Risk-aware Resource Allocation for Multiple UAVs-UGVs Recharging Rendezvous

Ahmad Bilal Asghar,[1*] Guangyao Shi,[1*] Nare Karapetyan,[1] James Humann,[2]
Jean-Paul Reddinger,[2] James Dotterweich,[2] Pratap Tokekar[1]

***Abstract*— We study a resource allocation problem for the cooperative aerial-ground vehicle routing application, in which multiple Unmanned Aerial Vehicles (UAVs) with limited battery capacity and multiple Unmanned Ground Vehicles (UGVs) that can also act as a mobile recharging stations need to jointly accomplish a mission such as persistently monitoring a set of points. Due to the limited battery capacity of the UAVs, they sometimes have to deviate from their task to rendezvous with the UGVs and get recharged. Each UGV can serve a limited number of UAVs at a time. In contrast to prior work on deterministic multi-robot scheduling, we consider the challenge imposed by the stochasticity of the energy consumption of the UAV. We are interested in finding the optimal recharging schedule of the UAVs such that the travel cost is minimized and the probability that no UAV runs out of charge within the planning horizon is greater than a user-defined tolerance. We formulate this problem (Risk-aware Recharging Rendezvous Problem (RRRP)) as an Integer Linear Program (ILP), in which the matching constraint captures the resource availability constraints and the knapsack constraint captures the success probability constraints. We propose a bicriteria approximation algorithm to solve RRRP. We demonstrate the effectiveness of our formulation and algorithm in the context of one persistent monitoring mission.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) are increasingly being used in applications such as surveillance [1], [2], package delivery [3], [4], environmental monitoring [5], [6], and precision agriculture [7] due to their ability to monitor large areas in a short period of time. One bottleneck that hinders long-term deployments of UAVs is their limited battery capacity. Recently, there have been efforts in overcoming this bottleneck by using Unmanned Ground Vehicles (UGVs) as mobile recharging stations [8]–[18]. However, these works make the simplifying but restrictive assumption that the energy consumption of the UAV is deterministic. In practice, the energy consumption is stochastic and the planning algorithms must be able to deal with the risk of running out of charge. In our recent work [9], we presented a risk-aware planning algorithm for planning for a single UAV and UGV. However, that algorithm scales exponentially with the planning horizon and the number of robots. In this paper, we present an efficient risk-aware coordination algorithm for scalable, long-term UAV-UGV missions.

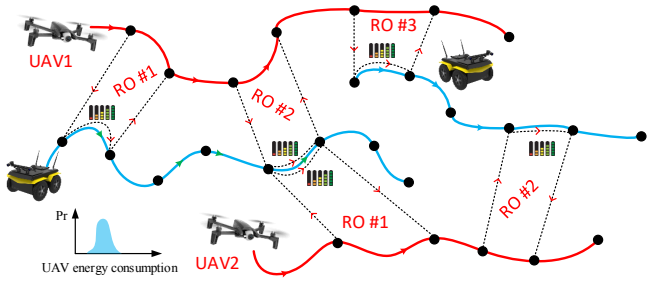


Fig. 1. An illustrative example of the recharging rendezvous problem considered in this paper. A team of two UAVs and two UGVs is executing their tasks in the given order. The UAVs need to decide when and where to land on the UGVs in order to recharge minimizing the total detour time. The energy consumption of the UAVs is stochastic. RO stands for rendezvous option.

We consider a scenario where the UAVs and the UGVs are executing a persistent monitoring mission by visiting a sequence of *task* nodes in the order given (Figure 1). The UAVs can take a detour from the planned mission to rendezvous with some UGV and recharge. The UGV can recharge the UAV while continuing towards its own next task node. We present several algorithms that decide *when* and *where* the UAVs should recharge and *which* UGV they should recharge on.

The rate of battery discharge of a UAV is stochastic in the real world. The planning algorithms should be able to deal with such stochasticity. One approach would be to take recharging detours more frequently, thereby reducing the risk of running out of charge[1]. However, if a UAV takes a detour, then the task nodes visited after the detour will experience a delay, thereby worsening the task performance. Since we are planning over a longer horizon with multiple UAVs and UGVs, there is a complex trade-off between task performance and risk tolerance.

Given a team of UGVs that move on the road network, and UAVs that can directly fly between any pair of nodes, we are interested in finding a scheduling strategy for the UAVs by treating UGVs as resources such that the sum of detour cost of the UAVs is minimized and the probability that no UAV runs out of charge within the planning horizon is greater than a user-defined tolerance. We formulate this

This work is supported in part by National Science Foundation Grant No. 1943368 and Army Grant No. W911NF2120076.

* indicates equal contribution and authors are listed alphabetically

[1]University of Maryland, College Park, MD 20742 USA gyshi, knare, abasghar, tokekar@umd.edu

[2]DEVCOM Army Research Laboratory, MD USA jean-paul.f.reddinger.civ, james.d.humann.civ, james.m.dotterweich.civ@army.mil

[1]In practice, when a UAV is about to run out of charge, it can land and then be retrieved by a human. By reducing the risk of running out of charge, we aim to reduce the overhead of such costly human interventions.

problem as a resource allocation problem with matching and knapsack constraints. The matching constraint captures the resource availability constraints (each UGV can serve a limited number of UAVs at a time). The knapsack constraint captures the risk-tolerance requirement (probability that no UAV runs out of charge).

The Risk-aware Recharging Rendezvous Problem (RRRP) can be formulated as an integer linear program (ILP). While there are sophisticated ILP solvers, we show how to exploit the structural properties of the problem to yield more efficient algorithms with theoretical performance guarantees. Other combinatorial problems with knapsack or budget constraints have been studied in literature. Approximation algorithms for the budgeted version of the minimum spanning tree problem and maximum weight matching problem are presented in [19] and [20] respectively. They both use Lagrangian relaxation which resembles our approach, however, due to different problem structures, we propose a bicriteria approximation algorithm to solve RRRP. We demonstrate the effectiveness of our formulation and algorithm in the context of a persistent monitoring mission.

Our work is built on results from the previous work on the cooperative routing problems, which are usually formulated as variants of the Vehicle Routing Problem (VRP) or the Traveling Salesman Problem (TSP) [3], [4], [8], [10], [12], [21]–[23]. In these works, authors usually assume a deterministic model for the energy consumption of the UAV or use the expected value for simplification. However, when the UAV executes the route, the actual energy consumption may be quite different from that used in route planning. Moreover, in the long-duration task, the estimation module may periodically update the energy consumption model. Given the routes of UAVs and UGVs generated by some existing algorithms, we consider the recharging scheduling problem within a planning horizon. Our formulation can be used in a receding horizon fashion to deal with the uncertainties in the long-duration tasks. More details will be discussed in the Sec IV.

The main contributions of this paper are:

- We introduce the Risk-aware Recharging Rendezvous Problem (RRRP) as the first risk-aware version of the multi-robot recharging problem. We formulate the RRRP in the cooperative aerial-ground system as a matching problem on a bipartite graph with an additional knapsack constraint.
- We present several algorithms to solve the problem, that build on each other. Our main theoretical result is a $(1+\epsilon, 2)$ bicriteria approximation algorithm that yields a solution that violates one of the constraints by a factor of 2, and has a detour cost within $1+\epsilon$ of the optimal cost.
- We demonstrate the effectiveness of our formulation and the proposed algorithm in a persistent monitoring application.

II. Finite Horizon Risk-Aware Recharging Rendezvous Problem

In this section we first define the Risk-aware Recharging Rendezvous Problem on a finite horizon, and then formulate the RRRP as a matching problem on a bipartite graph with an additional knapsack constraint which enforces that the probability of no UAV running out of charge within the horizon is greater than a risk-tolerance parameter.

A. Problem Statement

Consider a team of N_a UAVs and N_g UGVs persistently monitoring a set of locations in an environment. The UAVs and UGVs move on the graphs $G_a = (U_a, E_a)$ and $G_g = (U_g, E_g)$ respectively. The vertex sets U_a and U_g represent the locations to be monitored by the aerial and ground vehicles respectively. The edge set E_a may be complete since the UAVs can move between any of the tasks, whereas the edge set E_g represents the road network on which the ground vehicles can move. We assume that the ordering of the task nodes for the UAVs and the UGVs is given, i.e., we have pre-defined persistent monitoring tours for the UAVs and UGVs. These tours can be generated by planners that either do not consider recharging [24]–[26] or assume deterministic discharge [12], [14]. The tour for i^{th} UAV is denoted by $\mathcal{T}_i^a$ and the tour for UGV k is denoted by $\mathcal{T}_k^g$. Because of the persistent nature of the problem, the UAVs need to be recharged repeatedly in order for them to keep operating.

A UAV i can leave its monitoring tour $\mathcal{T}_i^a$ at any point along the tour to land on one of the UGVs, say UGV k, for recharging, while the UGV keeps moving along its tour $\mathcal{T}_k^g$. The UAV can also wait at a rendezvous location for the UGV if it reaches there before the UGV. The number of UAVs that can simultaneously charge on a given UGV is d. Once recharged, the UAV leaves the UGV and goes to the next task node along its monitoring tour. This procedure of a UAV leaving its monitoring tour to rendezvous with a UGV for recharging and then returning to continue its monitoring tour is referred to as a *detour*.

We assume that the UGVs do not run out of charge.[2] Unlike much of the existing work [10]–[12], we do not assume that the discharge rate of by the UAVs is deterministic. Instead, we consider a stochastic discharge model and assume that the probability of a UAV running out of charge within the next t time units given the current charge level can be calculated as that in [9]. The stochastic battery discharge is monotonic in the time traveled by the UAV.

Since it is a persistent monitoring scenario where the UAVs need to be recharged indefinitely, we take the approach of solving the problem in a receding-horizon manner. The problem within a given time window T is to find a recharging policy for the UAVs for up to a single recharge for each UAV. Moreover, as the battery discharge rate of UAVs is stochastic, there may be a non-zero probability of some UAVs not being

[2]This is a standard assumption in the literature [8], [10]–[12], [22], [27] since the runtime of the UGV can be an order of magnitude larger than the UAV. Furthermore, the UGV battery (or fuel, if its fuel powered) can be easily and quickly replenished unlike the UAV.

able to reach any of the recharging UGVs. Hence, we also need to have a notion of risk-aversion for the UAVs. On the other hand, to avoid frequent recharging, we also consider the detour time spent by the UAVs for recharging.

At a high level, we consider the following problem.

Given a time horizon T, a risk-tolerance probability $\rho \in (0,1)$, task routes for the N_a UAVs and N_g UGVs along with their current locations and state of charge, we seek to solve the problem of finding a recharging schedule for each UAV such that:

1) *each UAV recharges at most once during T,*
2) *no UGV can charge more than d UAVs at a time,*
3) *the probability that no UAV runs out of charge during T is at least ρ, and*
4) *the total detour time of the UAVs is minimized.*

In the following paragraphs, we provide more details regarding the Finite Horizon Recharging Rendezvous Problem and then propose an algorithm to solve the problem in the next section.

B. Integer Linear Program Formulation

We now formally define the setup of the problem. Since, a UAV can decide at any point along its tour to leave the path in order to rendezvous with a UGV at any point along the UGVs tour, we have a continuous constrained optimization problem. We discretize the tours of the UAVs and UGVs in order to get a combinatorial optimization problem. Given the tour $\mathcal{T}_k^g$ for UGV k, we discretize the tour by introducing vertices every f distance starting from the UGV's current position where f is the maximum distance the UGV needs to travel while a UAV completes its charging. These vertices representing possible rendezvous locations are denoted by $V(\mathcal{T}_k^g)$ and are shown as green discs in Figure 2. Similarly the set $V(\mathcal{T}_i^a)$ represents the set of locations from which UAV i can leave its monitoring tour $\mathcal{T}_i^a$ for a recharging rendezvous with a UGV. The set $V(\mathcal{T}_i^a)$ contains the current position of UAV i and its task nodes that can be visited within next T time by UAV i. We do not need to further discretize UAVs tours as it can be shown (Lemma 6) that there exists an optimal solution where the UAVs will leave their tours for recharging from either their current position or a task node. In Figure 2, the vertices in $V(\mathcal{T}_i^a)$ are shown as red discs. In the scenario shown in Figure 2 the UAV 1 chooses to leave its task route at the node $a_1^1 \in V(\mathcal{T}_1^a)$ in order to rendezvous with UGV 1 at vertex $g_1^3 \in V(\mathcal{T}_1^g)$. The recharging is complete when the UGV reaches node g_4^1 and the UAV moves to the next task node a_1^2 in its tour $\mathcal{T}_i^a$. Note that if $d = 1$, no other UAV can recharge on this UGV between g_1^3 and g_1^4.

We now define the problem formally on a bipartite graph $G = (V_a \cup V_g, E)$, defined as follows.

- *UAV vertices:* The vertex set V_a consists of N_a disjoint node sets, i.e., $V_a = \cup_{i=1}^{N_a} \mathcal{V}_i$ where $\mathcal{V}_i = V(\mathcal{T}_i^a) \cup a_i^\emptyset$. The vertex $a_i^\emptyset$ represents the scenario where the UAV i chooses not to rendezvous in the current horizon.

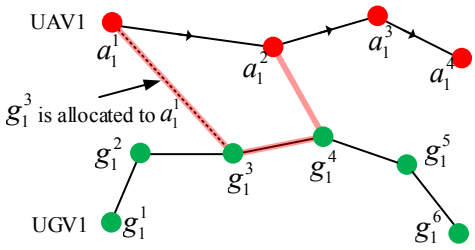


Fig. 2. Example of a recharging detour. The UAV 1 leaves its task route at the node a_1^1 to rendezvous with UGV 1 at the node g_1^3. The recharging is done when they reach the node g_1^4 and the UAV moves to the next task node a_1^2.

- *UGV vertices:* The vertex set V_g consists of $\{g_1^\emptyset, \ldots, g_i^\emptyset, \ldots, g_{N_a}^\emptyset\}$ and d copies of the set $\cup_{k=1}^{N_g} V(\mathcal{T}_k^g)$. The vertex $g_i^\emptyset$ for UAV i represents the scenario where UAV i chooses not to rendezvous for the current horizon. The d copies of each vertex in $V(\mathcal{T}_k^g)$ are to represent up to d different UAVs recharging at a time on a UGV.[3]
- *Edges:* The edge set E denotes the set of all feasible recharging detour options. If the UAV i, starting from a node $j \in V(\mathcal{T}_i^a)$, is able to rendezvous with the UGV k at $l \in V(\mathcal{T}_k^g)$, an edge exist between the corresponding two nodes in V_a and V_g. The vertex $a_i^\emptyset$ is connected to $g_i^\emptyset$ for all $i \in [N_a]$.
- *Edge cost:* For an edge $(i,j) \in E$ where $i \in V_a$ and $j \in V_g$, the edge cost c_{ij} represents the total time needed to finish the task route given that the recharging detour along edge (i,j) is taken. The time needed for the recharging detour consists of three parts: the time to reach the rendezvous node, the waiting time and the recharging time at the rendezvous node, and the time to go to the next node on the tour. The edge cost along edge $(a_i^\emptyset, g_i^\emptyset)$ is zero.
- *Edge success probability:* For an edge $(i,j) \in E$ where $i \in V_a$ and $j \in V_g$, the edge success probability p_{ij} is defined as the overall probability to finish the task route for the current horizon given that the recharging detour along edge (i,j) is taken. It is the product of the two success probabilities: the probability of a successfully completing the recharging detour, and the probability of finishing the rest of the route after recharging. The probability along edge $(a_i^\emptyset, g_i^\emptyset)$ is the probability of reaching the end of the current horizon without recharging.

One example of the constructed bipartite graph with $d = 1$ is shown in Figure 3.

The finite horizon recharge rendezvous problem can now be defined as the following Integer Linear Program (ILP).

Problem 1 (Risk-aware Recharging Rendezvous Problem (RRRP)). *Given a bipartite graph $G = (V_a \cup V_g, E)$ where*

[3]We use d copies of each vertex in $V(\mathcal{T}_k^g)$ to keep the analysis of the algorithm simple. Note that the problem can be defined without having d copies for each vertex in $V(\mathcal{T}_k^g)$ by changing Constraint (4) in Problem 1 to $\sum_i x_{ij} \leq d$.

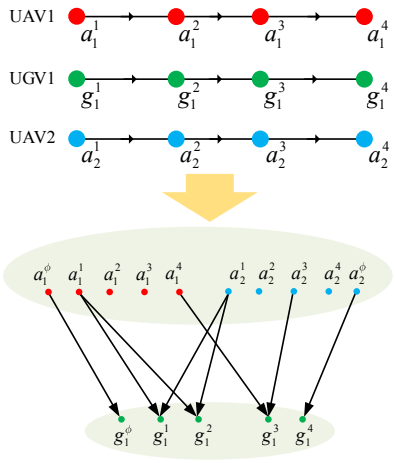


Fig. 3. An example to show how to construct a bipartite graph for one planning horizon from the UAV and UGV route segments.

$V_a = \cup_{i=1}^{N_a} \mathcal{V}_i$, with edge costs c_{ij} and probabilities p_{ij} for edge $(i,j) \in E$, solve:

$$\min \sum_{i,j} c_{ij} x_{ij} \tag{1}$$

$$s.t. \sum_{i,j} \log \frac{1}{p_{ij}} x_{ij} \leq \log \frac{1}{\rho} \tag{2}$$

$$\sum_{i \in \mathcal{V}_r} \sum_{j} x_{ij} = 1, \qquad \forall r \in [N_a] \tag{3}$$

$$\sum_{i} x_{ij} \leq 1, \qquad \forall j \in V_g \tag{4}$$

$$x_{ij} \in \{0,1\} \tag{5}$$

The decision variable x_{ij} indicates whether edge (i,j) is in the solution or not. The objective is to minimize the travel time overhead incurred by the recharging detours. Inequality (2) enforces that the probability that no UAV runs out of charge during the current horizon is at least ρ. We can write this as a linear constraint since the stochastic discharging processes of the UAVs are independent. For ease of notation, we will use the following inequality instead of Constraint (2).

$$\sum_{i,j} a_{ij} x_{ij} \leq B \tag{6}$$

where $B = \log 1/\rho$ and $a_{ij} = \log 1/p_{ij}$. Constraint (3) enforces that each UAV should be recharged at most once. Constraint (4) enforces that each UGV can recharge at most d UAVs at a time (by making sure that at most one UAV recharges at one of the d copies of a UGV vertex). Given a solution x_l, let M_l represent the corresponding solution on graph G. Let us define $c(M) = c(x) = \sum c_{ij} x_{ij}$, $a(M) = a(x) = \sum a_{ij} x_{ij}$ for ease of notation.

Remark 1. In some applications, it might be desirable to have a constraint on the total detour time while maximizing the probability of robots not running out of charge. Note that although we define the problem with sum of detour times as the objective and with a constraint on probability, the discussion and results in this paper hold with this constraint and the objective swapped as well.

C. Hardness

We now characterize the hardness of Problem 1 in the following result.

Lemma 1. *Problem 1 is NP-hard.*

Proof. We show the hardness by using a reduction from the problem EVENODDPARTITION which is shown to be NP-complete in [28].

In an instance of EVENODDPARTITION, we are given a finite set $\{z_1, \ldots, z_{2\ell}\}$ of positive integers, and the decision problem is to find whether there exists a subset $I \subseteq \{1, \ldots, 2\ell\}$ such that $\sum_{i \in I} z_i = \sum_{i \notin I} z_i$ and $|I \cap \{z_{2j-1}, z_{2j}\}| = 1$ for each $j \in \{1, \ldots, \ell\}$.

Given an instance of EVENODDPARTITION, we construct an instance of Problem 1 as follows. Consider a bipartite graph $G = (V_1 \cup V_2, E)$ where V_1 consists of one vertex for each $j \in \{1, \ldots, \ell\}$ and V_2 consists of one vertex each for $j \in \{1, \ldots, 2\ell\}$. Connect vertex $j \in V_1$ to vertex $2j \in V_2$ and to vertex $2j-1 \in V_2$. Let us denote these edges by e_{2j} and e_{2j-1} respectively. The cost and weight on edge e_{2j}, i.e., $c(e_{2j})$ and $a(e_{2j})$ are z_{2j} and z_{2j-1} respectively. The cost and weight on edge e_{2j-1} is z_{2j-1} and z_{2j} respectively. Exactly one outgoing edge from each vertex in V_1 should be selected in the solution. Finally we set the value of maximum allowed weight (equivalent to B in Constraint (6)) to $Z = \frac{1}{2} \sum_{i=1}^{2\ell} z_i$.

If the instance of EVENODDPARTITION is a YES instance, then there exists a solution in the bipartite graph G such that each vertex in V_1 has exactly one outgoing edge in the solution and the sum of weights on that solution is Z. Moreover, by the construction of G, the sum of costs of that solution is also G.

If the instance of EVENODDPARTITION is a NO instance, then either the weight or the cost on any solution that has exactly one outgoing edge from each $j \in V_1$, is more than Z. Since the solutions with a weight of more than Z are infeasible, the minimum cost of any feasible solution is more than Z.

Therefore, if the optimal cost in the constructed bipartite graph is more than Z, then the corresponding EVENODDPARTITION is a NO instance, and it is a YES instance otherwise. ∎

We present a bicriteria approximation algorithm for Problem 1 in this paper. The main result regarding the algorithm is summarized below and we provide a detailed analysis in the next section.

Theorem 1. *For a given $\epsilon \in (0,1]$, there is a $(1+\epsilon, 2)$-bicriteria approximation algorithm for Problem 1, i.e., we get a solution x such that*

- $c(x) \leq (1+\epsilon)\mathtt{opt}$, *and*
- $\sum_i x_{ij} \leq 2, \quad \forall\, j \in V_g$.

Note that the approximation algorithm gives a solution where one of the UGV's may have to recharge at most $d+1$

Algorithm 1 BINARYSEARCH

Input: Graph $G = (V_a \cup V_g, E)$ with edge costs c and weights a, bound B

Set a threshold $\Delta\lambda_{\min}$
$\lambda_l \leftarrow 0$, $\lambda_l \leftarrow \infty$
Start from a positive value of λ
while $\lambda_u - \lambda_l \geq \Delta\lambda_{\min}$ **do**
    Solve Problem 2 using λ to get solution x
    if $a(x) \leq B$ **then**
        $M_1 \leftarrow x$
        $\lambda_u \leftarrow \lambda$
        $\lambda \leftarrow (\lambda_u + \lambda_l)/2$
    else
        $M_2 \leftarrow x$
        $\lambda_l \leftarrow \lambda$
        $\lambda \leftarrow \min\{2\lambda, \lambda_u\}$
return M_1, M_2, λ

Algorithm 2 LOCALSEARCH

Input: Graph $G = (V_a \cup V_g, E)$ with edge costs c and weights a, bound B

Get λ, M_1 and M_2 such that $a(M_1) \leq B \leq a(M_2)$ using Algorithm 1 or [20]
$M' \leftarrow M_1 \oplus M_2$
for connected components Y in M' **do**
    if M' has one connected component **then**
        return M_1, M_2, λ
    else
        if $a(M_1 \oplus Y) \leq B$ **then**
            $M_1 \leftarrow M_1 \oplus Y$
        else
            $M_2 \leftarrow M_1 \oplus Y$
    Remove Y from M'

UAVs at a time, and since $d \geq 1$, it results in violation of Constraint (4) by a factor of two as mentioned in Theorem 1.

III. ALGORITHMS AND ANALYSIS

Although Problem 1 can be solved using ILP solvers, as the number of variables in the problem increases, the runtime of the solver will become intractable. Since we need to solve this problem repeatedly in a receding horizon approach, an efficient solver is desirable. In this section, we present a series of algorithms to solve Problem 1, that build on top of each other. Algorithm 1 is a heuristic algorithm that uses binary search on the Lagrangian multiplier to find a feasible solution. Algorithm 2 uses Algorithm 1 to provide two solutions that are adjacent in the solution polytope, one of them being feasible. A bicriteria approximation algorithm is presented in Algorithm 3, which uses Algortithm 2 as a subroutine. The bicriteria approximation algorithm may violate one of the constraints of the problem (one of the UGVs may need to recharge up to $d+1$ UAVs at a time). In practice, we can use Algorithm 3 to solve the problem and if the solution violates the constraint, we can use the solution returned by Algorithm 2.

We start the analysis of our algorithms by considering the following problem :

Problem 2.

$$\min \sum c_{ij}x_{ij} + \lambda(\sum_{i,j} a_{ij}x_{ij} - B), \tag{7}$$

such that Constraints (3)*,* (4) *and* (5) *hold.*

Let $w_\lambda(M) = w_\lambda(x) = c(x) + \lambda a(x)$. Also, let $\mathcal{S}$ denote the polytope formed by the set of constraints (3) and (4). We first show that Probelm 2 is solvable in polynomial time.

Lemma 2. *Problem 2 is solvable in* $O(|E|\log|V|(|E| + |V|\log(V))$ *time on graph* $G(V_a \cup V_b, E)$ *where* $V = V_a \cup V_b$.

Proof. We can solve Problem 2 using a minimum cost network flow problem. Given the bipartite graph $G = (V_a \cup V_g, E)$, introduce N_a nodes $v_1, \ldots, v_{N_a}$. Connect the vertices in $\mathcal{V}_i$ to v_i and connect all vertices $v_1, \ldots, v_{N_a}$ to a source vertex s. Connect all the vertices in V_g to a sink vertex q. The cost of all the edges $(i, j) \in E$ is set to $c_{ij} + \lambda a_{ij}$. The cost of all other edges is zero. The capacity of all edges is one and N_a amount of flow is to be sent from s to q. This instance has $O(|E|)$ edges and $O(|V_a| + |V_b|)$ vertices. A solution to this minimum cost network flow problem will satisfy Constraints (3) and (4). Since the total flow and capacities are integers, there exist integer solutions satisfying Constraint (5), that can be found using network flow algorithms such as Orlin's algorithm [29] which runs in $O(|E|\log|V|(|E| + |V|\log(V))$ time on a graph with $|V|$ vertices and $|E|$ edges. ■

Since Problem 2 is solvable in polynomial time, the following observation enables us to use binary search in Algorithm 1 to find two solutions M_1 and M_2 such that $a(M_1) \leq B \leq a(M_2)$.[4]

Observation 1. *Let* x_1 *and* x_2 *be the optimal solutions to Problem 2 with Lagrangian multipliers* λ_1 *and* λ_2*, where* $\lambda_1 > \lambda_2$*. Then* $a(x_1) \leq a(x_2)$*, and* $c(x_1) \geq c(x_2)$*.*

Although the run time of Algorithm 1 depends on the stopping threshold value $\Delta\lambda_{\min}$, an optimal Lagrangian multiplier λ and two solutions x_1 and x_2 to Problem 2, with $w_\lambda(x_1) = w_\lambda(x_2)$ and satisfying $a(x_1) \leq B \leq a(x_2)$ and $c(x_1) \geq c(x_2)$ can be found in polynomial time [20], [30, Theorem 24.3]. We can use the procedure from [20] and [30, Theorem 24.3] or Algorithm 1 in Line 1 of Algorithm 2 to find M_1, M_2 and λ. We use Algorithm 1 in experiments to find M_1 and M_2 as it is simple to implement and works well in practice as seen in Section IV. Since $w_\lambda(M_1) \leq w_\lambda(M^*)$ where M^* is the optimal solution to Problem 1 with cost

[4]Note that if no solution M_2 exists such that $a(M_2) > B$, then solving Problem 2 with $\lambda = 0$ solves Problem 1.

Algorithm 3 RENDEZVOUSSCHEDULING

Input: Graph $G = (V_a \cup V_g, E)$ with edge costs c and weights a, bound B, $\epsilon \in (0, 1]$

1: $M_H^* \leftarrow$ Guess $p = \lceil \frac{1}{\epsilon} \rceil$ edges of highest cost in M^*
2: $E \leftarrow E \setminus \{e : c(e) \geq \min_{e' \in M_H^*} c(e')\}$
3: **for** $(i, j) \in M_H^*$ **do**
4: $\quad V_a \leftarrow V_a \setminus \{\mathcal{V}_r : i \in \mathcal{V}_r\}$
5: $\quad V_g \leftarrow V_g \setminus \{j\}$
6: $B \leftarrow B - a(M_H^*)$
7: Find M_1, M_2 and λ from Algorithm 2
8: $Z \leftarrow M_1 \oplus M_2$
9: **for** edge z_i in $Z = \{z_0, z_1, \ldots, z_{k-1}\}$ **do**
10: $\quad$ **if** $z_i \in M_1$ **then** $\alpha_i = w_\lambda(z_i)$
11: $\quad$ **if** $z_i \in M_2$ **then** $\alpha_i = -w_\lambda(z_i)$
12: Find z_i such that $\sum_{j=i}^{i+h} \alpha_{i \pmod k} \leq 0$ for all h
13: Find the longest sequence Z'' starting from z_i such that $a(M_1 \oplus Z') \leq B$
14: **if** Last edge of Z'' is not in M_2 **then**
15: $\quad$ Remove the last edge of Z''
16: $M_L \leftarrow M_1 \oplus Z''$
17: **if** M_L has two edges connected to $\mathcal{V}_r$ for some r **then**
18: $\quad$ Remove one of those edges from M_L
19: **if** M_L has two edges (i_1, j) and (i_2, j) for a $j \in V_g$ **then**
20: $\quad$ **if** $\exists$ free j' such that $a(i_\ell, j') \leq a(i_\ell, j)$ **then**
21: $\quad\quad M_L \leftarrow M_L \setminus \{(i_\ell, j)\} \cup \{(i_\ell, j')\}$
22: **return** $M_L \cup M_H^*$

opt, we have,

$$\begin{aligned} w_\lambda(M_1) - \lambda B &\leq w_\lambda(M^*) - \lambda B \\ &\leq w_\lambda(M^*) - \lambda a(M^*) \\ &= c(M^*) = \texttt{opt}. \end{aligned} \tag{8}$$

We will need the following definition for the analyses of Algorithms 2 and 3.

Definition 1. Given the bipartite graph $G = (V_a \cup V_g, E)$ where $V_a = \cup_{i=1}^{N_a} \mathcal{V}_i$, consider the bipartite graph G' with the vertices for each robot merged together, i.e., $G' = (V_c \cup V_g, E')$ where $V_c = \{\mathcal{V}_1, \ldots, \mathcal{V}_{N_a}\}$, and E' has an edge between $\mathcal{V}_r$ and $j \in V_g$ if E has an edge between some $i \in \mathcal{V}_r$ and j. Then with abuse of notation, we define a subgraph H of G as a *connected component* if the edges of H form a connected path or cycle in G'.

Now we show a property of adjacent extreme points in the solution polytope $\mathcal{S}$ similar to a property of graph matchings shown in [31].

Lemma 3. *Two vertices x_1 and x_2 of $\mathcal{S}$ are adjacent if and only if the symmetric difference of their corresponding solutions M_1 and M_2 contain exactly one connected component.*

Proof. The symmetric difference of M_1 and M_2 is a union of connected components (paths and cycles) because each edge in the symmetric difference can have a degree at most two. Suppose that $M_1 \oplus M_2$ contains exactly one connected component. Then we can construct an objective function (edge costs) such that M_1 and M_2 are the only two optimal extreme point solutions as follows. Let the number of edges in $M_1 \oplus M_2$ from M_1 and M_2 be h_1 and h_2 respectively. Assign a cost of one to all edges in $(M_1 \cap M_1 \oplus M_2)$, a cost of h_1/h_2 to all edges in $(M_2 \cap M_1 \oplus M_2)$, a cost of zero to all edges in $M_1 \cap M_2$, and a negative cost to all other edges in the graph. Then M_1 and M_2 are the only two extreme point solutions in $\mathcal{S}$ for this cost function.

Now suppose that the symmetric difference has at least two connected components. Let Z be one of those connected components and let $M_3 = M_1 \oplus Z$ and $M_4 = M_2 \oplus Z$. M_3 and M_4 are also solutions in $\mathcal{S}$ and

$$\frac{1}{2}(x_1 + x_2) = \frac{1}{2}(x_3 + x_4).$$

So if there is any objective function for which M_1 and M_2 are optimal solutions, then the mid point also has the same cost, and hence M_3 or M_4 also has the same cost. Therefore, M_1 and M_2 are not adjacent extreme points in $\mathcal{S}$. ■

A procedure to find two adjacent extreme point solutions in the solution polytope for a maximum weight matching problem is given in [20]. We use a similar procedure for our problem to find two adjacent solutions in $\mathcal{S}$ in Algorithm 2. We have the following result regarding Algorithm 2.

Lemma 4. *Algorithm 2 returns λ and two solutions M_1, M_2, such that they correspond to adjacent extreme points in $\mathcal{S}$ and*

- $w_\lambda(M_1) = w_\lambda(M_2)$,
- $a(M_1) \leq B \leq a(M_2)$,
- $c(M_1) \geq c(M_2)$.
- $c(M_1) \leq \texttt{opt} + \lambda B$

Proof. If $M' = M_1 \oplus M_2$ consists of more than one connected component, we can pick one of those connected components, say Y, and let $N = M_1 \oplus Y$. If $a(N) \leq B$, replace M_1 by N. Otherwise replace M_2 by N. We can repeat this step until $M1 \oplus M2$ consists of only pone connected component. Note that at each step, $|M_1 \cap M_2|$ increases by at least one, so this procedure stops in at most N_a steps. Moreover, M_1 always remains feasible, and by the optimality of M_1 and M_2 with respect to w_λ, $w_\lambda(M_1)$ and $w_\lambda(M_2)$ do not change during this process. The last inequality follows from the definition of $w_\lambda(M)$ and (8). ■

The following result shows that given M_1, M_2 and λ from Algorithm 2, we can get a solution M that may violate at most one constraint and the cost of M is within $c_{\max}$ of the optimal, where $c_{\max}$ is the largest edge cost in G. Lines 8 to 22 of Algorithm 3 find such a solution.

Lemma 5. *Let M^* be the optimal solution to Problem 1 with cost opt. There is a polynomial time algorithm that finds a scheduling M (corresponding to a solution x) such that*

- $c(M) \leq \texttt{opt} + c_{max}$, *and*

- *one of the UGVs may have to recharge at most $d+1$ UAVs at a time.*

Proof. From Lemma 4 and Lemma 3 we have two solutions M_1 and M_2 such that $w_\lambda(M_1) = w_\lambda(M_2)$, and $M_1 \oplus M_2$ contains exactly one connected component $Z = (z_0, z_1, \ldots, z_{k-1})$. Consider the sequence

$$\alpha_0 = \delta_0 w_\lambda(z_0), \alpha_1 = \delta_1 w_\lambda(z_1), \ldots, \alpha_{k-1} = \delta_{k-1} w_\lambda(z_{k-1}),$$

where $\delta_i = 1$ if $z_i \in M_1$ and $\delta_i = -1$ if $z_i \in M_2$. Since $w_\lambda(M_1) = w_\lambda(M_2)$, $\sum_i \alpha_i = 0$. Moreover there exists an edge $z_i, i \in \{0, 1, \ldots, k-1\}$, such that for any cyclic subsequence $(z_i, z_{(i+1) \pmod k}, \ldots, z_{(i+h) \pmod k})$, $\sum_{j=i}^{i+h} \alpha_{i \pmod k} \leq 0$ [20, Lemma 3]. Therefore,

$$\sum_{e \in Z \cap M_2} w_\lambda(e) - \sum_{e \in Z \cap M_1} w_\lambda(e) = \sum_{j=i}^{i+h} \alpha_{i \pmod k} \leq 0. \quad (9)$$

Consider the smallest such subsequence Z' such that $a(M_1 \oplus Z') \geq B$. Assume that Z' contains at least two edges, otherwise $c(M_1) \leq c(M_2) + c_{\max}$ and we are done. Note that removing one edge from the end of Z' will mean that Z' is the longest sequence such that $a(M_1 \oplus Z') \leq B$. Since the connected component is alternating, remove at least one and at most two edges from the end of the sequence Z' to get Z'' such that the last edge in Z'' belongs to M_2. Also note that by construction, the first edge in Z'' also belongs to M_2. Therefore the first and last edges from Z'' will appear in $M = M_1 \oplus Z''$ and M may result in more than one edge connected to some set $\mathcal{V}_i$ or some $j \in V_g$. If one of the robots has more than one connected edges in M (because the first and last edges in Z'' are from M_2), we remove one of those edges. Note that M can have one more edge connected to a UGV location as compared to M_1, i.e., $\sum_i x_{ij} \leq 2$. Note that this can happen at one of the two ends of Z'' as it is alternating. So one of the UGVs may have to recharge at most $d+1$ UAVs at a time. Also note that $a(M_1 \oplus Z'') \leq B$. Now we lower bound the value of $c(M)$. Since we remove at most one edge from $M_1 \oplus Z''$ to get M,

$$c(M) \leq c(M_1 \oplus Z'') \leq c(M_1 \oplus Z') + c_{\max},$$

where the second inequality is due to the fact that we remove at most two edges from Z' to get Z''.

Now,

$$\begin{aligned} c(M_1 \oplus Z') &= w_\lambda(M_1 \oplus Z') - \lambda a(M_1 \oplus Z') \\ &= w_\lambda(M_1 \oplus Z') - \lambda B - \lambda(a(M_1 \oplus Z') - B) \\ &\leq w_\lambda(M_1) - \lambda B - \lambda(a(M_1 \oplus Z') - B) \\ &\leq \texttt{opt} - \lambda(a(M_1 \oplus Z') - B) \\ &\leq \texttt{opt}, \end{aligned}$$

where the first inequality is due to (9) and the second last inequality is due to (8). Hence

$$c(M) \leq \texttt{opt} + c_{\max}.$$

■

Corollary 1. *If each UAV can recharge at atleast N_a different UGV recharging locations, there is a polynomial time algorithm that finds a feasible scheduling M satisfying Constraints* (3) *and* (4) *such that $a(M) \leq B + 2a_{max}$ and $c(M) \leq \texttt{opt} + 3c_{max}$.*

Now we prove the main result regarding Algorithm 3.

Proof of Theorem 1. Given $\epsilon \in (0, 1]$, guess $p = \lceil 1/\epsilon \rceil$ edges with highest cost value of c_{ij} in the optimal solution M^*. Let these edges be M_H^*. Remove M_H^* and all the edges with costs higher than the lowest cost in M_H^* from the graph. Also for $(i, j) \in M_H^*$ where $i \in \mathcal{V}_r$, remove the vertices $\mathcal{V}_r$ from V_a and j from V_g. Also decrease B by $a(M_H^*)$. Then if the optimal solution to the resulting instance is M_L^*, $M_H^* \cup M_L^*$ is the optimal solution to the original problem. The maximum cost of an edge in the resulting instance will be at most $c(M_H)/p \leq \epsilon c(M_H)$. Then by Lemma 5, we get a solution M_L for the new instance such that

$$\begin{aligned} c(M_H^*) + c(M_L) &\leq c(M_H^*) + c(M_L^*) + \epsilon c(M_H^*) \\ &\leq c(M^*) + \epsilon c(M^*) \leq (1+\epsilon)\texttt{opt}. \end{aligned}$$

Since one of the UGVs may have to recharge at most $d+1$ UAVs at a time, and because $d \geq 1$, we get a $(1+\epsilon, 2)$-bicriteria approximation. The algorithm requires $O(N_a^{1/\epsilon})$ guesses for M_H^*. ■

IV. Experimental Results

In this section, we first present a qualitative example of the persistent monitoring mission. Next, we study how system parameters (various risk tolerances) influence the recharging behaviors between the UAVs and the UGVs and the task performances of the UAVs. Then, we compare the performance of our scheduling strategy with a baseline (greedy strategy) using the mean time before the first failure and the travel distance overhead as metrics. Moreover, we empirically evaluate the performance of the proposed heuristic algorithm. All experiments in Section IV-B are conducted using Python 3.8 on a PC with the i9-8950HK processor. The baseline solver is Gurobi 9.5.0.

A. Experimental Setup

We consider a team consisting of two UAVs and two UGVs. The task routes $\mathcal{T}_a$ and $\mathcal{T}_g$ used in the problem can be either generated jointly by some task planners similar to those in [10], [27] or can be generated separately by different task planners. In our case study, the task of two UGVs is to persistently monitor the road nodes. The setup here is similar to our previous work [9] on Intelligence, Surveillance, and Reconnaissance (ISR) where the focus is on improving high-level solutions.

The UAV and UGV move at 9.8 m/s and 4.5 m/s respectively based on the field test data [9] in our ongoing project. The recharging process (swapping battery) takes 100 s. The UAV and UGV need to persistently monitor the task nodes on the route. We apply our recharging strategy in a receding

horizon fashion: every two minutes, the UAVs-UGVs team solves the RRRP problem to decide the UAVs' recharging schedule for the next $T = 2500$ seconds. For each UAV, the current position will be the first node when we construct the bipartite graph. If some UAV is on a detour, we do not replan until the UAV has finished its detour.

We consider two sources of stochasticity in the energy consumption model of UAVs: weight and wind velocity contribution to longitudinal steady airspeed. The deterministic energy consumption model of the UAV is a polynomial fit constructed from analytical aircraft modeling data, given as,

$$P(\boldsymbol{v_\infty}) = b_0 + b_1\boldsymbol{v_\infty} + b_2\boldsymbol{v_\infty^2} + b_3\boldsymbol{v_\infty^3} + b_4\boldsymbol{w} + b_5\boldsymbol{v_\infty}\boldsymbol{w}, \quad (10)$$

where b_0 to b_5 are coefficients, and their experimental values are listed in Table I.

TABLE I
COEFFICIENTS FOR STOCHASTIC ENERGY CONSUMPTION MODEL

	b_0	b_1	b_2	b_3	b_4	b_5
Value	-88.77	3.53	-0.42	0.043	107.5	-2.74

Weight is randomly selected following a normal distribution with a mean of 2.3 kg and a standard deviation of 0.05 kg, $\boldsymbol{w} \sim \mathcal{N}(\mu_{\boldsymbol{w}}, \sigma^2_{\boldsymbol{w}})$. Vehicle airspeed, v_∞, is the sum of the vehicle ground speed, v, and the component of the wind velocity that is parallel to the vehicle ground speed, ignoring sideslip angle and lateral wind components.

$$v_\infty = |\bar{v_g} + \cos(-\psi)\boldsymbol{\xi}_{\mathrm{a,b}}| \quad (11)$$

The longitudinal wind speed contribution is derived from two random parameters; wind speed, and wind direction. Wind speed is modeled using the Weibull probability distribution model of wind speed distribution, $\boldsymbol{\xi}_{a,b}$, with a characteristic velocity $a = 1.5$ m/s and a shape parameter $b = 3$. This is representative of a fairly mild steady wind near ground level. Wind direction ψ is the heading direction of the wind, and is uniformly randomly selected on a range of $[0, 360)$ degrees.

B. Simulation Results

a) Qualitative Example: An illustrative example of the input and the output of the problem considered is shown in Figure 4. The input of the problem is shown in Figure 4a, which consists of UAV task nodes (red dots), and nodes of the road network (blue nodes). Figure 4b shows one tour route of one UAV when the system executes the proposed strategy in a receding horizon fashion. The UAV monitors the task route persistently. When the UAV reaches node a, it doesn't move forward to its next task node (connected through a dashed red line). Instead, the new schedule is to rendezvous with UGV at a_s and takes off from the UGV at a_t, and then go to its next task node. Similarly, the UAV will rendezvous with the UGV when it is close to nodes b, c, d and e. Subscriptions s and t denote the start and the terminal of the recharging process. A sample of the history of the state of charge (SOC) is shown in Figure 4c. We can observe in Figure 4c that the UAV's recharging strategy is more than a simple rule (for example get recharged when the SOC is below 50 $\%$) and may get recharged at various values of SOC.

TABLE II
STATISTICAL RESULTS FOR UAVS

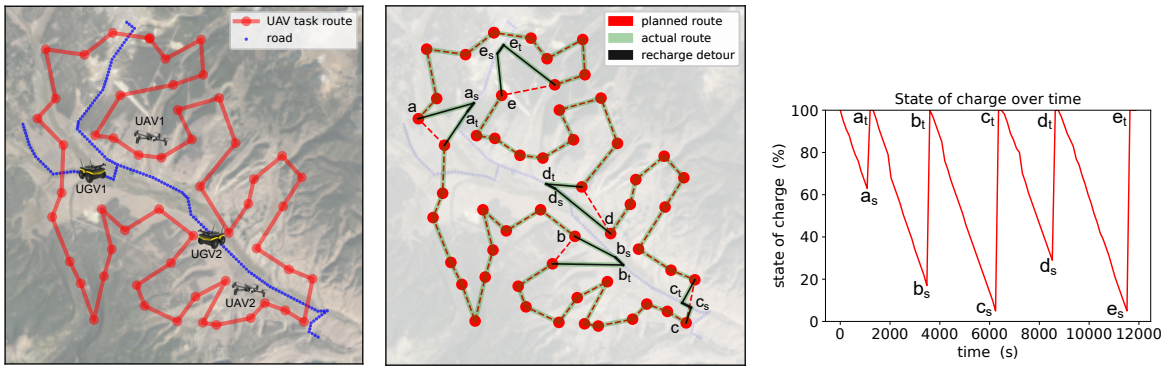
UAV data	$\rho = 0.01$	$\rho = 0.1$	$\rho = 0.3$
Avg. travel time before out of charge (s)	39660	27600	24360
Avg. travel time overhead	19.7 %	18.5 %	17.8 %
Avg. # of task nodes visited	158	110	105
Avg. # of rendezvous per $T = 2500$ s	1.4	1.3	1.3

b) Effect of Risk Tolerance: Next, we show how different risk tolerances influence the recharging behaviors under our RRRP formulation. In these experiments, we set the risk tolerance ρ to be 0.01, 0.1, and 0.3. Some statistical data are summarized in table II. We use four metrics to quantify the performance of the strategy:

- Mean time before the first failure here failure refers to the case when one UAV in the team is out of charge and needs human intervention. This quantity reflects how frequent the system needs human involved and we expect this quantity to be large enough.
- Travel time overhead this quantity is computed as
$$\frac{\text{actual travel time} - \text{task time}}{\text{task time}},$$
where task time is the time of the route obtained when we project the actual flight route into the planned route. This quantity reflects how well the UAV is performing its task and we expect this quantity to reasonably large.
- Avg. # of task nodes visited the average number of task nodes visited by the UAV before its first failure.
- Average number of rendezvous per planning horizon T if this number is too large, it suggests that the UAV is scheduled too many recharging detours, which should be avoided.

In general, we observe that when the risk tolerance is set to be smaller, the mean time before the first failure will be longer. Similarly, the travel time overhead and the average number of rendezvous per planning horizon will be greater, which implies the UAV spends more portion of flight time in the recharging detours.

c) Comparison of Algorithm 1 with Baseline: To validate that the scheduling strategy constructed from RRRP, we compare our strategy with a greedy baseline. The greedy policy is set as: choose to rendezvous when state-of-charge drops below a set value. We consider three set values 30%, 40%, and 50%, and the corresponding strategies are denoted as *Greedy-30*, *Greedy-40*, and *Greedy-50*. In this experiment, we set $\rho = 0.1$. The first observation is that Algorithm 1 achieves close performances in both metrics compared to



(a) (b) (c)

Fig. 4. A qualitative example to illustrate how UAV and UGV rendezvous with each other under the proposed scheduling strategy that is obtained by solving the RRRP. The risk tolerance is set to be $\rho = 0.1$ in this case study. Subscriptions s and t denote the start and the terminal of the recharging process. (a) The input of the RRRP problem including the UAV and UGV tasks and the road network. (b) One sample tour of UAV 1 when it persistently monitors the route. (c) One sample history of SOC for UAV 1.

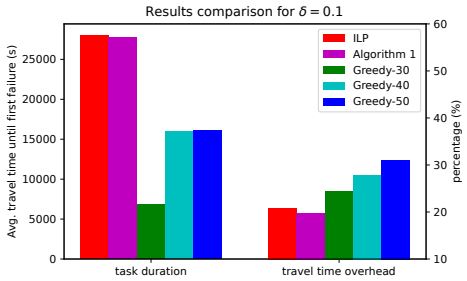


Fig. 5. Results comparisons for the RRRP scheduling strategy and the greedy strategies with $\rho = 0.1$. ILP and Algorithm 1 refer to the solution returned by the ILP solver and Algorithm 1 by solving RRRP.

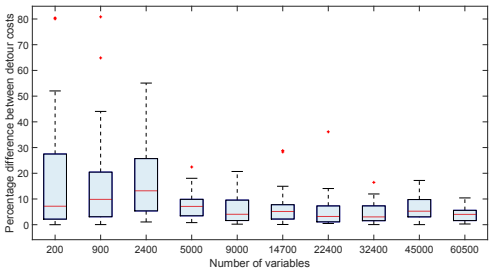


Fig. 6. Percentage increase in the objective function of Algorithm 2 as compared to an ILP solver. The boxplot shows the result of 20 experiments for each problem size.

that of ILP. Second, as shown in Figure 5, our strategy (obtained by both ILP and Algorithm 1) can achieve longer travel time before first failure on average (left group) and a relatively lower travel distance overhead (right group), which implies that our strategy will not lead too many unnecessary rendezvous. Moreover Algorithm 1 has a better travel time overhead than that of ILP although ILP solves RRRP optimally for a given horizon. This is likely due to solving RRRP repeatedly in a receding horizon manner.

d) Scalability of Algorithm 2: We also compare the performance of Algorithm 2 with an ILP solver empirically. We used Algorithm 2 for comparison instead of Algorithm 3 as Algorithm 2 and the ILP always return a feasible solution, making the objective value comparison fair. The ILP solver used for this set of experiments is `intlinprog` function from MATLAB, and Algorithm 2 was also implemented in MATLAB for fair comparison. Since ILP solves Problem 1 optimally, the cost returned by Algorithm 2 is at least that of ILP solver. The percentage difference in the objective function values for different problem sizes is shown in Figure 6. For each problem size, represented by the number of edges or variables, twenty random problem instances were created and the boxplot of resulting objective value difference is shown in the plot. On average, among all the instances, Algorithm 2 was within 15% of the optimal solution. Note that the performance of Algorithm 2 improves as the number of variables increases.

Figure 7 shows the average runtime comparison between Algorithm 2 and the ILP solver. Note that the y-scale is logarithmic. For smaller problem instances, both the algorithms solved the problem within a second, with ILP being faster, however, as the number of variables increases, ILP becomes much slower, with the runtime for ILP being up to seven times more than that of Algorithm 2 for 60500 variables. Note that there may be other solvers for ILP that have better run time, but since ILP is NP-complete, the exponential gap between run times is likely to continue as the number of variables increases.

V. Conclusion

In this paper, we study a resource allocation problem with matching and knapsack constraints for the UAVs-UGVs recharging rendezvous problem. We formulate this problem (Risk-aware Recharging Rendezvous Problem (RRRP)) as one Integer Linear Program (ILP) and propose one bicriteria approximation algorithm to solve RRRP. We validate our formulation and the proposed algorithm in one persistent monitoring application. In our current formulation, we assume that the task routes of vehicles are given. In future

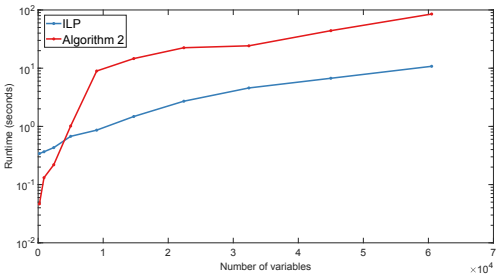


Fig. 7. Comparison of runtimes of Algorithm 2 and ILP solver. Note that the y-axis is logarithmic.

work, one direction that we will explore is to propose efficient and scalable routing algorithms to generate task routes for vehicles.

REFERENCES

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

[2] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas, "Cooperative air and ground surveillance," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 16–25, 2006.

[3] C. Lin, "A vehicle routing problem with pickup and delivery time windows, and coordination of transportable resources," *Computers & Operations Research*, vol. 38, no. 11, pp. 1596–1609, 2011.

[4] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 86–109, 2015.

[5] J. Diaz, E. Corrales, Y. Madrigal, D. Pieri, G. Bland, T. Miles, and M. Fladeland, "Volcano monitoring with small unmanned aerial systems," in *Infotech@ Aerospace 2012*, 2012, p. 2522.

[6] Y. Sung, D. Dixit, and P. Tokekar, "Environmental hotspot identification in limited time with a uav equipped with a downward-facing camera," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 264–13 270.

[7] H. Dhami, K. Yu, T. Xu, Q. Zhu, K. Dhakal, J. Friel, S. Li, and P. Tokekar, "Crop height and plot estimation for phenotyping from unmanned aerial vehicles using 3d lidar," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2643–2649.

[8] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[9] G. Shi, N. Karapetyan, A. B. Asghar, J.-P. Reddinger, J. Dotterweich, J. Humann, and P. Tokekar, "Risk-aware uav-ugv rendezvous with chance-constrained markov decision process," *2022 IEEE 61th Annual Conference on Decision and Control (CDC)*, 2022.

[10] K. Yu, A. K. Budhiraja, and P. Tokekar, "Algorithms for routing of unmanned aerial vehicles with mobile recharging stations," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 5720–5725.

[11] N. Mathew, S. L. Smith, and S. L. Waslander, "Multirobot rendezvous planning for recharging in persistent tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 128–142, 2015.

[12] P. Maini, K. Sundar, M. Singh, S. Rathinam, and P. Sujit, "Cooperative aerial–ground vehicle route planning with fuel constraints for coverage applications," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 55, no. 6, pp. 3016–3028, 2019.

[13] P. Maini and P. Sujit, "On cooperation between a fuel constrained uav and a refueling ugv for large scale mapping applications," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2015, pp. 1370–1377.

[14] K. Sundar and S. Rathinam, "Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 287–294, 2013.

[15] L. Liu and N. Michael, "Energy-aware aerial vehicle deployment via bipartite graph matching," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2014, pp. 189–194.

[16] H. Y. Jeong, B. D. Song, and S. Lee, "Truck-drone hybrid delivery routing: Payload-energy dependency and no-fly zones," *International Journal of Production Economics*, vol. 214, pp. 220–233, 2019.

[17] A. Karak and K. Abdelghany, "The hybrid vehicle-drone routing problem for pick-up and delivery services," *Transportation Research Part C: Emerging Technologies*, vol. 102, pp. 427–449, 2019.

[18] H. Li, J. Chen, F. Wang, and M. Bai, "Ground-vehicle and unmanned-aerial-vehicle routing problems from two-echelon scheme perspective: A review," *European Journal of Operational Research*, vol. 294, no. 3, pp. 1078–1095, 2021.

[19] R. Ravi and M. X. Goemans, "The constrained minimum spanning tree problem," in *Scandinavian Workshop on Algorithm Theory*. Springer, 1996, pp. 66–75.

[20] A. Berger, V. Bonifaci, F. Grandoni, and G. Schäfer, "Budgeted matching and budgeted matroid intersection via the gasoline puzzle," *Mathematical Programming*, vol. 128, no. 1, pp. 355–372, 2011.

[21] Y. Liu, Z. Luo, Z. Liu, J. Shi, and G. Cheng, "Cooperative routing problem for ground vehicle and unmanned aerial vehicle: The application on intelligence, surveillance, and reconnaissance missions," *IEEE Access*, vol. 7, pp. 63 504–63 518, 2019.

[22] S. G. Manyam, D. W. Casbeer, and K. Sundar, "Path planning for cooperative routing of air-ground vehicles," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 4630–4635.

[23] S. G. Manyam, K. Sundar, and D. W. Casbeer, "Cooperative surveillance in the presence of time sensitive data," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 343–348.

[24] N. Nigam and I. Kroo, "Persistent surveillance using multiple unmanned air vehicles," in *2008 IEEE Aerospace Conference*. IEEE, 2008, pp. 1–14.

[25] A. B. Asghar, S. L. Smith, and S. Sundaram, "Multi-robot routing for persistent monitoring with latency constraints," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 2620–2625.

[26] S. Hari, S. Rathinam, S. Darbha, K. Kalyanam, S. Manyam, and D. Casbeer, "Efficient computation of optimal uav routes for persistent monitoring of targets," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 605–614.

[27] S. G. Manyam, K. Sundar, and D. W. Casbeer, "Cooperative routing for an air–ground vehicle team—exact algorithm, transformation method, and heuristics," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 537–547, 2019.

[28] D. S. Johnson and M. R. Garey, *Computers and intractability: A guide to the theory of NP-completeness*. WH Freeman, 1979.

[29] J. Orlin, "A faster strongly polynomial minimum cost flow algorithm," in *Proceedings of the Twentieth annual ACM symposium on Theory of Computing*, 1988, pp. 377–387.

[30] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[31] M. Balinski and A. Russakoff, "On the assignment polytope," *Siam Review*, vol. 16, no. 4, pp. 516–525, 1974.

APPENDIX

Lemma 6. *Given metric travel costs for the UAVs, there exists an optimal solution to Problem 1 where the UAVs will only leave their tours for recharging from either their current position or a task node.*

Proof. Suppose there exist an optimal solution with cost $c*$ where UAV i has to leave its tour $\mathcal{T}_i^a$ from a point p, that is not its current location or a task node, to rendezvous with a UGV at location g. Let v_1 be the task node that precedes p in $\mathcal{T}_i^a$, or the current position of UAV i, if p is between the current position and first task node). Also let v_2 be the task node following p in $\mathcal{T}_i^a$. Then using the triangle inequality, the total travel time for the edges (v_1, g) and (g, v_2) is not more than the the total travel time on edges $(v_1, p), (p, g)$ and (g, v_2). Moreover, since battery discharge depends on

the time traveled by the UAV, the probability of the UAV not running out of charge for the detour taken from v_1 is not less than the probability of the UAV running out of charge for the detour taken from p. Hence, we get a solution where the UAV leaves its tour from its current position or a task node, with a cost at most c^*. ■