Multi-robot Rendezvous Planning for Recharging in Persistent Tasks

Neil Mathew Stephen L. Smith Steven L. Waslander

***Abstract*—This paper presents multi-robot path-planning strategies for recharging autonomous robots performing a persistent task. We consider the case of surveillance missions performed by a team of UAVs. The proposal is to introduce a separate team of dedicated charging robots that the UAVs can dock with in order to recharge and ensure continued operation. The goal is to plan minimum cost paths for charging robots such that they rendezvous with and recharge all the UAVs as needed. To this end, planar UAV trajectories are discretized into sets of charging locations and a partitioned directed acyclic graph subject to timing constraints is defined over them. Solutions consist of paths through the graph for each of the charging robots. The rendezvous planning problem for a single recharge cycle is first formulated as a Mixed Integer Linear Program (MILP), and an algorithmic approach, using a transformation to the Travelling Salesman Problem (TSP), is presented as a scalable heuristic alternative to the MILP. The solution is then extended to longer planning horizons using both a receding horizon and an optimal fixed horizon strategy. Simulation results demonstrate and benchmark the performance of the presented algorithms.**

***Index Terms*—Path Planning for Multiple Mobile Robot Systems, Surveillance Systems, Scheduling and Coordination.**

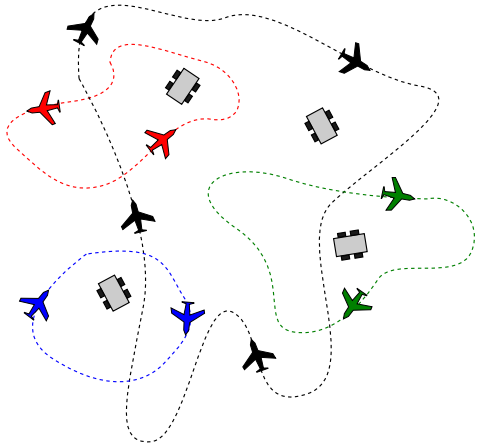


Fig. 1: A sample scenario with coordinated UAVs performing a persistent surveillance task in planar trajectories. We introduce a team of four ground robots capable of docking with and recharging the UAVs.

I. Introduction

COORDINATED teams of autonomous robots are often proposed as a means to continually monitor changing environments in applications such as air quality sampling [2], forest fire or oil spill monitoring [3], [4] and border security [5]. These 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 [6], [7], [8].

The challenge with using autonomous robots in persistent tasks is that mission durations generally exceed the run time of the robots, and in order to maintain continuous operation they need to be periodically recharged or refuelled. In accordance with current persistent surveillance literature, we consider the case of a team of UAVs monitoring an environment as illustrated in Figure 1. This paper presents a recharging or refuelling strategy for a team of *working robots* (UAVs) performing a surveillance task, using one or more dedicated *charging robots* (UGVs) to keep them operational over extended periods of time. The objective is to plan a set of paths for the charging robots to rendezvous with the working robots along their trajectories and recharge them as needed during a surveillance operation. It is advantageous to use mobile charging robots because of the ease of deployment in unknown environments and a greater flexibility that comes with dynamic charging locations. The charging process could be performed, for example, by automated docking and battery swap systems, as demonstrated in [9] and [10]. The charging robots carry a payload of batteries than can be swapped into docked working robots to replenish their charge.

A. Related Work

The problem of persistent coverage and surveillance with mobile robots has been previously investigated in a variety of contexts. Cortes et al. [6] employ Lloyd's algorithm to develop a centroidal Voronoi tessellation-based controller that optimally covers a convex area with a team of mobile robots. Smith et al. [7] design optimal velocity controllers along precomputed paths to persistently cover a set of discrete points with varying desired frequencies of observation, taking advantage of the ability of mobile robots to sense while in motion. While both these works present persistent surveillance scenarios, neither tackle the problem of limited energy resources in the robotic agents.

Persistent surveillance tasks by definition will exceed the range capabilities of any inspection robot, and therefore naturally require inclusion of recharging in their formulations. Derenick et al. [11] propose a modification to [6] which

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)

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 with the assumption that the combined sensor footprint of the agents is sufficient to cover the entire environment. There is also no notion of charge scheduling as each agent is assigned a dedicated static charging station. In the worst case scenario all agents will move towards their charging locations simultaneously, leaving the environment unattended.

Contrary to [11], we introduce the notion of mobile charging stations to minimize hindrances to the surveillance objective and plan optimal paths for charging robots to rendezvous with each working robot. Litus et al. [12], [13] consider the problem of finding a set of meeting points for working robots and a single charging robot, given a static set of locations for all robots and a fixed order of working robots to charge. Since our work addresses a dynamic surveillance scenario, we discretize UAV trajectories into rendezvous locations using a sampling-based roadmap method employed by Obermeyer et al. [14] to plan paths for a UAV conducting visual reconnaissance. Obermeyer et al. abstract the path planning problem onto a roadmap graph, formulate it as a variant of the Travelling Salesman Problem and develop approximation algorithms to solve it. In this paper, we will extend these graph-based techniques to design rendezvous paths for charging robots.

The work presented in [12], [13] and [14] only considers optimal paths that visit desired targets once to fulfill mission objectives. However, in persistent surveillance missions, working robots may require multiple periodic recharges to ensure functionality over the lifetime of the mission. A common approach, as investigated by Bellingham et al. [15] is to use a receding horizon strategy to dynamically extend shorter trajectories over a larger planning horizon. A challenge with this approach is ensuring that each subsequent planning iteration can provide a feasible solution path. Schouwenaars et al. [16] present an iterative MILP path planning approach with implicit safety guarantees that ensure feasibility of successive planning iterations.

An alternative approach is to formulate an optimal path planning problem over the entire planning horizon. Michael et al. [17] investigate a persistent surveillance problem for a team of UAVs to periodically visit a set of interest points with varying frequencies. They formulate the problem as a Vehicle Routing Problem with Time Windows (VRPTW), which is a variant of the classical Vehicle Routing Problem (VRP) that seeks to design routes for multiple vehicles to visit all vertices in a graph. In this work we address an additional challenge of planning optimal paths for a team of charging robots to rendezvous with moving targets (UAVs).

B. Contributions

Our approach to the problem is to position multi-robot persistent recharging in the space of graph-based optimal path planning problems formulated using mixed integer linear programs (MILP). Based on certain enabling assumptions we formulate the problem as finding optimal paths for the charging robots to meet the working robots along their trajectories and recharge them as needed. The main contributions of the paper are the following.

(i) We formally introduce the idea of multi-robot recharging in dynamic persistent tasks using dedicated teams of mobile charging stations.
(ii) We present a graph-based abstraction of the problem and establish it as a one-in-a-set path problem on a partitioned directed acyclic graph (DAG). We prove the problem is NP-hard even on a DAG.
(iii) We formulate the problem as a MILP and investigate special properties of the problem that distinguish it from Travelling Salesman Problem (TSP) and Generalized Travelling Salesman Problem (GTSP) formulations.
(iv) We present an algorithmic solution using novel modifications to existing methods developed for solving the GTSP.
(v) Based on the developed path planning framework we extend the recharging problem to longer planning horizons using receding horizon and fixed horizon strategies.

A preliminary version of this work appeared in [1]. Building on previous work, we now present our algorithmic solution to compute rendezvous paths for multiple charging robots, address the periodic recharging problem over longer planning horizons, and provide extensive simulation results for all algorithms.

II. Preliminaries

The following is a review of relevant graph-theory and fundamental optimal path computation problems that comprise the problem formulations and algorithms employed in this work.

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. The number of vertices in G is given by the cardinality of the set V and is denoted $N = |V|$. 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 exhaustive and mutually exclusive sets $(V_1, \ldots, V_R)$ such that (i) $V_i \subseteq V$ for all i, (ii) $\cup_i V_i = V$, and (iii) $V_i \cap V_j = \emptyset$ for all $i \neq j$.

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.

Definition II.1 (Hamiltonian Path/Tour)**.** A Hamiltonian Path in a graph G is a path P that visits every vertex in G exactly once. Similarly, a Hamiltonian tour T is a closed Hamiltonian Path.

Problem II.2 (The Hamiltonian Path Problem)**.** Given a complete, undirected graph G, does G contain a Hamiltonian

path?

Problem II.3 (Travelling Salesman Problem (TSP))**.** Given a complete 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.

The TSP can be formulated as a MILP and solved using a variety of exact, approximate and heuristic solvers. In this work we use the IBM CPLEX Optimization Studio to compute exact solutions. Since the TSP is an NP-hard problem, it is intractable to compute exact solutions for large problem instances. In such cases, heuristic algorithms may be used to obtain near-optimal solutions with significant reductions in run time. One of the best known algorithms is the Lin-Kernighan heuristic [18] implemented as the Concorde LinKern TSP solver and an adaptation proposed by Helsgaun [19] 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 often produce solutions within 2% of the optimal [20].

There are a number of TSP variants that extend the problem by imposing additional constraints on the graph optimization. The variant employed in this paper is the Generalized Travelling Salesman Problem (GTSP). Let us first define a *One-in-a-set Path*.

Definition II.4 (One-in-a-set Path/Tour)**.** A One-in-a-set Path in a partitioned graph G with a vertex partition $(V_1, \ldots, V_R)$, is a path P that visits a single vertex in every vertex set $V_i \subset G$ exactly once. Similarly, a One-in-a-set tour T is a closed One-in-a-set path.

Problem II.5 (Generalized Travelling Salesman Problem (GTSP))**.** Given a partitioned complete graph G, find a One-in-a-set tour T 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.

The TSP can be seen as a special case of the GTSP where all vertex sets have a cardinality of one [21]. There are a number of approaches to solve GTSP instances developed in literature. Exact algorithms using Lagrangian relaxation and Branch and Cut methods have been proposed by Noon and Bean [22] and Fischetti et al. [23]. Improvement heuristics [24], and meta-heuristics like genetic algorithms [25] and ant colony optimizations [26] have also been proposed.

Another approach, proposed by Noon and Bean [21], involves transforming (or reducing) a GTSP instance into an equivalent TSP instance through a graph transformation, such that the optimal solution to the TSP can be used to construct the optimal solution to the original GTSP. The benefit of such a transformation is that it allows the vast body of existing TSP approaches to be applied to the transformed problems. While such an approach may not necessarily outperform a specialized GTSP algorithm, it provides a platform to verify optimality and compare results of complex routing problems on the foundation of well-studied TSP approaches.

The GTSP can be extended to compute multiple One-in-a-set tours originating at multiple depots as defined in Problem II.6.

Problem II.6 (Multiple Generalized Traveling Salesman Problem (MGTSP))**.** Consider a complete partitioned graph G with vertex partition $(V_0, V_1, \ldots, V_R)$, where V_0 consists of M start-depots. Find a collection of M paths, one for each start depot, which collectively visit each vertex set $(V_1, \ldots, V_R)$ exactly once, with minimum total cost.

In an MGTSP, since all start-depots may not be used in the solution, the decision of the number of routes chosen for the optimal solution is implicit in the formulation. The objective of the optimization can be chosen to minimize the total sum of path costs *(min-sum objective)*, or to minimize the maximum individual path cost *(min-max objective)*.

III. Motion Planning For Charging Robots

Given a team of robots performing a persistent task, the problem in question is to compute paths for the team of charging robots such that they optimally rendezvous with every working robot exactly once. We extend our results to multiple charging rendezvous' over finite planning horizons in Section VI-A. The working robots are not required to divert from their trajectories. This minimizes hindrances to the persistent mission caused by the recharging process. The assumption is made that charging robots possess sufficient energy resources and need not be refuelled or restocked within the planning horizon. The problem can now be formally stated.

A. Continuous Problem Statement

Consider an environment, $\mathcal{E} \subset \mathbb{R}^2$, which contains R working robots, denoted by the set $\mathcal{R} = \{1, \ldots, R\}$, performing a persistent task. Each working robot, indexed by $r \in \mathcal{R}$, is described by its motion along a known trajectory, $p_r(t) \in \mathcal{E}$ within a planning horizon $t \in [0, T_r]$ determined by its lifetime on a single charge, and a charging time window $[\underline{T}_r, \overline{T}_r] \subseteq [0, T_r]$.

The environment also contains M charging robots, denoted by the set $\mathcal{M} = \{1, \ldots, M\}$, that are free to move arbitrarily within $\mathcal{E}$. Each charging robot, indexed by $m \in \mathcal{M}$, is described by its initial position $p_m(0)$ 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, $p_m(t) \in \mathcal{E}$ (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 $p_m(t_r) = p_r(t_r)$.

This constraint states that the team of charging robots must rendezvous at least once with each working robot at a point along its respective path before it runs out of charge. Figure 2 illustrates the problem statement with a team of four working robots following a single path, along with two charging robots.

The continuous-time problem, as stated, requires an optimization over the space of all charging robot trajectories [27]. Hence, discretizing the formulation converts the problem into a more tractable form and allows the application of graph-based linear program techniques to obtain a solution.

B. Problem Discretization

For each working robot r, given that the trajectory $p_r(\cdot)$ is known over the planning horizon, we can discretize its

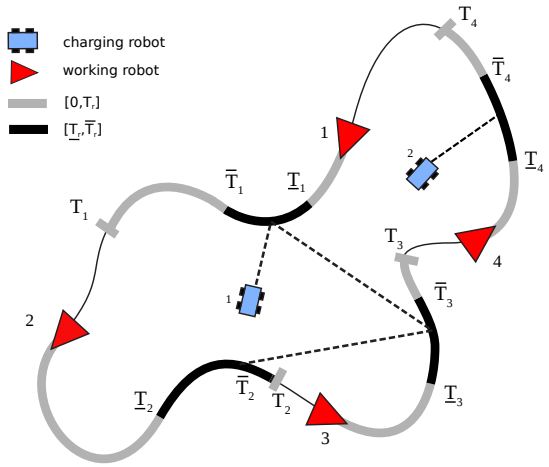


Fig. 2: Four working robots (red triangles) traveling along one path. For each working robot r, $[0, T_r]$ is denoted by a bold grey line and $[\underline{T}_r, \overline{T}_r]$, by a bold black line. The two blue charging robots must meet all working robots on their paths within their charging windows to guarantee persistent operation.

charging time window to generate a set of K_r charging times $\tau_r = \{t_{r,1}, \ldots, t_{r,K_r}\} \subseteq [\underline{T}_r, \overline{T}_r]$ at which it can be reached along its trajectory. The set of charging points 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 its position along the robot path $p_r(t_{r,i})$.

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, as discussed in Remark III.1.

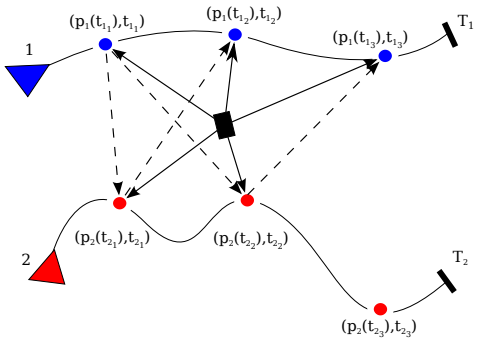
The discrete problem is one of finding paths for the charging robots that visit one charging point in each set C_r. We can encode every possible charging path in a partitioned directed graph G, defined as follows.

a) Vertices: The vertices, are defined by $R + 1$ disjoint vertex sets, $V_0, V_1, \ldots, V_R$. 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 $V = V_0 \cup V_1 \cup \cdots \cup V_R$.

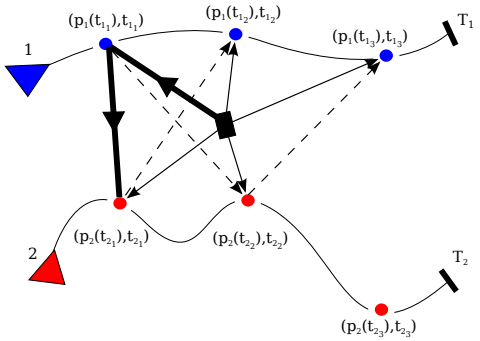
b) Edges: An edge (v_i, v_j) is added to E, where $v_i \in V_{r_1}$ and $v_j \in V_{r_2}$ for some $r_1, r_2 \in \mathcal{R}$ with $r_1 \neq r_2$, to E if there exists a feasible traversable path from charging point $(p_{r_1}(t_{r_1,i}), t_{r_1,i})$ to $(p_{r_2}(t_{r_2,j}), t_{r_2,j})$. That is, if

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

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



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



(b) Optimal Recharge Path Solution

Fig. 3: Building a traversal graph for two working robots and one charging robot. The resulting graph is a directed acyclic graph with vertex partitions.

the objective of the optimization such as minimizing total distance travelled by charging robots or total makespan of the recharging process. Further, in order to avoid recharging UAVs too early, a penalty proportional to the voltage level of robot r at its charging point $(p_r(t_{r,i}), t_{r,i})$ can be added to all incoming edges at each vertex $v_i \in V_r$.

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 to charge robot r at the i^{th} charging point. 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 slightly to

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

•

As a simple illustrative example, Figure 3(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, an interesting property of the constructed graph G is that there are no edges between

vertices of the same vertex set. This property makes 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 a *partitioned directed acyclic graph (DAG)*.

C. Optimization on a Partitioned Directed Acyclic Graph

Given a partitioned DAG G, the goal is to find an optimal path or set of paths that collectively visit each set in the partition once, as shown in Figure 3(b). To characterize the complexity of our problem, it will be helpful to state it as a graph optimization.

We begin by defining the stated problem of computing a set of charging robot rendezvous paths as a decision problem known as the *One-in-a-set DAG Path 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 $i \in \mathcal{R}$?

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 [28]. However, in the following section we prove that the One-in-a-set DAG Path problem is in fact an NP-hard problem.

D. Hardness of Discrete Problem

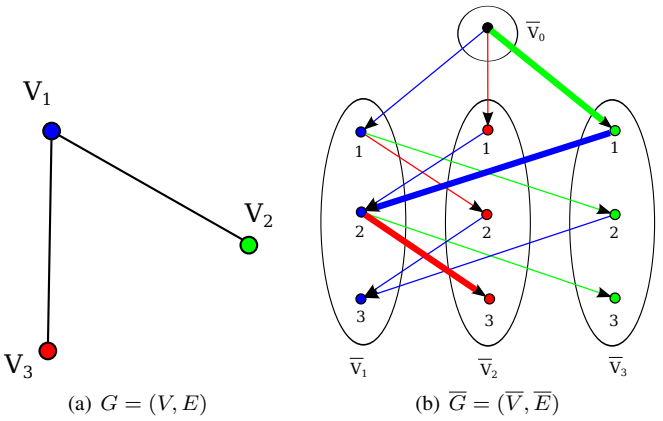
We will prove NP-hardness of the One-in-a-set DAG Path problem by using a reduction from the NP-Complete Hamiltonian path problem [29].

Theorem III.3 (NP-Completeness of Problem III.2)**.** *The One-in-a-set DAG Path problem is NP-Complete.*

Proof: Suppose we have an instance of the Hamiltonian path problem defined on graph G. We will give a polynomial transformation of G into an input $\overline{G}$ for the One-in-a-set DAG Path decision problem.

Given the undirected graph G, we need to create a DAG $\overline{G} = (\overline{V}, E)$ along with the vertex partition $(\overline{V}_0, \overline{V}_1, \ldots, \overline{V}_R)$. Our approach will be to encode every possible Hamiltonian path order in $\overline{G}$. 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 $\overline{V}_R$ be given by R copies of v_r, which we will denote by $\overline{V}_r := (v_{r,1}, \ldots, v_{r,R})$. 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 $\overline{V}_0$ and define $\overline{V} = \overline{V}_0 \cup \overline{V}_1 \cup \cdots \cup \overline{V}_R$.



(a) $G = (V, E)$ (b) $\overline{G} = (\overline{V}, \overline{E})$

Fig. 4: 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 $\overline{G}$

Now, we define the edges $\overline{E}$ as follows. We begin by adding an the edge $(\overline{V}_0, v_{r,1})$ to $\overline{E}$ for each $r \in \{1, \ldots, R\}$. Then for any two sets $\overline{V}_i$ and $\overline{V}_j$ 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 4 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 $\overline{G}$ to the One-in-a-set DAG decision problem. It is easy to see that $\overline{G}$ 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 $\overline{G}$ has $R^2 + 1$ vertices.

Finally, we just need to show that G contains a Hamiltonian path if and only if $\overline{G}$ 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 $\overline{G}$ contains a One-in-a-set path $\overline{P}$. By the definition of the edges $\overline{E}$, the path must be of the form $\overline{V}_0, v_{r_1,1} v_{r_2,2} \cdots v_{r_R,R}$. 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 Hamiltonian path in G. ■

NP-Completeness of the One-in-a-set DAG decision problem implies that Problem III.2 is NP-Complete, and thus 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 MILP for the single charging robot case and use it to characterize the structure of the optimization and inform our solution methods. We then extend the problem to include multiple charging robots and investigate algorithmic alternatives to generate near optimal solutions.

IV. Mixed Integer Linear Program Formulation

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

Given a partitioned graph G defined for the One-in-a-set DAG Path problem, we make a small modification to apply degree constraints. A dummy finish vertex, v_f, is added to V_0 and equal cost edges are assigned from every vertex back to v_f.

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 MILP 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}. Given $M = 1$, the total number of constraints in this formulation is $(2M + 2R + N)$. The maximum number of binary decision variables on the edges of a complete graph is $N(N-1)$. However, owing to the multipartite nature of the graph G, a decision variable x_{ij} is only defined if v_i and v_j belong to different vertex sets.

The complexity of the MILP problem is influenced by the number of binary variables and constraints, which grows with the number of vertices in 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. Special Problem Characteristics

The optimal solution to the MILP 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) [21] as stated in Problem II.5.

Despite structural similarities to the GTSP, it is interesting to note that the MILP formulated for the One-in-a-set DAG Path problem introduces a significantly smaller constraint set than TSP and GTSP routing problems. In addition to the degree and flow constraints stated, routing problems require subtour elimination constraints to ensure a continuous path and avoid disjoint subtours in the solution. A sub-tour elimination constraint in both classes of problems is formulated as

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, \quad \forall S \subset V,\ 2 \leq |S| \leq N - 2,$$

where, for a graph G, N is the total number of vertices in V, S is any subset of vertices in V that can form a sub-tour and x_{ij} is the decision variable on edge $(v_i, v_j) \in E$. A TSP with N vertices requires $2^N - 2N - 2$ subtour elimination constraints. Likewise for a GTSP with N vertices and R vertex sets, the linear program contains as many as $2^{R-1} - R - 1$ subtour elimination constraints [30].

In comparison, the lack of directed cycles in a DAG eliminates any need for sub-tour elimination constraints in our formulation. Further, the multipartite nature of the graph removes the need for binary decision variables on intraset edges. This significant reduction in the number of constraints means that 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 that could be solved in comparable time. Nevertheless, given the NP-hardness of the problem, optimally solving the MILP will not always be computationally tractable and Section V describes the algorithmic approach we use to compute near-optimal solutions.

B. Extending the MILP for Multiple Charging Robots

The 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. Thus 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

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

This expression represents a *min-sum* objective that seeks to minimize the total path cost of all charging vehicles. The MILP can be redefined to include a *min-max* objective that minimizes the maximum path cost of any single charging robot, or, to add constraints to bound each path cost, similar to the well-known Capacitated Vehicle Routing Problem (CVRP) [31]. These extensions are applicable when we wish to set limits on the maximum load capacities of the charging robots, balance their work loads or include depots to restock their charging payload.

Similar to the single charging robot problem, the multiple charging robot problem can be modelled as an MGTSP (see Definition II.6), where each of the M charging robots is assigned a start-depot and, at most, M paths that optimally visit each vertex set once must be computed.

V. Algorithmic Approach: Graph Transformations

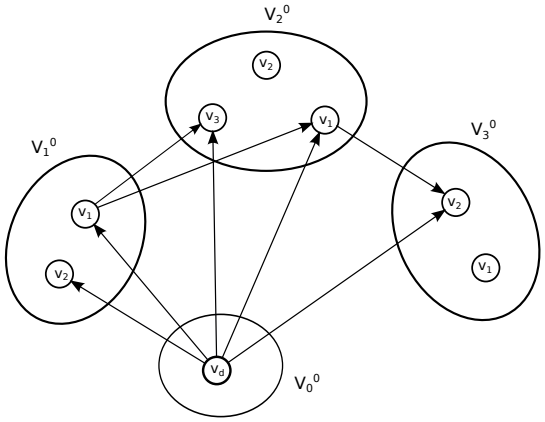
As covered in Section VI-A1, there are a number of algorithmic approaches for solving the GTSP. A common approach is the Noon-Bean Transformation [14], which transforms any GTSP instance into an equivalent TSP instance. The method guarantees that the optimal TSP solution to the transformed problem will always correspond to the optimal solution to the original GTSP.

Section V-A presents an implementation to transform the One-in-a-set DAG path problem, with a single charging robot, into a TSP using the Noon-Bean Transformation. The Noon-Bean method applies only to a problem modelled as a GTSP and in order to solve the multiple charging robot route problem, a specialized solution approach for MGTSP instances is required. Hence, in Section V-B, we propose a novel modification to the Noon-Bean method to allow a transformation of any MGTSP instance into a TSP. The optimal solution to the TSP can then be used to construct the optimal MGTSP solution.

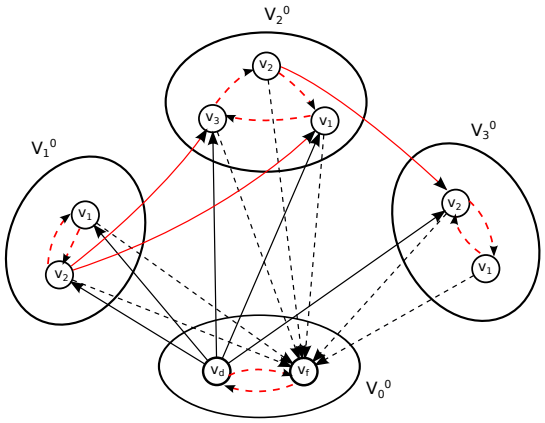
A. Path Computation for a Single Charging Robot

In what follows we define a sequence of problems, beginning with the One-in-a-set DAG Path problem, and ending with a TSP problem. To begin, we define Problem (P0) to be an instance of the One-in-a-set DAG Path problem for a single charging robot. Problem (P0) is a GTSP instance defined on a Partitioned DAG G^0 with a partition of vertices into $R+1$ mutually exclusive sets $V^0 = (V_0^0, V_1^0, \ldots, V_R^0)$. Set V_0^0 contains the start-depot of the charging robot, v_d. We seek the shortest path starting at v_d, and visiting each vertex set exactly once. Figure 5(a) shows a sample instance of the problem.

As shown in [21], the Noon-Bean Transformation can now be used to transform the graph instance in Problem (P0) to new problem (P1), which is a TSP defined on a graph G^1. The vertices V^1, edges E^1 and cost function c^1 are defined as follows.



(a) A sample instance of (P0) with vertex sets (V_1^0, V_2^0, V_3^0) and set V_0^0, which contains the charging robot depots.



(b) The problem instance (P1) generated using the Noon-Bean Transformation. Transformed interset and instraset edges are shown in red.

Fig. 5: The Noon-Bean transformation for GTSPs

(i) Define the set of vertices of G^1, as $V^1 = V^0$. In set V_0^1, add a depot v_f, as the charging robot route finish-depot. Add edges (v_j, v_f), where $v_j \in V^1 \setminus V_0^1$ and assign a cost based on the desired optimization objective.

(ii) For each vertex set V_r^1, create an arbitrary ordering of its vertices $(v_i, v_{i+1}, \ldots, v_{|V_r^1|})$. Add zero-cost directed edges that create a directed cycle through the vertices in the chosen order. The dotted black edges in figure 5(b) show these intraset edges in Problem (P1).

(iii) Shift the tail end of each interset edge $(v_i, v_k) \in E^0$, to (v_{i-1}, v_k), the vertex immediately preceding it in the corresponding intraset cycle. Add these edges to E^1.

(iv) A large penalty $\beta > \sum_{e \in E^1} c^1(e)$ is added to all the interset edges to ensure that the lowest cost TSP tour will never exit a vertex set without traversing the entire intraset vertex cycle.

Problem (P1) is a TSP instance, which can be solved using a variety of freely and commercially available TSP solvers. The goal of the Noon-Bean method is to transform the GTSP into a TSP instance, in which an optimal tour visits all vertices in a vertex set in a clustered manner before moving on to other sets. The penalty β added to interset edges ensures that the shortest tour always contains a clustered solution.

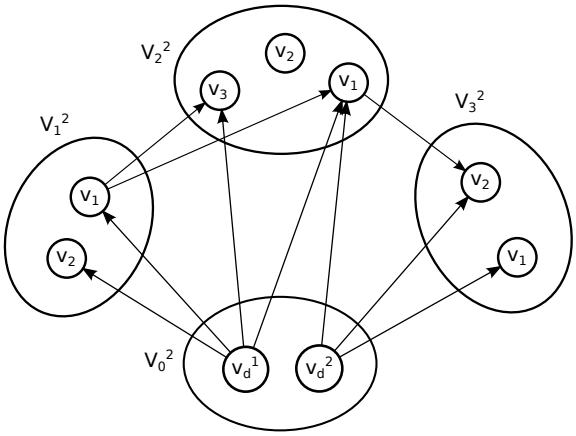


Fig. 6: An sample instance of Problem (P2), with $R = 3$, and $M = 2$

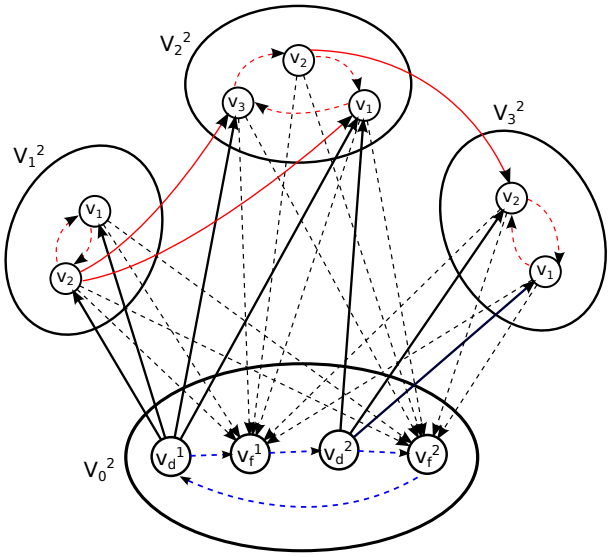


Fig. 7: Problem instance (P3), generated using the modified Noon-Bean algorithm. Red edges represent the Noon-Bean transformation and blue edges represent new additions in the modified algorithm.

The TSP solution to Problem (P1), denoted by Υ^1 can be used to construct the GTSP solution, Υ^0, to Problem (P0), given that it satisfies $\sum_{e\in E_{\Upsilon^1}} c^1(e) \leq (R+2)\beta$. This condition ensures the clustered nature of the TSP tour and stems from the fact that a feasible GTSP solution through $R+1$ vertex sets contains only $R+1$ interset edges. Since $\beta > \sum_{e\in E^1} c^1(e)$, has been added to every interset edge, we know that the cost of a TSP tour that corresponds to a feasible GTSP tour cannot be greater than $(R+1)\beta + \beta = (R+2)\beta$.

A feasible GTSP solution Υ^0, to Problem (P0), can now be constructed by sequentially extracting the entry vertex at every cluster in the TSP tour Υ^1. The total cost of the reconstructed GTSP solution $\sum_{e\in E_{\Upsilon^0}} c^0(e) = \sum_{e\in E_{\Upsilon^1}} c^1(e)$.

B. Path Computation For Multiple Charging Robots

In this section we propose a novel extension to the Noon-Bean method to transform the MGTSP into a TSP. The transformation 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 6 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.

1) Transformation Algorithm: The new transformation algorithm converts the MGTSP problem instance (P2), into a new problem instance (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 Section V-A.

(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 7 illustrates the transformed graph G^3, for Problem (P3). Before proving correctness of the modified Noon-Bean theorem, we require some intermediate results.

Lemma V.1. *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.2 and V.3 define the method and conditions under which the TSP solution to Problem (P3) provides the MGTSP solution to Problem (P2).

Lemma V.2. *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.1. As stated in Lemma V.1, 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.3 extends this result to define the condition under which a feasible TSP solution to (P3) can provide a feasible solution to (P2)

Lemma V.3. *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 Subsection V-A, 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 $\sum_{e \in E_{\Upsilon^3}} c^3(e) < (R+2)\beta$. ∎

In the case of both Lemma V.2 and V.3, the cost of the constructed MGTSP solution is equal to the cost of the TSP solution. Hence, $\sum_{e \in E_{\Upsilon^3}} c^3(e) = \sum_{e \in E_{\Upsilon^2}} c^2(e)$.

C. Reconstructing the MGTSP Solution

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. .

Given the optimal solution Υ^3 to Problem (P3), we can construct, Υ^2, the optimal solution to Problem (P2) as follows. 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 $\{V_1^2, \ldots, V_R^2\}$, as described in Section V-A. The MGTSP solution to (P2) can be constructed from the TSP solution to (P3) only if Lemma V.3 is satisfied.

Combining the methods described by Noon and Bean and the lemmas stated in this section, we can transform an MGTSP to a TSP, solve it, and use the TSP solution to re-construct the original MGTSP solution. Our final result can be formally stated as the *Modified Noon-Bean Theorem.*

Theorem V.4 (Modified 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 $\sum_{e \in E_{\Upsilon^3}} c^3(e) < (R+2)\beta$.*

VI. Persistent Recharging In Extended Planning Horizons

To this point, we have restricted our problem formulation to a single recharge per working robot. For persistent surveillance tasks, however, it is necessary to consider multiple recharging events per working robot to maintain functionality over longer planning horizons. One approach is to formulate the recharging problem as a path optimization over the entire planning horizon, known as a *fixed horizon* plan. Computing an optimal fixed horizon path may be intractable for larger problem sizes and an alternative approach to reduce computational effort is to consider an iterative computation of the single recharge cycle plan over a receding planning horizon. We first formulate the fixed horizon plan as a MILP to generate an optimal recharge schedule over a finite planning horizon and then present the *receding horizon* planning approach as an extension of the single recharge cycle problem.

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. In this section, we formally state the *optimal periodic recharging* problem and present a MILP solution, which extends the approach in Section IV for a single recharge cycle.

As in the single recharge cycle computation, in the periodic recharging problem, working robot trajectories are known for the entire planning range $[0, T]$. However, the objective is now 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 over the planning range. Our approach to the problem is as follows.

1) Approach: 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.

Given these considerations, we formulate the periodic charging problem as an optimization on a partitioned graph, G, for a set of working robots $\mathcal{R}$ and a set of charging robots $\mathcal{M}$ in an environment $\mathcal{E} \subset \mathbb{R}^2$. The graph G is defined 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.

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 8.

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 $v_i \in V_r$.

2) Periodic MILP Formulation: Given the problem statement, the periodic charging MILP can be defined as an extension to the single charge cycle MILP 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.

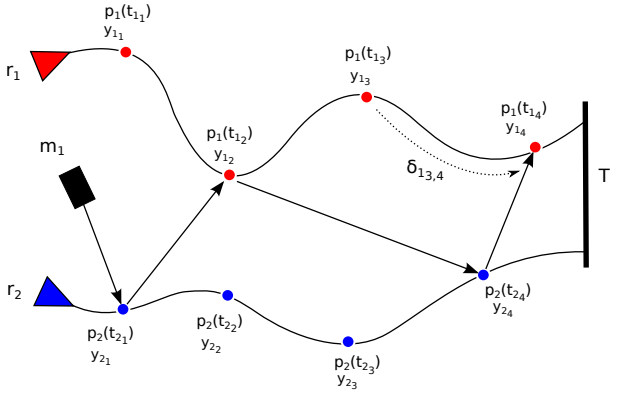


Fig. 8: The periodic MILP representation: An sample problem instance illustrating charging point discretization and key variables. A path for charging robot m_1 is computed to visit charging point sets V_{r_1} and V_{r_2}, for robots r_1 and r_2, periodically to ensure $y_{r,i} < \tau_r$.

The objective of the periodic charging problem, as inherited from the single recharge cycle MILP, 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 [32].

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

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 MILP 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 MILP 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.

B. Receding Horizon Planning

Receding horizon methods have been extensively applied to MILP based motion planning [15] 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 planning event, for the duration of the planning horizon.

In the persistent recharging scenario, we maintain a set of R working robots waiting to be charged. At each planning iteration, a time horizon $[0, T_r]$ is defined using the current state of the robots as their starting state. A One-in-a-set DAG problem is defined over the R working robots and a set of rendezvous paths for charging robots are computed using the single recharge cycle method of either Section IV or Section V. Each time a working robot is charged (i.e., a rendezvous occurs), we have an option to re-plan. Thus, given a team of R working robots, the receding horizon window can be varied from to R rendezvous' between re-plans.

The size of this window influences the quality of solutions generated. A larger planning window results in fewer planning iterations, a lower cumulative computation time and a lower total path cost per recharge cycle. However, it suffers from a greater possibility that a subsequent planning iteration will produce an infeasible path problem. A smaller planning window, while producing a more myopic path and a larger number of planning iterations, is more robust to uncertainties and is less likely to reach an infeasible solution.

In Section VII-B, we examine the effects of the planning window on the performance, and compare this method to the multiple charge MILP.

VII. Simulation Results

The optimization framework for this paper was implemented and tested in simulated experiments generated in MATLAB®. 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 [19]. 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.

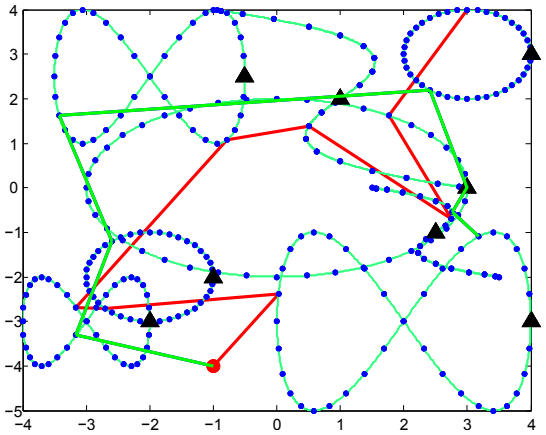
The simulation environment consists of a test set of planar trajectories that are assigned to a team of R working robots. Each working robot r is defined by its assigned trajectory, current pose, voltage level and battery lifetime T_r. The environment also contains a set of M randomly located charging 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.

Using these simulations, we benchmark rendezvous path solutions and examine the performance of the receding horizon and fixed horizon strategies for persistent recharging.

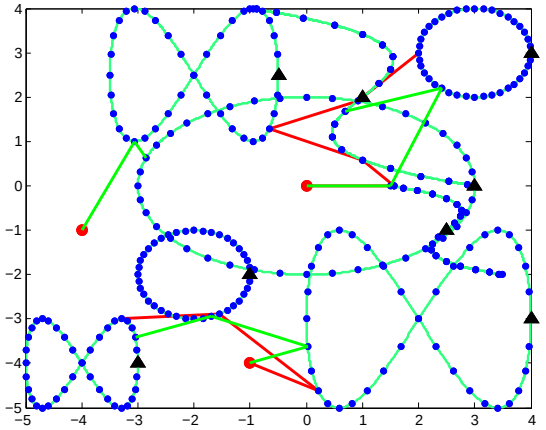
A. Single Recharge Cycle Path Computation

The recharge path is algorithmically computed using the Noon-Bean Transformation in the case of a single charging robot and the Modified Noon-Bean Transformation for multiple charging robots.

Figure 9(a) illustrates a sample problem instance with 8 working robots distributed along 8 paths and 1 charging robot. The result compares the heuristic solution obtained with the Noon-Bean transformation and LKH solver against the optimal



(a) Comparison of the optimal CPLEX solution (light grey/green path) against the Noon bean Transform and LKH Heuristic solution (dark grey/red path). The problem consists of 8 working robots (triangles) on 8 paths.



(b) 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 8 working robots (triangles) on 8 paths and 3 charging robots.

Fig. 9: Rendezvous path computation for a single recharge cycle.

MILP solution computed with CPLEX, for a single charging robot. The generated DAG contains 500 vertices. The optimal solution for a single charging robot was computed by CPLEX in 102 seconds. The heuristic solution was computed by LKH in 2 seconds and resulted in a path cost 12.8% higher than the optimal cost.

Figure 9(b) presents a sample problem instance with eight working robots distributed among eight paths and 3 charging robots. The result compares the heuristic solution obtained with the Modified Noon-Bean transformation followed by the LKH solver against the optimal MILP solution for multiple charging robots. The DAG 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.

The results demonstrate that the performance of the Modified Noon-Bean transformation closely matches the original Noon-Bean method as an algorithmic strategy in comparison with the optimal MILP solutions.

To further benchmark the performance of the LKH heuristic against the optimal CPLEX solution, we conducted an experiment to examine the effect of growth in problem complexity 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 Figure 10 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.

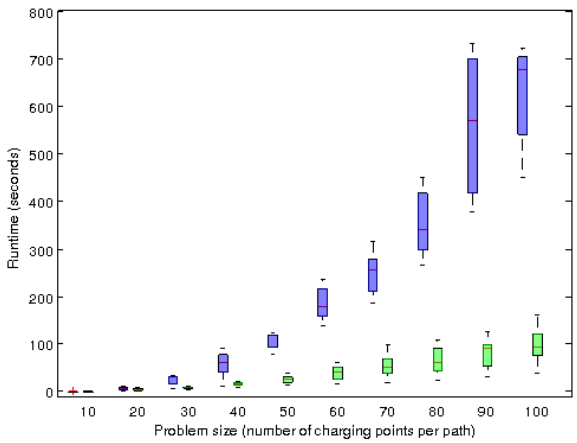
Figure 10(a) demonstrates the the growth in runtime for the optimal CPLEX solution and the LKH heuristic solution with a growth in problem complexity. Similarly, Figure 10(b) demonstrates the trend in path costs for the optimal and heuristic solutions. The optimal path cost for each simulation environment 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.

Finally, since an optimal MILP solution is not computationally tractable for large problem sizes an extremely large problem was solved as a performance benchmark. The environment consists of sixteen working robots, evenly distributed among eight paths and one charging robot. The resulting graph for the TSP solution contains 5761 vertices. The LKH solver found a TSP solution in 368 seconds. The optimal MILP solver was not able to compute an solution within a reasonable time frame.

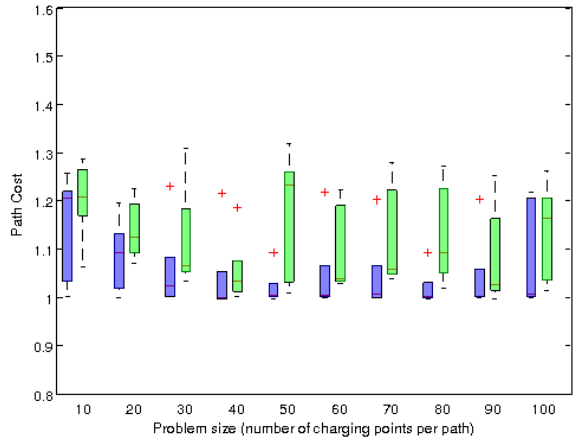
B. Persistent recharging in extended planning horizons

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 MILP solution at each planning iteration and compared with the optimal fixed horizon path over the planning horizon.

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 MILP 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, to examine the effect of growth in problem complexity on the runtime and solution quality for both methods.



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



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

Fig. 10: Performance comparison of Optimal CPLEX and LKH TSP heuristic solutions

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 11 compares the normalized path

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

costs for both methods with a box plot.

The results show that, on average, as problem complexity 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 MILP solution at each iteration. On average, the receding horizon method is seen to produce solutions with a total path cost within 20% of the optimal fixed horizon solution.

Next, we investigate the effect of varying the planing

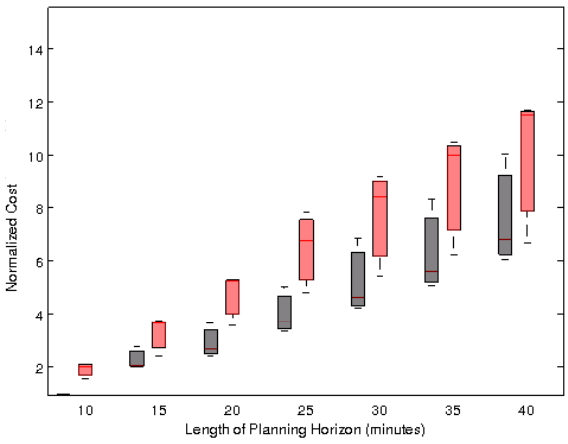


Fig. 11: Total path cost: Fixed horizon (dark/black) and Receding horizon (light/red)

TABLE III: Receding horizon cumulative runtime

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

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

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.

VIII. Conclusions and Future Work

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

We prove that the One-in-a-set DAG problem is NP-hard and present a MILP 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.

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 [33]. 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* [34] 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. Casbeer, R. Beard, T. McLain, S.-M. Li, and R. Mehra, "Forest fire monitoring with multiple small UAVs," in *American Control Conference, 2005. Proceedings of the 2005*, 2005, pp. 3530–3535 vol. 5.

[4] B. White, A. Tsourdos, I. Ashokaraj, S. Subchan, and R. Zbikowski, "Contaminant cloud boundary monitoring using network of UAV sensors," *Sensors Journal, IEEE*, vol. 8, no. 10, pp. 1681–1692, 2008.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Y. Litus, R. T. Vaughan, and P. Zebrowski, "The frugal feeding problem: Energy efficient, multi-robot, multi-place rendezvous," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 27–32.

[14] K. J. Obermeyer, P. Oberlin, and S. Darbha, "Sampling-based path planning for a visual reconnaissance unmanned air vehicle," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 2, pp. 619–631, 2012.

[15] 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.

[16] T. Schouwenaars, J. How, and E. Feron, "Receding horizon path planning with implicit safety guarantees," in *American Control Conference, 2004. Proceedings of the 2004*, vol. 6, 2004, pp. 5576–5581 vol.6.

[17] 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.

[18] 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.

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

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

[21] 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.

[22] ——, "A Lagrangian based approach for the asymmetric generalized traveling salesman problem," *Operations Research*, vol. 39, no. 4, pp. 623–632, 1991.

[23] C.-M. Pintea, P. C. Pop, and C. Chira, "The generalized traveling salesman problem solved with ant algorithms," *Journal of Universal Computer Science*, vol. 13, no. 7, pp. 1065–1075, 2007.

[24] D. Karapetyan and G. Gutin, "Linkernighan heuristic adaptations for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 208, no. 3, pp. 221 – 232, 2011.

[25] L. V. Snyder and M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 174, no. 1, pp. 38 – 53, 2006.

[26] J. Yang, X. Shi, M. Marchese, and Y. Liang, "An ant colony optimization method for generalized {TSP} problem," *Progress in Natural Science*, vol. 18, no. 11, pp. 1417 – 1422, 2008.

[27] 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.

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

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

[30] C. E. Noon and J. C. Bean, "A Lagrangian based approach for the asymmetric generalized traveling salesman problem," *Operations Research*, vol. 39, no. 4, pp. pp. 623–632, 1991.

[31] G. Laporte, H. Mercure, and Y. Nobert, "An exact algorithm for the asymmetrical capacitated vehicle routing problem," *Networks*, vol. 16, no. 1, pp. 33–46, 1986.

[32] J. Hooker and M. Osorio, "Mixed logical-linear programming," *Discrete Applied Mathematics*, vol. 9697, no. 0, pp. 395 – 442, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166218X99001006

[33] 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.

[34] "A MILP framework for batch reactive scheduling with limited discrete resources," *Computers & Chemical Engineering*, vol. 28, no. 67, pp. 1059 – 1068, 2004, ¡ce:title¿FOCAPO 2003 Special issue¡/ce:title¿.