The vertex set V_0^2 contains M start-depots for charging robots. We seek a set of paths starting at the depots $v_d^i, i \in \mathcal{M}$ that visit all the vertex sets $V_1^2, \ldots, V_R^2$ exactly once.

The transformation converts the MGTSP problem (P2), into a new problem (P3) on which a TSP solution may be computed. Problem (P3) is a TSP defined over a graph G^3. The vertices, V^3, edges E^3 and cost function c^3 are defined as follows.

(i) Define the set of vertices of G^3, as $V^3 = V^2$. In set V_0^3, add M vertices, $v_f^i, i \in \mathcal{M}$, as the charging robot route finish-depots. At each vertex v_f^i, add edges (v_j, v_f^i), where $v_j \in V^2 \setminus V_0^2$ and assign costs based on the optimization objective.

(ii) In vertex set V_0^3, arrange all start-finish depot pairs (v_d^i, v_f^i) in an arbitrary sequential ordering to obtain $V_0^3 = \{v_d^1, v_f^1, v_d^2, v_f^2, \ldots, v_d^M, v_f^M\}$. Create zero-cost intraset edges forming a single directed cycle through all vertices in V_0^3, in the chosen order. Hence, create edges $(v_d^1, v_f^1), (v_f^1, v_d^2), \ldots, (v_d^M, v_f^M), (v_f^M, v_d^1)$.

(iii) For the definition of all edges (v_i, v_j) where $v_i, v_j \in V^3 \setminus V_0^3$ and $i \neq j$, use the original Noon-Bean method presented in [28].

(iv) Add the penalty $\beta > \sum_{e \in E^3} c_{ij}$ to all edges (v_i, v_j) where $v_i, v_j \in V^3 \setminus V_0^3$ and $i \neq j$. Further, add penalty β to all outgoing interset edges from start-depots v_d^i in set V_0^3. Penalty β is not added to any edges incident on finish-depot vertices in V_0^3.

Figure 4(c) illustrates the transformed graph G^3, for Problem (P3). The transformed graph G^3 defined in Problem (P3) can now be used to compute the TSP solution using a variety of freely and commercially available TSP solvers. The experimental simulations in this work use the LKH solver based on the Lin-Kernighan Helsgaun heuristic to solve TSP instances.

B. Correctness of the Multi-Robot Transformation

We will begin by stating the main result, which parallels that of the original Noon-Bean transformation.

Theorem V.3 (Multi-robot Noon-Bean Theorem)**.** *Given a MGTSP in the form of Problem (P2) with R vertex sets and M depots, we can transform the problem into a TSP in the form of Problem (P3). Given a solution Υ^3 to Problem (P3), we can construct a corresponding solution Υ^2 to Problem (P2) if $\sum_{e \in E_{\Upsilon^3}} c^3(e) < (R+2)\beta$.*

Note, this theorem implies that an optimal solution to the TSP problem (P3) provides an optimal solution to the MGTSP problem (P2).

Before proving this result, we highlight the procedure for constructing the optimal solution Υ^2 given the optimal solution Υ^3 to Problem (P3). First, we find the indices of all the finish-depot vertices used in Υ^3. If the indices are $\{l_1, l_2, \ldots, l_M\}$, pick the vertices immediately following them in the tour as $\{l_1+1, l_2+1, \ldots, l_M+1\}$. These are the start-depots of each

individual path. Between every pair of start-depot and finish-depot indices $(l_i{+}1, l_{i+1})$, use the Noon-Bean method to select vertices for each set in $\{V_1^2, \ldots, V_R^2\}$. This can be performed as long as the condition in Theorem V.3 is satisfied.

We now prove Theorem V.3 through a sequence of lemmas.

Lemma V.4. *In any TSP solution to Problem (P3), each start-depot vertex $v_d^i \in V_0^3$ will be immediately preceded by the finish-depot vertex, $v_f^{i-1} \in V_0^3$ in the chosen cyclic ordering of vertices in set V_0^3.*

Proof. Every start-depot $v_d^i \in V_0^3$ has an in-degree of one. Hence a path visiting a start-depot can do so only through the preceding finish-depot vertex in the given cyclic ordering of V_0^3. □

This simple result implies that the indices of the finish-depot vertices will allow us to "cut" a single TSP tour into paths for each working robot. Lemmas V.5 and V.6 define the method and conditions under which the TSP solution to Problem (P3) provides the MGTSP solution to Problem (P2).

Lemma V.5. *The optimal TSP solution to Problem (P3) can be used to construct the optimal MGTSP solution to Problem (P2).*

Proof. According to the modified Noon-Bean transformation, if an optimal MGTSP solution to Problem (P2), Υ^2, is defined by the set of M paths as,

$$\{\{v_d^1, v_j, \ldots, v_k, v_f^1\}, \ldots, \{v_d^M, v_a \ldots, v_b, v_f^M\}\},$$

then the corresponding optimal TSP solution Υ^3 to the transformed problem (P3) will be,

$$v_d^1, v_j, v_{j+1}, \ldots, v_{j-1}, \ldots, v_k, v_{k+1}, \ldots, v_{k-1}, v_f^1,$$
$$v_d^M, v_a, v_{a+1}, \ldots, v_{a-1} \ldots, v_b, v_{b+1}, \ldots, v_{b-1}, v_f^M, v_d^1$$

The optimal TSP path visits all vertices in vertex sets $\{V_1^3, \ldots, V_R^3\}$ in a clustered manner as shown in the Noon-Bean transformation. The vertices of set V_0^3 are visited intermittently between interset transitions in finish-depot, start-depot pairs as specified in Lemma V.4. As stated in Lemma V.4, the TSP tour can be cut into optimal paths for each of the charging robots. Further, given that each interset edge of Υ^3 has a cost equal to the corresponding interset edge in Υ^2, we can determine that $\sum_{e \in E_{\Upsilon^3}} c^3(e) = \sum_{e \in E_{\Upsilon^2}} c^2(e)$. □

We know that an optimal solution (P3) always corresponds to the optimal solution to (P2). Lemma V.6 extends this result to define the condition under which a feasible TSP solution to (P3) can provide a feasible solution to (P2)

Lemma V.6. *A feasible TSP solution, Υ^3, to Problem (P3) provides a feasible MGTSP solution, Υ^2, to Problem (P2) given that $\sum_{e \in E_{\Upsilon^3}} c^3(e) < (R+2)\beta$.*

Proof. From the Noon-Bean transformation [28], we know that a feasible GTSP solution through $R+1$ vertex sets contains $R+1$ interset edges and the cost of a corresponding TSP solution cannot exceed $(R+2)\beta$.

In the case of multiple charging robots, the number of interset edges in the solution depends on the number of charging robot routes. However, since the edges incident on finish-depots in V_0^3 do not have the penalty, β, added to their cost, the number of large-cost interset edges in the solution is $R{+}1$, independent of the number of charging robot routes used. Hence, a feasible solution to Problem (P2) can be constructed from a solution to Problem (P3), if $\sum_{e \in E_{\Upsilon^3}} c^3(e) < (R{+}2)\beta$. □

Combining the three lemmas above we arrive at the final result of Theorem V.3. Notice that in the case of both Lemma V.5 and V.6, the cost of the constructed MGTSP solution is equal to the cost of the TSP solution. Hence, $\sum_{e \in E_{\Upsilon^3}} c^3(e) = \sum_{e \in E_{\Upsilon^2}} c^2(e)$.

VI. Recurring Charge Cycles Over Long Planning Horizons

This section extends the single recharge cycle problem to multiple recharge events over a longer mission span by computing an optimal periodic recharge schedule over the entire planning horizon. A fixed horizon mixed integer linear program (MILP) is presented and solved in Section VI-A. Further, an alternative approach to greatly reduce computational effort using a receding horizon approach is shown to empirically perform sufficiently well.

A. Optimal Periodic Recharging

The fixed horizon approach to path planning involves computing an optimal path over the entire planning horizon. This approach, although significantly increasing the size of the problem, guarantees optimality of rendezvous paths over the lifetime of the mission. As in the single recharge cycle computation, in the periodic recharging problem, working robot trajectories are known for the entire planning range $[0, T]$. In this section, however, the objective is to compute charging robot paths that rendezvous with working robots at a sequence of charging points such that no working robot runs out of charge during the mission. The problem is approached as follows.

1) Challenges: Three main factors distinguish the periodic charging problem from the single charge cycle problem:

(i) Arrival times of working robots at charging points cannot be determined *a-priori*, since they depend on previous rendezvous' in their paths.
(ii) The time elapsed between consecutive recharges of each robot must be constrained to ensure successful continued operation.
(iii) The variability of arrival times at charging points implies that the feasibility condition applied on a path between them, as defined in Equation 2, cannot be predetermined.

2) Approach: Given these considerations, we formulate the periodic charging problem as an optimization on a partitioned graph G as follows.

Vertices: Define a set of vertices V that is partitioned into $R+1$ disjoint vertex sets, $V_0, V_1, \ldots, V_R$. The set V_0 is the set of start-depots of the charging robots. The vertices in each set V_r, $r \in \mathcal{R}$, correspond to charging locations in C_r.

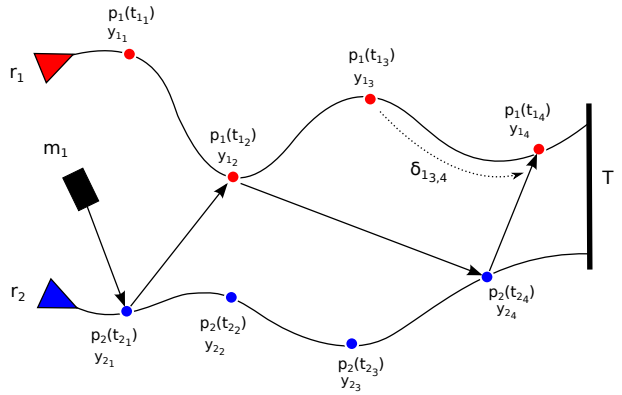


Fig. 5: The periodic MILP representation: An sample problem instance illustrating charging point discretization and key variables. A path for charging robot m_1 is computed to visit charging point sets V_{r_1} and V_{r_2}, for robots r_1 and r_2, periodically to ensure $y_{r,i} < \tau_r$.

The charging point set for periodic charging, $C_r = \{p_r(t)|t \in T\}$ for robot $r \in \mathcal{R}$ is defined as the set of locations $p_r(t)$ that a robot would visit along its trajectory, given infinite charge and no recharge stops. The estimated arrival times at the charging points will be updated as part of the optimization.

Edges: Edge-feasibility is subject to change based on UAV arrival times at charging points. Hence define all edges (v_i, v_j) where $v_i \in V_a$, and $v_j \in V_b$ for $a, b \in \mathcal{R}$ and $a \neq b$ as valid edges in the periodic charging graph. The edge-feasibility condition will be applied as a constraint in the optimization.

Costs: The cost on an edge can be defined based on the optimization objective. In this formulation we consider the distance between two charging locations.

In addition to the graph G, we introduce two sets of variables, $y_{r,i}$ and $t_{r,i}$. The variable, $y_{r,i} \in \mathbb{R}_+$, stores the value of the time elapsed since the last recharge of robot r, at each charging point i in C_r. By placing a bound, τ_r, on the maximum value of $y_{r,i}$, we can ensure that robot r will always rendezvous with a charging robot before it is completely discharged. The variable, $t_{r,i} \in \mathbb{R}_+$, computes the time of arrival of a UAV r at its charging point i in set C_r. The value of $t_{r,i}$ is computed at each point taking into account the service times at charging points chosen for rendezvous'.

A sample instance of the discretized problem for optimal periodic charging is shown in Figure 5.

We can now formally state the optimal periodic recharging problem.

Problem VI.1 (Optimal Periodic Charging Problem)**.** Consider a partitioned DAG G with the partition $(V_0, V_1, \ldots, V_R)$ of V where $V_0 = \{v_m|m \in \mathcal{M}\}$. Find a set of paths $P = \{P_1, \ldots, P_M\}$ in G that minimize $\sum_{i=1}^{M} \sum_{e \in E_{P_i}} c(e)$ and satisfy the constraints (i) $P_m \in P$ starts at $v_m \in V_0$, such that $|V_r \cap V_P| \geq 1$ for all $r \in \mathcal{R}$ and (ii) $y_{r,i} < \tau_r$ for all $r \in \mathcal{R}$ and all $v_i \in V_r$.

B. Periodic MILP Formulation

Given the problem statement, the periodic charging MILP can be defined as an extension to the single charge cycle ILP defined in Section IV. For ease of presentation, the MILP is formulated to compute a single charging robot path through a team of UAVs performing a persistent task. The extension to multiple robots is straightforward as shown in Section IV-B.

The periodic charging MILP refers to a vertex v_i with an index i in the context of the complete vertex list V, as well as an index within each working robot vertex set V_r. Hence, to avoid ambiguities in the indices, we define the set of vertex indices of V_r as $I_{V_r} = \{1, \ldots, |V_r|\}$, and the set of vertex indices of V as $I_V = \{1, \ldots, N\}$. Finally, we define the index function $\sigma : \mathcal{R} \times I_{V_r} \rightarrow I_V$ as a function that takes a working robot index $r \in \mathcal{R}$ and the local index of the charging vertex $i \in I_{V_r}$ and returns the global index of the vertex in the complete vertex list I_V.

The objective of the periodic charging problem, as inherited from the single recharge cycle integer program, is to minimize the total sum of path costs of the charging robots.

$$\min \quad \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{11}$$

The constraints of the periodic optimization inherit degree and flow constraints of the One-in-a-set DAG path problem and extend the problem definition to fulfill periodic charging. Constraints (5), (6) and (9) are inherited directly. Set degree constraints (8) and (9) are modified to form Constraints (12) and (13) to allow multiple recharge rendezvous' within each set V_r of a robot r:

$$\sum_{j \in V_r} \sum_{i \in V} x_{ij} \geq 1, \quad \forall r \in \mathcal{R} \tag{12}$$

$$\sum_{i \in V_r} \sum_{j \in V} x_{ij} \geq 1, \quad \forall r \in \mathcal{R} \tag{13}$$

Constraint (14) computes the value of $t_{r,i}$, the arrival time at each charging point, as the sum of the arrival time at the previous charging point, $t_{r,i-1}$, the service time s_r at the point if a recharge has taken place, and the travel time, $\delta_{r_{i-1,i}}$, between two consecutive charging points.

$$t_{r,i} = t_{r,i-1} + s_r \sum_{j \in V} x_{j\sigma(r,i-1)} + \delta_{r_{i-1,i}}, \quad \forall r \in \mathcal{R} \quad \forall i \in I_{V_r} \tag{14}$$

Given the value for $t_{r,i}$ at each charging point, an edge feasibility constraint for every edge in the graph can now be defined. Constraint (15) is defined as a logical or implication constraint which ensures that if the value of $x_{\sigma(r_1,i)\sigma(r_2,j)} = 1$, signifying an active edge in the solution, then the feasibility constraint as shown in constraint (15) must be satisfied. Logic constraints can be reformulated into MILP constraints using linear relaxations and *big-M* formulations as shown in [40].

$$x_{\sigma(r_1,i)\sigma(r_2,j)} = 1 \implies t_{r_2,j} - t_{r_1,i} > \frac{d_{\sigma(r_1,i)\sigma(r_2,j)}}{v} \quad \forall r_1, r_2 \in \mathcal{R};\ r_1 \neq r_2 \quad \forall i \in I_{V_{r_1}} \quad \forall j \in I_{V_{r_2}} \tag{15}$$

Note that $d_{\sigma(r_1,i)\sigma(r_2,j)}$ is the distance between the two charging points. The final three constraints (16), (17) and (18) compute the value of $y_{r,i}$ and ensure that it is bounded by τ_r.

Constraint (16) computes the value of $y_{r,i}$ at every charging point, where a rendezvous does not occur, as the sum of $y_{r,i-1}$ and $\delta_{r_{i-1,i}}$.

$$\sum_{j \in V} x_{j\sigma(r,i)} = 0 \implies y_{r,i} - y_{r,i-1} = \delta_{r_{i-1,i}} \quad \forall r \in \mathcal{R} \quad \forall i \in I_{V_r} \tag{16}$$

Constraint (17) resets the value of $y_{r,i}$ to 0 at a charging point chosen for rendezvous. Thus the value of $y_{r,i}$ increments throughout the charging point set, occasionally resetting to 0 at points where recharge rendezvous' occur.

$$\sum_{j \in V} x_{j\sigma(r,i)} = 1 \implies y_{r,i} = 0 \quad \forall r \in \mathcal{R} \quad \forall i \in I_{V_r} \tag{17}$$

Finally Constraint (18) limits the growth of $y_{r,i}$ to guarantee that robot r is consistently charged through the mission.

$$0 \leq y_{r,i} \leq \tau_r, \quad \forall r \in \mathcal{R} \quad \forall i \in I_{V_r} \tag{18}$$

The total number of constraints in this formulation is $2M + 2R + N^2 + 5N$, of which $N^2 + N$ are implication constraints. Similar to the single recharge cycle ILP defined in Section IV, the complexity of the fixed horizon problem is influenced by the number of vertices in the graph G. For a given scenario of working and charging robots, the size of the G grows with the length of the planning horizon $[0, T]$ and the density of charging locations on each robot path.

This formulation produces a significantly larger constraint set than the single recharge cycle ILP and as a result, the fixed horizon MILP quickly becomes intractable for larger problem instances. An alternative approach to minimize computational effort is a receding horizon strategy.

Remark VI.2 (Receding Horizon Planning). Receding horizon (RH) methods have been extensively applied in MILP based motion planning [41] to minimize computational effort and enhance robustness of the computed path. In a general receding horizon formulation, a path plan is computed and implemented over a shorter time window and then iteratively updated from the state reached at each re-planning event, for the duration of the planning horizon. In the context of this problem, an RH strategy may be employed as an alternative to the fixed horizon optimization, using an iterative computation of single cycle recharge paths. The downside of this approach is that there is no guarantee that every consecutive planning iteration will generate a feasible path solution. However, with a good re-planning strategy, and sufficient robustness measures we argue that this method empirically presents a significant improvement over the fixed horizon method as described in Section VII-C and shown in the simulation results in Section VII. •

VII. Simulation Results

The optimization framework for this paper was implemented and tested in simulated experiments generated in two simulation environments, (i) MATLAB®, to investigate the solution quality and runtime performance of the proposed algorithms and (ii) ROS Gazebo to study the robustness of our methods in a realistic implementation. The mixed integer linear programs were solved optimally using the IBM CPLEX® solver and the TSP heuristic used in the computation was the freely available LKH Solver [26]. The solutions were computed on a laptop computer running a 32 bit Ubuntu 12.04 operating system with a 2.53 GHz *Intel Core2 Duo* processor and 4GB of RAM.

The simulation environment consists of a test set of planar trajectories that are assigned to a team of R working robots. Each working robot r is defined by its assigned trajectory, current pose, voltage level and battery lifetime T_r. The environment also contains a set of M randomly located charging robots, each defined by an initial position and a maximum velocity, v. The goal is to enable the working robots to persistently traverse their assigned trajectories for the duration of mission such that the total distance travelled by the charging robots is minimized.

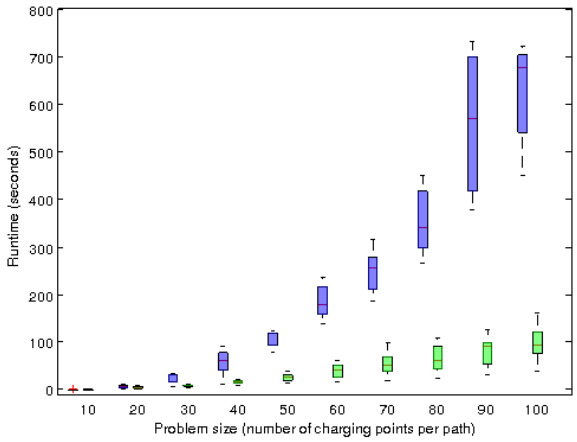
A. Single Recharge Cycle Path Computation

To benchmark the performance of the TSP-based solver against the optimal CPLEX solution, we conducted an experiment examining the effect of growth in problem complexity based on the charging point density (path sampling resolution) on the runtime and solution quality for both solvers.

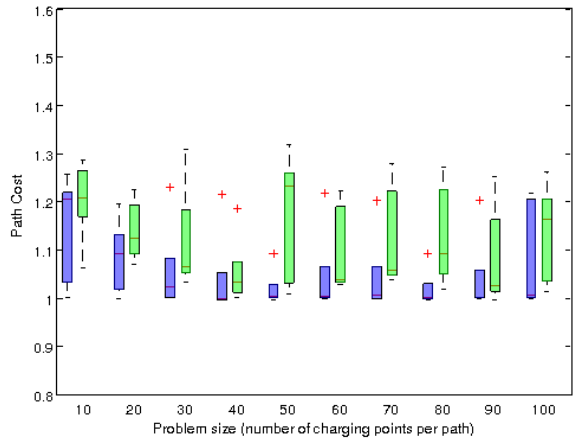
A test set of simulation environments with different path and robot configurations was created. Given each environment configuration, the complexity of the path optimization was varied by incrementing the density of charging points along each working robot trajectory from $\{10, 20, 30, \ldots, 100\}$ charging points per path. The recharge path was computed several times for each charging point density level. The resulting runtimes and path lengths for each environment are normalized to show trends in performance with growth in problem complexity. The results are summarized in Figure 6 using box plots that show the spread of results over each charging point density level, using quartiles (box edges), extreme data points (whiskers) and outliers (crosses). Boxes for the optimal and heuristic solutions are plotted adjacently for each x-axis data point.

Figure 6(a) demonstrates the growth in runtime for the optimal CPLEX solution and the LKH heuristic solution with a growth in problem complexity. Similarly, Figure 6(b) demonstrates the trend in path costs for the optimal and heuristic solutions. The optimal path cost for each simulation environment is generally consistent for all problem sizes since the complexity is varied by only increasing the number of charging points per path. The results show that, on average, as problem complexity grows, the optimal solver grows exponentially in runtime and the heuristic solver consistently provides solutions within 10% of the optimal with significant savings in computational effort.

An example environment is shown in Figure 7, consisting of eight working robots distributed among eight paths and three charging robots. The DAG for this problem instance consists of 500 vertices. The optimal solution was computed by CPLEX in 97 seconds. The heuristic solution was computed by LKH in 1.2 seconds and resulted in a path cost 7.8% higher than the optimal cost.



(a) Runtime comparison: Optimal CPLEX (blue) vs. LKH heuristic (green).



(b) Path cost comparison: Optimal CPLEX (blue) vs. LKH heuristic (green).

Fig. 6: Performance comparison of Optimal CPLEX and LKH TSP heuristic solutions.

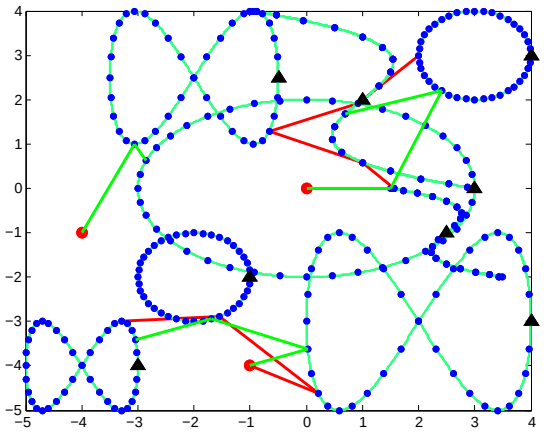


Fig. 7: Comparison of the optimal CPLEX solution (light grey/green path) against the Modified Noon bean Transform and LKH Heuristic solution (dark grey/red path). The problem consists of eight working robots (triangles) on eight paths and three charging robots.

B. Real-World Simulation Environment

In order to observe the effects of uncertainties in the dynamics and controller response of the vehicles on the robustness of the computed recharge schedule, we have constructed a simulation using the ROS/Gazebo environment, with full dynamics and control models of quadrotors and ground vehicles operating on non-planar terrain and with 3D trajectories.

The simulation includes the *Gazebo Hector Quadrotor* UAV model and the *Gazebo Husky* UGV model to simulate the dynamics of aerial and ground vehicles in operation. In these simulations we have observed that issues such as nonlinearities in thrust generation, rotations and controller performance do not significantly effect the ability of the robots to execute their recharge plans. This appears to be a result of the difference in the time scales of the controller performance (which is on order of seconds) and that of the battery life and recharge schedule (which is on the order of tens of minutes) [42], [43]. The accompanying video submission shows a representative simulation. The paths used in this video were chosen for demonstration purposes, and are not limited by the proposed approach.

C. Robustness Measures in Offline Plan

Through the real-world simulations, we have observed that the plans computed offline typically have enough safety margin in their timings that they remain feasible even with timing errors that occur in practice (e.g., a charging robot arrives late for a charging rendezvous). However, we can explicitly build safety margins into the offline plans to increase robustness. In this subsection we briefly describe methods for building safety margins into i) docking and recharge times, and ii) the UAV battery life estimation. For both sections, the cost used is the total distance traveled by the recharging vehicles. In this section we have attempted to create "hard" problem instances in which small timing deviations result in many charging failures.

1) Docking and Recharge Time Errors: As shown in equation (2), the offline plan relies on estimates of the time to dock and recharge robot r_1 at charging point i as $\Delta t_{r_1,i}$. When estimating this time, we can add a safety margin of $\lambda > 0$. The condition for adding an edge to the graph in equation (2) then becomes $\frac{\|p_{r_2}(t_{r_2,j}) - p_{r_1}(t_{r_1,i})\|}{v} \leq t_{r_2,j} - (t_{r_1,i} + \Delta t_{r_1,i} + \lambda)$.

By increasing λ, we increase the robustness of the solution to uncertainty in docking and charging times. This is demonstrated in Figure 8, which shows results for a persistent surveillance mission consisting of two UGVs and four UAVs along independent paths monitoring a 250000 m^2 area. The UAV battery life is set to 30 minutes and for each value of λ from 0 to 3 minutes, we execute 20 simulation runs in which the true docking and recharging time is a uniform random number between $0.5\Delta t_{r_1,i}$ and $1.5\Delta t_{r_1,i}$. In each simulation run, we record the number of UAVs that fail to recharge before reaching zero charge as well as the distance traveled by UGVs. One can see that by building in a larger safety margin, we are able to charge all UAVs (even in this adversarially chosen problem instance), but that comes at the expense of the distance traveled by each UGV).

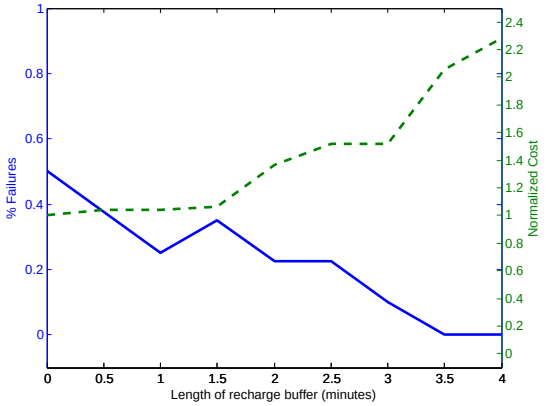


Fig. 8: Failure count against buffer size λ.

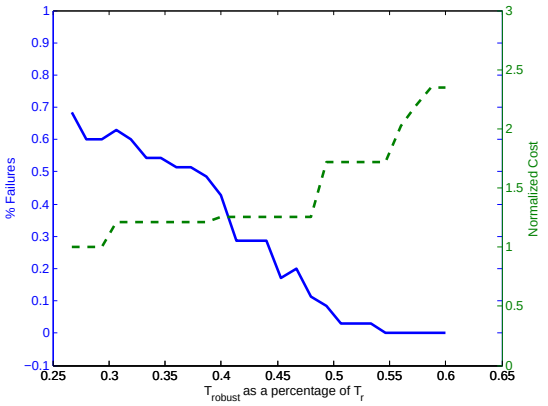


Fig. 9: Failure count against T_{robust}.

2) UAV Battery Life Estimation: In the computation of an optimal recharge schedule, the battery life of UAV r is estimated as T_r. This quantity can be estimated using existing battery discharge curves for UAVS [44], [45], [46], but will be subject to errors.

We can build a safety margin into the plan by underestimating the battery life by a value $T_{\text{robust}} < T_r$, thus requiring the recharge to occur before $T_r - T_{\text{robust}}$. We have performed extensive simulations using varying battery lifetimes and vehicle speeds. A sample set of results is presented in Figure 9 with a team of four UAVs and two UGVs in a $250000m^2$ environment, and estimated battery lifetimes of 30 minutes for each UAV. For a set of values of T_{robust} within 0 and 60% of T_r, we ran 30 simulation runs each with a true battery lifetime uniformly distributed between $0.7T_r$ and $1.3T_r$.

We see that the safety margin successfully reduces the number of failures, but at the expense of system performance. Also note that in practice, we can estimate the battery level in real-time using Coulomb counting and then re-plan periodically when the charge remaining deviates from the values used in planning.

D. Persistent recharging in extended planning horizons

The following simulation experiments examine the receding horizon and fixed horizon methods of computing recharge paths over an extended planning horizon. For appropriate benchmarking, the receding horizon strategy is implemented by computing the optimal ILP solution at each planning iteration and compared with the optimal fixed horizon path over the planning horizon.

TABLE I: Fixed Horizon Runtimes

Horizon (minutes)	Runtime Quartiles (seconds)		
	25%	Median	75%
10	0.12	0.33	0.42
15	5.34	10.23	14.04
20	12.65	21.14	45.87
25	91.71	200.14	500.32
30	254.03	401.55	801.39
35	712.28	900.61	+1000
40	968.75	+1000	+1000

TABLE II: Receding Horizon Runtimes

Horizon (minutes)	Runtime Quartiles (seconds)		
	25%	Median	75%
10	0.11	0.14	0.18
15	0.12	0.15	0.20
20	0.16	0.20	0.30
25	0.21	0.28	0.38
30	0.25	0.32	0.49
35	0.31	0.43	0.58
40	0.37	0.54	0.68

The computational effort required by the receding horizon method is significantly less than the fixed horizon strategy due to a shorter planning window and a much smaller ILP formulation. For the same reason, however, global optimality is not guaranteed over the entire planning horizon. To investigate this trade-off, we conducted an experiment, similar to the single recharge cycle tests, examine the runtime and solution quality for both methods.

A test set of simulation environments with different path and robot configurations was created. For each simulated environment, the recharge path was computed using both the receding and fixed horizon methods for a set of different planning horizons from $\{10, 15, 20, \ldots, 40\}$ minutes assuming the estimated lifetime of each working robot to be 6 minutes. For all receding horizon simulations, the replanning window was chosen to be $R/2$ rendezvous' per iteration. The optimization of each planning strategy was aborted if a solution was not found in 1000 seconds. The aggregate results for cumulative runtime and total path cost are summarized in Tables I and II due to the large differences in results between the two strategies.

Tables I and II demonstrate the spread in the growth of runtime for the fixed horizon strategy and the receding horizon strategy respectively with a growth in planning horizon, using quartiles, similar to the box plots. For each planning horizon, the 25th percentile, 75th percentile and median of the runtime results are shown. Figure 10 compares the normalized path costs for both methods with a box plot.

The results show that, on average, as problem complexity

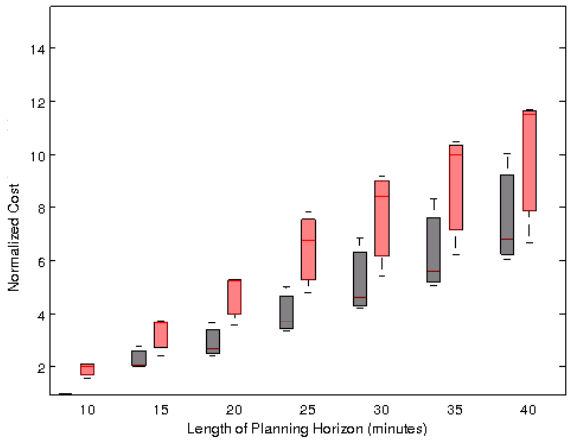


Fig. 10: Total path cost: Fixed horizon (dark/black) and Receding horizon (light/red)

TABLE III: Receding horizon cumulative runtime

Planning window	1	2	4	6	8
Runtime (seconds)	6.14	3.49	2.03	1.41	infeasible

grows, the growth in runtime for the fixed horizon solver is exponential with a wide spread of growth rates based on the problem configuration. On the contrary, the receding horizon strategy consistently results in a significantly smaller cumulative runtime even with an optimal ILP solution at each iteration. On average, the receding horizon method produces solutions with a total path cost within 20% of the optimal fixed horizon solution.

We also investigated the effect of varying the planning window size on the receding horizon strategy. For a set of 8 working robots and 3 charging robots, a test set of simulation environments with different path configurations and charging point densities was generated. For each environment, the planning window was varied from 1 rendezvous to 8 rendezvous' and the receding horizon solution was computed for each window size. Tables III and IV show the normalized results of cumulative runtime and path cost, respectively, averaged over all the experiments.

It is important to note that in the presented receding horizon strategy, at each iteration, regardless of planning window, the optimal recharge path is computed to visit all working robots. Hence, Table III shows that the cumulative runtime generally drops as the planning window grows, due to fewer replanning iterations. However, a larger planning window also increases the possibility of the path reaching an infeasible solution as seen with the planning window of 8 rendezvous' per iteration. Table IV shows that the cumulative path cost over the planning horizon is not significantly affected by the size of the planning window.

TABLE IV: Receding horizon total path cost

Planning window	1	2	4	6	8
Total Path Cost	1.22	1.01	1.23	1.02	infeasible

VIII. Conclusions and Future Work

In this paper, we presented the problem of persistent recharging with coordinated teams of autonomous robots.

We proved that the One-in-a-set DAG problem is NP-hard and presented a ILP formulation for a single recharge cycle. We also presented an approach which uses the Noon-Bean transformation to obtain a TSP problem instance that can be solved with a TSP heuristic solver. Subsequently, we proposed a novel modification to the Noon-Bean transformation to address the MGTSP case and find multiple rendezvous paths for M charging robots. Simulation results show that the heuristic solution using the Modified Noon-Bean transformation and LKH solver is a viable alternative that produces solutions of comparable cost and significant runtime savings. Finally, we extend the problem to longer planning horizons using a receding horizon and fixed horizon approach. Simulations demonstrate the trade-off between optimality and computational complexity presented by the two alternatives.

Multiple avenues for extension present themselves. The restriction that working robots cannot be charged except at fixed charging points could be relaxed to allow for deviation from the prescribed trajectory to accommodate recharging, and static charging depots could be added to the formulation as well. Heterogeneity of ground vehicle speed could be addressed by expanding the graph definition to have duplicate nodes for each charging vehicle, so that edge costs could be made vehicle dependent, but at significant computational cost. These modifications are left as areas of future work.

The main challenge faced by the receding horizon approach is ensuring that each subsequent planning iteration admits a feasible solution. One way to mitigate this issue is to incorporate terminal constraints for each planning iteration to ensure continued feasibility of path solutions [47]. Implementing safety constraints in the MILP formulation is a future direction for this work. In the fixed horizon strategy, in addition to high computational complexity, another drawback is poor robustness to uncertainties or modelling errors. Since the computation is performed offline, this strategy does not adapt the charging schedule to incorporate disturbances and mistiming errors in the execution of the optimal plan. However, robustness strategies such as *reactive rescheduling* [48] may be used to make it an effective planning strategy. Robustness of optimal path plans is a future direction for this work.

References

[1] N. Mathew, S. L. Smith, and S. L. Waslander, "A graph based approach to multi-robot rendezvous for recharging in persistent tasks," in *IEEE International Conference on Robotics and Automation*, 2013.

[2] J. A. D. E. Corrales, Y. Madrigal, D. Pieri, G. Bland, T. Miles, and M. Fladeland, "Volcano monitoring with small unmanned aerial systems," in *American Institute of Aeronautics and Astronautics Infotech Aerospace Conference*, 2012, p. 2522.

[3] D. Kingston, R. Beard, and R. Holt, "Decentralized perimeter surveillance using a team of UAVs," *Robotics, IEEE Transactions on*, vol. 24, no. 6, pp. 1394–1404, 2008.

[4] M. Burri, J. Nikolic, C. Hürzeler, J. Rehder, and R. Siegwart, "Aerial service robots for visual inspection of thermal power plant boiler systems," in *Proc. of the 2nd Int. Conf. on Applied Robotics for the Power Industry*, 2012.

[5] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243 – 255, 2004.

[6] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 410–426, 2012.

[7] F. Pasqualetti, J. W. Durham, and F. Bullo, "Cooperative patrolling via weighted tours: Performance analysis and distributed algorithms," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1181 –1188, 2012.

[8] K. Swieringa, C. Hanson, J. Richardson, J. White, Z. Hasan, E. Qian, and A. Girard, "Autonomous battery swapping system for small-scale helicopters," in *IEEE International Conference on Robotics and Automation*, May 2010, pp. 3335 –3340.

[9] K. Suzuki, P. Kemper Filho, and J. Morrison, "Automatic battery replacement system for UAVs: Analysis and design," *Journal of Intelligent and Robotic Systems*, vol. 65, pp. 563–586, 2012.

[10] A. Couture-Beil and R. Vaughan, "Adaptive mobile charging stations for multi-robot systems," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, Oct 2009, pp. 1363–1368.

[11] N. Michael, E. Stump, and K. Mohta, "Persistent surveillance with a team of MAVs," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011, pp. 2708–2714.

[12] J. Derenick, N. Michael, and V. Kumar, "Energy-aware coverage control with docking for robot teams," in *IEEE International Conference on Intelligent Robots and Systems*, 2011, pp. 3667–3672.

[13] Z. Jin, T. Shima, and C. Schumacher, "Optimal scheduling for refueling multiple autonomous aerial vehicles," *Robotics, IEEE Transactions on*, vol. 22, no. 4, pp. 682–693, Aug 2006.

[14] ——, "Scheduling and sequence reshuffle for autonomous aerial refueling of multiple uavs," in *American Control Conference, 2006*, June 2006, pp. 6 pp.–.

[15] J. Barnes, V. Wiley, J. Moore, and D. Ryer, "Solving the aerial fleet refueling problem using group theoretic tabu search," *Mathematical and Computer Modelling*, vol. 39, no. 68, pp. 617 – 640, 2004, defense transportation: Algorithms, models, and applications for the 21st century.

[16] H. Shen and P. Tsiotras, "Optimal scheduling for servicing multiple satellites in a circular constellation," in *AIAA/AAS Astrodynamics Specialists Conference*, 2002, pp. 5–8.

[17] Y. Litus, P. Zebrowski, and R. Vaughan, "A distributed heuristic for energy-efficient multirobot multiplace rendezvous," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 130 –135, 2009.

[18] J. Daly, Y. Ma, and S. Waslander, "Coordinated landing of a quadrotor on a skid-steered ground vehicle in the presence of time delays," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sept 2011, pp. 4961–4966.

[19] C. Phan and H. Liu, "A cooperative uav/ugv platform for wildfire detection and fighting," in *Int. Conf. on System Sim. and Scientific Computing, 2008.*, Oct 2008, pp. 494–498.

[20] M. Saska, V. Vonasek, T. Krajnik, and L. Preucil, "Coordination and navigation of heterogeneous uavs-ugvs teams localized by a hawk-eye approach," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct 2012, pp. 2166–2171.

[21] P. Oberlin, S. Rathinam, and S. Darbha, "A transformation for a heterogeneous, multiple depot, multiple traveling salesman problem," in *American Control Conference, 2009. ACC '09.*, June 2009, pp. 1292–1297.

[22] ——, "A transformation for a multiple depot, multiple traveling salesman problem," in *American Control Conference, 2009. ACC '09.*, June 2009, pp. 2636–2641.

[23] K. Sundar and S. Rathinam, "Route planning algorithms for unmanned aerial vehicles with refueling constraints," in *American Control Conference (ACC), 2012*, June 2012, pp. 3266–3271.

[24] J. Bae and S. Rathinam, "An approximation algorithm for a heterogeneous traveling salesman problem," in *ASME 2011 Dynamic Systems and Control Conference and Bath/ASME Symposium on Fluid Power and Motion Control*. American Society of Mechanical Engineers, 2011, pp. 637–644.

[25] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, vol. 21, no. 2, pp. pp. 498–516, 1973.

[26] K. Helsgaun, "General k-opt submoves for the Linkernighan TSP heuristic," *Mathematical Programming Computation*, vol. 1, pp. 119–163, 2009.

[27] D. Applegate, R. Bixby, V. Chvtal, and W. Cook, *The Traveling Salesman Problem: A Computational Study*. New Jersey, USA: Princeton University Press, 2011.

[28] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," *INFOR*, vol. 31, no. 1, pp. 39 – 44, 1993.

[29] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209 – 219, 2006.

[30] H. Chen and X. Li, "Long heterochromatic paths in edge-colored graphs," *The Electronic J. Combin*, vol. 12, no. 1, 2005.

[31] H. Broersma, X. Li, G. Woeginger, and S. Zhang, "Paths and cycles in colored graphs," *Australasian Journal on Combinatorics*, vol. 31, pp. 299–311, 2005.

[32] F. Rubin, "A search procedure for hamilton paths and circuits," *J. ACM*, vol. 21, no. 4, pp. 576–580, Oct. 1974.

[33] M. M. Flood, "The traveling-salesman problem," *Operations Research*, vol. 4, no. 1, pp. 61–75, 1956.

[34] G. Laporte, A. Asef-Vaziri, and C. Sriskandarajah, "Some applications of the generalized travelling salesman problem," *The Journal of the Operational Research Society*, vol. 47, no. 12, pp. pp. 1461–1467, 1996.

[35] A. Scheuer and T. Fraichard, "Continuous-curvature path planning for car-like vehicles," in *Intelligent Robots and Systems, 1997. IROS '97., Proceedings of the 1997 IEEE/RSJ International Conference on*, vol. 2, 1997, pp. 997–1003 vol.2.

[36] D. Eppstein, "Finding the k shortest paths," in *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, 1994, pp. 154 –165.

[37] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 4th ed., ser. Algorithmics and Combinatorics. Springer, 2007, vol. 21.

[38] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

[39] D. L. Applegate, R. E. Bixby, and V. Chvátal, *The Traveling Salesman Problem: A Computational Study*, ser. Applied Mathematics Series. Princeton University Press, 2006.

[40] J. Hooker and M. Osorio, "Mixed logical-linear programming," *Discrete Applied Mathematics*, vol. 9697, no. 0, pp. 395 – 442, 1999.

[41] J. Bellingham, A. Richards, and J. P. How, "Receding horizon control of autonomous aerial vehicles," in *in Proceedings of the American Control Conference*, 2002, pp. 3741–3746.

[42] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, 2012.

[43] L. Garca Carrillo, E. Rondon, A. Sanchez, A. Dzul, and R. Lozano, "Stabilization and trajectory tracking of a quad-rotor using vision," *Journal of Intelligent And Robotic Systems*, vol. 61, no. 1-4, pp. 103–118, 2011. [Online]. Available: http://dx.doi.org/10.1007/s10846-010-9472-1

[44] O. Tremblay, L.-A. Dessaint, and A.-I. Dekkiche, "A generic battery model for the dynamic simulation of hybrid electric vehicles," in *Vehicle Power and Propulsion Conference, 2007. VPPC 2007. IEEE*, Sept 2007, pp. 284–289.

[45] M. Jongerden and B. Haverkort, "Battery modeling," Enschede, January 2008. [Online]. Available: http://doc.utwente.nl/64556/

[46] M. Valenti, D. Dale, J. How, and J. Vian, "Mission health management for 24/7 persistent surveillance operations," in *Vehicle Power and Propulsion Conference, 2007. VPPC 2007. IEEE*, 2007.

[47] M. Earl and R. D'Andrea, "Iterative MILP methods for vehicle-control problems," *Robotics, IEEE Transactions on*, vol. 21, no. 6, pp. 1158–1167, 2005.

[48] C. A. Mendez and J. Cerdá, "An MILP framework for batch reactive scheduling with limited discrete resources," *Computers & chemical engineering*, vol. 28, no. 6, pp. 1059–1068, 2004.

Neil Mathew received the B.A.Sc. degree from the University of Waterloo, Waterloo, ON, Canada in 2010 and the M.A.Sc. degree in Mechanical Engineering with the Waterloo Autonomous Vehicles Laboratory at the University of Waterloo in 2014. His research interests include robotic motion planning in unknown environments, multi-robot coordination and energy aware task scheduling with an emphasis on long-term autonomy.

Stephen L. Smith (SM'05-M'09) received the B.Sc. degree in engineering physics from Queen's University, Kingston, ON, Canada, in 2003, the M.A.Sc. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, in 2005, and the Ph.D. degree in mechanical engineering from the University of California, Santa Barbara, in 2009.

He is an Assistant Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON. From 2009 to 2011, he was a Postdoctoral Associate with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge. His main research interest includes the control of autonomous systems, with a particular emphasis on robotic motion planning and distributed coordination.

Steven L. Waslander received the B.Sc.E degree in Applied Mathematics and Mechanical Engineering from Queen's University, Kingston, ON, Canada, in 1998, the M.S. in Aeronautics and Astronautics from Stanford University in 2002 and the Ph.D. in Aeronautics and Astronautics from Stanford University in 2007.

He is currently an Assistant Professor with the Department of Mechanical and Mechatronics Engineering, University of Waterloo, Waterloo, ON. He is also the Principal Investigator at the Waterloo Autonomous Vehicles Laboratory. His main research interests include the planning and control of autonomous aerial rotorcrafts and ground rovers with a focus on perception and navigation in unknown environments.