The Persistent Robot Charging Problem for Long-Duration Autonomy

Nitesh Kumar[1], Jaekyung Jackie Lee[1], Sivakumar Rathinam[2], Swaroop Darbha[2], P.B. Sujit[3] and Rajiv Raman[4]

Abstract— This paper introduces a novel formulation aimed at determining the optimal schedule for recharging a fleet of n heterogeneous robots, with the primary objective of minimizing resource utilization. This study provides a foundational framework applicable to Multi-Robot Mission Planning, particularly in scenarios demanding Long-Duration Autonomy (LDA) or other contexts that necessitate periodic recharging of multiple robots. A novel Integer Linear Programming (ILP) model is proposed to calculate the optimal initial conditions (partial charge) for individual robots, leading to the minimal utilization of charging stations. This formulation was further generalized to maximize the servicing time for robots given adequate charging stations. The efficacy of the proposed formulation is evaluated through a comparative analysis, measuring its performance against the thrift price scheduling algorithm documented in the existing literature. The findings not only validate the effectiveness of the proposed approach but also underscore its potential as a valuable tool in optimizing resource allocation for a range of robotic and engineering applications.

I. INTRODUCTION

The UAV industry is experiencing rapid expansion, and this technological advancement is immediately felt in everyday life. An increasing number of companies, such as Amazon, Google, UPS, FedEx, and DHL, are actively integrating UAVs into their operations for last-mile delivery [1], [2]. Despite their wide usage, UAVs face limitations in flight time due to their limited battery capacity. For example, a heavy-duty drone like the Prism Lite, which uses 16,000mAh batteries, can achieve only 40 minutes of flight time without any external payload [3]. In Long Duration Autonomy (LDA) and persistent monitoring applications, UAVs are expected to operate over long durations (typically, a few days) without human intervention and hence, this necessitates UAVs to frequently recharge their batteries at a charging station while carrying out their tasks.

We assume, without any loss of generality, that no more than one UAV may be recharged at a charging station at any time. Hence, scheduling UAVs for recharging at the charging station becomes pivotal from an operational efficiency viewpoint. Improper scheduling could lead to multiple UAVs running out of charge at the same time. This

can happen if any two UAVs have identical charging and flying times and start with the same charge; clearly, the determination of the initial battery charge becomes crucial for staggering/scheduling their deployment, especially in LDA applications requiring persistent monitoring and data collection. Since charging stations are valuable resources that come with significant costs, reducing their number can lead to substantial savings in setup expenses. Therefore, scheduling the charging and departures of UAVs at the charging stations will aid in efficiently utilizing the charging stations and reducing the overall cost.

In this paper, we divide time into uniform time slots (or simply slots) at the charging station; scheduling the charging station implies (a) assigning each time slot to at most one UAV while allowing a UAV to charge fully and (b) making the charging station available for recharging before it runs out of battery charge. We also specify the charging time (time to fully charge starting from no charge state) and the flying time as integer multiples of the time slots; moreover, we call the cycle time for a UAV to be the sum of its charging and flying times.

In this paper, we explore two problem variants: (a) scheduling the UAVs to minimize the number, m_{min} of charging stations, and (b) selecting and scheduling a subset of UAVs to maximize the total flying time of UAVs with given $m \leq m_{min}$ charging stations.

The first variant can be formulated as a generalized non-preemptive windows scheduling problem as discussed by [4], where the objective is to non-preemptively schedule all jobs (or charging UAVs fully before flying and arriving at the charging station before running out of charge) on the fewest possible number of parallel machines (charging stations). This problem has been proven to be NP-hard [4].

One can relax the generalized non-preemptive scheduling problem by requiring the UAV to arrive at the charging station only when it has completely run out of charge upon reaching the station. This is similar to the thrift price non-preemptive scheduling problem described by [4]. Notably, the algorithm proposed by [4] offers an 8-approximation solution to the problem. In this paper, we relax the first variant analogous to the thrift price scheduling problem and compare the results with the the 8-approximation approach. Then, we utilize the structure of the solution approach from the first variant to solve the second variant.

Some other existing work in literature such as [5], [6], [7] uses a game-theoretic approach to model the energy trading between UAVs and charging stations in a cost-optimal manner. Their proposed model allocates a time slot to the

[1] Doctoral student, J. Mike Walker' 66 Department of Mechanical Engineering, Texas A&M University, College Station, TX - 77843-3123. {niteshk, jkleecontrols}@tamu.edu
[2] Professor, J. Mike Walker '66 Department of Mechanical Engineering, Texas A&M University, College Station, TX {srathinam, dswaroop}@tamu.edu

UAVs using an auctioning process. The above algorithms are designed to work in open groups where there can be any number of participants without any prior knowledge of UAV's parameters such as its charging time and flying time. These methods do not guarantee that each UAV that needs a charging station will have a readily available charging station.

Other recent work on UAVs for LDA applications focuses on optimizing routes while taking into account the energy limitations of UAVs, assuming a sufficient number of available stationary or mobile charging stations. Reference [8], [9], [10] presents an algorithm to determine the sequence in which different sites and charging stations can be visited. Their algorithm plans tours for unmanned ground vehicles (UGVs) acting as mobile charging stations as well as determining the optimal locations to place stationary charging stations. References [11], [12] also discuss planning the paths for UAVs using multiple stationary recharging stations. Reference [13] developed an agent-based modeling framework for the multi-UAV rendezvous recharging problem, which consists of energy-limited aerial vehicles that rendezvous with a mobile or fixed charging station. Another recent work, outlined in [14], addresses a multi-robot persistent monitoring problem involving battery-limited robots. The goal is to determine the minimum number of robots needed to meet latency constraints while also ensuring that the robots periodically recharge at a designated depot. Developing an optimal charging schedule that minimizes the required charging stations can be regarded as a natural follow-on problem of the aforementioned studies.

The primary novelty and contribution of this paper lies in the development of scheduling strategies for the UAVs at the charging depot. Our first contribution is the introduction of a novel Integer Linear Programming (ILP)-based scheduling algorithm that leverages the cyclic charging and discharging behavior of UAVs and determines their optimal initial charge levels and time slots for charging. These levels are then used to establish a charging schedule, which minimizes the required charging resources at the depot. Second, in scenarios with limited charging stations, we propose a method for selecting and scheduling UAVs to maximize overall operational efficiency, specifically focusing on maximizing their total total flying time.

Since the application of our proposed method is not limited to UAVs alone, we will use the more general term "robots" for the remainder of the discussion.

II. PROBLEM STATEMENT

We address the challenge of scheduling charging slots for a fleet of n heterogeneous robots at a charging depot equipped with m number of distinct charging stations. The primary constraint is to ensure that each robot requiring charge obtains a charging station while avoiding running out of battery charge. Specifically, a robot must have an empty charging station ready after completing its mission; otherwise, it is not deployed. The operational parameters for each robot include the charging time and the operational

time. It is assumed that the specifications for each robot i.e., the operational time and the necessary charging time to recharge from empty to full are known beforehand.

One of the main requirements for efficient scheduling is to guarantee the availability of a charging station for each robot in need over an infinite time horizon. However, if we can pose our problem as a periodic problem then the scheduling problem is periodic with a cycle time, say T, then we need to satisfy the aforementioned constraint only until time T. Each robot is allocated consecutive charging slots equivalent in duration to its specified charging time. After charging fully, we assume the robot will require the charging station again exactly after its total operational time, establishing a periodic problem framework. Notably, partial charging and partial operation are not permissible, except during the initial deployment.

A feasible solution involves using at least as many charging stations as the number of robots, i.e. $m \geq n$; then, it guarantees the availability of a charging station for each robot when needed. However, that might not be an efficient usage of the charging stations because the corresponding charging station will be left unutilized whenever a robot is in operation. This setup raises two critical questions:

1) Does there exist a charging schedule that maximizes the resource utilization by minimizing the required resources m for a given set of n heterogeneous robots?
2) How can we construct an optimal schedule that maximizes the total flight time of the chosen set of robots from a fleet of n when limited by m resources?

In the next section, we will address the two questions raised above using this non-preemptive approach.

III. SCHEDULING FRAMEWORK

In subsection III-A, we discuss the periodic nature of scheduling and the associated scheduling horizon (period); we also propose a method to determine the cycle time that can serve as the scheduling horizon for optimizing the charging schedule. In subsection III-B, we delve into the methodology for determining the optimal charging schedule for each robot that minimizes the required charging stations. In subsection III-C, we outline a strategy for selecting robots from a given set and scheduling them optimally on limited charging stations to maximize their cumulative operational time across the scheduling horizon.

A. Finding a Scheduling Horizon

Let C and D respectively denote the sets of charging and flying times for the robots, i.e.,

$$C = \{c_1, c_2, \ldots, c_n\}, \ c_i \in \mathbb{Z}^+, \ i = 1, 2, \ldots, n, \quad (1)$$

$$D = \{f_1, f_2, \ldots, f_n\}, \ f_i \in \mathbb{Z}^+, \ i = 1, 2, \ldots, n. \quad (2)$$

The charging time c_i, and operational time f_i are assumed to be integers and are known a priori.

For each robot i, we consider a sequence of actions consisting of charging for c_i slots followed by discharging for f_i slots resulting in a cycle time $T_i = c_i + f_i$ slots. This

cycle time T_i represents the interval after which the robot returns to its initial state. Therefore, the process of charging and discharging is periodic for each robot with a cycle time T_i.

If each robot begins with an initial charge and operates on an individual cycle time T_i, and assuming there are enough charging stations available, all robots will return to their initial conditions simultaneously after T slots, where T represents the least common multiple (LCM) of the individual cycle times of the robots,

$$T = \text{LCM}(T_1, T_2, \ldots, T_n). \quad (3)$$

The non-preemptive approach simplifies the problem to a periodic problem, whose cycle time is given by Eq. (3), irrespective of the initial conditions of the robots. This information can be utilized to reduce the optimization problem from an infinite time horizon problem to the time with scheduling horizon T.

B. Minimizing resources

In this subsection, we present a novel formulation that will aid the determination of the initial conditions (charge) of the robots and find the corresponding minimum number of charging stations required over the scheduling horizon T. Here, the initial condition represents the amount of charge available with the robot and whether it is charging or in operation at the moment.

One can associate a time wheel as shown in Fig.1 that helps visualize the periodic evolution of charging and discharging of the robot's battery. Fig.1 depicts four charging slots and six operational slots with a cycle time of ten slots for a robot. The state evolves in a clockwise direction along the time wheel as time increases. For example, from Fig.1(a), we know that the robot has just begun charging at time t, and from Fig.1(b), it is in the second slot of charging at time $t+1$. The depiction of the number 1 in the time wheel indicates the state of the robot's battery and its status (charging/discharging).

(a) State at time t, illustrating the initiation of charging for this specific example.

(b) State at time $t+1$, showcasing the robot's completion of 1 slot of charging.

Fig. 1: Evolution of robot's states: Illustration of the charging (green) and discharging (red) phases as the robot transitions from one state to the next.

Fig. 2: State Transition Schematic: Depiction of the cyclic evolution of robot's states throughout the charging cycle. The transition initiates with charging, indicated by the green segment, progressing clockwise. Upon reaching full charge, the state transitions clockwise into the red segment.

Reflecting the state of the battery charge as depicted in the time wheel depicted above Fig.1, we associate a *binary cyclic state vector* $\mathbf{r_i(t)}$ of size T_i which contains exactly one component that is a "1". This constraint can be captured through the equation:

$$\mathbf{1}^{T_i}\mathbf{r}_i(t) = 1, \ t \geq 0, \quad (4)$$

where $\mathbf{1}$ is a *row* vector of the size T_i with all its components being 1. The periodic state transition of $\mathbf{r}_i(t)$ can be described by:

$$\mathbf{r_i(t) = A_i r_i(t-1)}, \quad t \geq 1 \quad (5)$$

where

$$\mathbf{A_i} = \begin{bmatrix} 0 & 0 & \ldots & 0 & 1 \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ldots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 0 \end{bmatrix}. \quad (6)$$

The state transition matrix A_i is a permutation matrix and is also circulant. The state vector $r_i(t)$ at time t can be used to determine whether the robot is recharging (and if so, it will a charging station is required for this purpose); this is done through a charging indicator vector $\mathbf{p_i^\top}$. The first c_i elements of the vector $\mathbf{p_i}$ are 1 and remaining f_i elements are 0; this vector captures the charging and discharging characteristics of the ith robot's battery. A binary decision variable $z_i(t)$ indicates whether the robot i is utilizing the resource (charging station) or not at time t through:

$$z_i(t) = \mathbf{p_i^\top r_i(t)}. \quad (7)$$

From this discussion, we can now formulate an Integer Linear Program (ILP) to find the initial condition $\mathbf{r_i(0)}$ to determine the minimum number, m_{min}, of charging stations for scheduling over scheduling horizon T. Mathematically,

$$m_{min} = \min \ m, \quad (8)$$

subject to the following constraints (4), (5) and

$$
\begin{bmatrix}
\sum_{i=1}^{n} z_i(1) \\
\sum_{i=1}^{n} z_i(2) \\
\vdots \\
\sum_{i=1}^{n} z_i(T)
\end{bmatrix}
=
\begin{bmatrix}
\mathbf{p_1^\mathsf{T}} & \cdots & \mathbf{p_n^\mathsf{T}} \\
\mathbf{p_1^\mathsf{T} A_1} & \cdots & \mathbf{p_n^\mathsf{T} A_n} \\
\vdots & \cdots & \vdots \\
\mathbf{p_1^\mathsf{T} A_1}^{T-1} & \cdots & \mathbf{p_n^\mathsf{T} A_n}^{T-1}
\end{bmatrix}
\begin{bmatrix}
\mathbf{r_1(0)} \\
\mathbf{r_2(0)} \\
\vdots \\
\mathbf{r_n(0)}
\end{bmatrix}
\leq
\begin{bmatrix}
m \\
m \\
\vdots \\
m
\end{bmatrix}.
\tag{9}
$$

The last set of constraints indicates that no more than m resources (charging stations) be utilized at any time.

The above-formulated problem ILP can be solved for the initial conditions $\mathbf{r_i(0)}$, and the minimum required resources m_{min}. Leveraging the obtained initial conditions $\mathbf{r_i(0)}$ and the relationship expressed in

$$
z_i(t) = \mathbf{p_i^\mathsf{T} A_i}^t \mathbf{r_i(0)}.
\tag{10}
$$

one can determine the charging schedule for each robot.

C. Maximizing flying time

In this section, we establish an optimal schedule for deploying robots by selecting which robots to operate and initializing their battery states. Unlike in the previous section, we assume specific knowledge of the number of available resources, denoted as $m \leq m_{min}$. The objective of this problem is to maximize the total operational time of all robots within the final schedule, given a finite number of resources m.

A robot, once chosen, is utilized throughout the entire mission; if not selected, it is not deployed. This decision is represented by a binary variable u_i, where $u_i = 1$ if the ith robot is deployed, and $u_i = 0$ otherwise.

Let us define a vector $\mathbf{q}_i = [\mathbf{0}^{c_i} \ \mathbf{1}^{f_i}]^T$ that can be used to determine if the ith robot is in operation or charging. The vector $\mathbf{q}_i$ is specific to the ith robot as c_i, f_i represent its charging and operation times respectively. For instance, if the ith robot has a charging time $c_i = 3$ and a operation time $f_i = 5$, the corresponding vector is $\mathbf{q}_i^T = [\mathbf{0}^3 \ \mathbf{1}^5] = [0\,0\,0\,1\,1\,1\,1\,1]$. Define a binary variable $y_i(t)$ that indicates whether the robot is in operation or not, depending on whether it is 1 or 0:

$$
y_i(t) = \mathbf{q}_i^T \mathbf{r_i(t)} = \mathbf{q}_i^T \mathbf{A}_i^t \mathbf{r_i(0)}.
\tag{11}
$$

In this scenario, our objective is to maximize the total operation time of all robots and across the time horizon:

$$
\max \sum_{i=0}^{T-1} \sum_{i=1}^{n} y_i(t),
\tag{12}
$$

subject to the following constraints:

$$
\begin{bmatrix}
\mathbf{p_1^\mathsf{T}} & \cdots & \mathbf{p_n^\mathsf{T}} \\
\mathbf{p_1^\mathsf{T} A_1} & \cdots & \mathbf{p_n^\mathsf{T} A_n} \\
\vdots & \cdots & \vdots \\
\mathbf{p_1^\mathsf{T} A_1}^{T-1} & \cdots & \mathbf{p_n^\mathsf{T} A_n}^{T-1}
\end{bmatrix}
\begin{bmatrix}
\mathbf{r_1(0)} \\
\mathbf{r_2(0)} \\
\vdots \\
\mathbf{r_n(0)}
\end{bmatrix}
\leq
\begin{bmatrix}
m \\
m \\
\vdots \\
m
\end{bmatrix},
\tag{13}
$$

$$
\begin{bmatrix}
\mathbf{1}^{T_1} & \mathbf{0}^{T_2} & \cdots & \mathbf{0}^{T_n} \\
\mathbf{0}^{T_1} & \mathbf{1}^{T_2} & \cdots & \mathbf{0}^{T_n} \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{0}^{T_1} & \mathbf{0}^{T_2} & \cdots & \mathbf{1}^{T_n}
\end{bmatrix}
\begin{bmatrix}
\mathbf{r_1(0)} \\
\mathbf{r_2(0)} \\
\vdots \\
\mathbf{r_n(0)}
\end{bmatrix}
= \mathbf{W} =
\begin{bmatrix}
u_1 \\
u_2 \\
\vdots \\
u_n
\end{bmatrix}.
\tag{14}
$$

The vectors $\mathbf{r}_i(0)$ and $\mathbf{W}$ are to be determined for each robot to optimize the desired operational outcome, which in this case, is the maximization of the operational time for the entire fleet.

Fig. 3: Graph representing the candidate cycle times

IV. ROBUST SCHEDULING

From a practical perspective, maintaining a reserve of fuel is essential for robots to account for unforeseen delays. This reserve ensures that, even if a robot is delayed, it can still reach the charging station. In Subsection IV-A, we explore the concept of safety margins, which function not only as a fuel reserve but also have the potential to reduce the scheduling horizon. This reduction in the scheduling horizon can, in turn, decrease computational complexity and simplify the overall scheduling process. Subsequently, in Subsection IV-B, we discuss a strategy for managing delayed robots without disrupting the schedules of other robots.

A. Safety margins

The cycle time T_i of the ith robot is defined as $T_i = f_i + c_i$ and, as discussed in Section III-A, the overall cycle time (scheduling horizon) T is calculated as $T = \text{LCM}(T_1, T_2, \ldots, T_n)$. However, T can potentially be large if T_i's are co-prime. Having a large T implies a longer scheduling horizon, leading to increased computational complexity of determining the charging schedule; this may not be desirable in certain applications. A shorter scheduling horizon allows for tactical flexibility.

To reduce T and achieve a lower scheduling horizon, we propose a method inspired by Dijsktra's shortest path algorithm. This approach involves leveraging the fact that T is the LCM of the individual robots's cycle times. By modifying the cycle time T_i of individual robots, T can be controlled. This modification can be achieved in two ways: a) by allowing the robot to wait at the charging station, which increases its cycle time, or b) by limiting the robot's operational time, which decreases its cycle time and provides a safety margin for the robot.

Increasing the cycle time through waiting can help reduce the scheduling horizon but also increases the number of variables, potentially increasing the computational time

for finding an optimal schedule. Conversely, reducing the operational time decreases the cycle time, the number of variables and the scheduling horizon; ultimately, it results in a lower computation time. We adopt the latter approach here.

We relax the operational time for the ith robot to belong to a set, with each candidate operational time in the set bounded by f_i. The problem thereafter reduces to finding an element in each set so that the LCM of their corresponding cycle times is the smallest.

We refer to each robot as a node and the candidate cycle times for that robot as vertices of that node. Using these candidate cycle times as vertices, we create a weighted graph, as illustrated in Fig.3 where each vertex of ith node is connected to each vertex of $(i+1)^{th}$ node. We then apply the idea of Dijkstra's shortest path to find the path that minimizes the LCM of the chosen vertices from each node.

The candidate cycle time, T_i^c, for the ith robot is:

$$T_i^c = f_i^{new} + c_i \ , \ f_i^{new} \leq f_i, \ f_i^{new} \in \mathbb{Z}^+, \ \text{for } i = 1, 2, \ldots, n. \quad (15)$$

Here, c_i (charging time) is fixed and known in advance, while f_i^{new} (operational time) is a variable that can be adjusted.

A control parameter, ε, further restricts the set, T_i^{set}, to which T_i^c can belong:

$$T_i^{set} = \{V : (1-\varepsilon_i)T_i \leq V \leq T_i \text{ and } V \in \mathbb{Z}^+\}, \ \forall \ i = 1, 2, \ldots, n. \quad (16)$$

The parameter ε represents the maximum fraction of residual battery charge when the robot arrives at the charging station for recharging.

We then construct a graph (shown in Fig.3) using these candidate cycle times V_j^i, where each vertex $V_{k_1}^i$ is connected to $V_{k_2}^{i+1}$ where $V_{k_1}^i \in T_i^{set}$ and $V_{k_2}^{i+1} \in T_{i+1}^{set}$. Additionally, we introduce two pseudo vertices at the start and end of the graph with a value of 1.

The cost associated with choosing a path through vertices $V_{k_1}^1, V_{k_2}^2, \ldots, V_{k_n}^n$ is defined as $LCM(V_{k_1}^1, V_{k_2}^2, \ldots, V_{k_n}^n)$ instead of the conventional sum of edge lengths. This approach allows us to reduce the scheduling horizon T by adjusting the operational times f_i^{new} for each robot while controlling the maximum safety margin using ε_i.

B. Managing Delay

This section addresses scenarios where a robot does not arrive at the charging station at its scheduled time slot. There are two possible situations: a) The robot arrives at the depot before its scheduled slot. In this case, the robot can wait in the waiting area until its turn comes, or b) the robot arrives later than its scheduled slot. In the second case, accommodating the tardy robot can disrupt the schedule of other robots. Since it arrived late, the tardy robot will release the resource (depart from the charging station) at a later time, causing the next robot in line to wait. This can result in a cascading effect of delays, where each subsequent robot must wait at the charging station for its turn, leading to changes in the schedule of every robot.

From Eq.(10), we can conclude that the charging schedule of a robot is solely a function of its initial conditions and physical parameters. Since this robot has missed its deadline to avail the charging station, we need to find an alternative charging schedule or initial condition that guarantees the earliest possible charging slot without disrupting the schedules of other robots.

Let us assume that the kth robot misses the deadline. From our earlier discussion from section III we can safely assume that the required charging stations m and the corresponding initial condition $\mathbf{r_i(0)} \ \forall \ i \in \{1, 2, \ldots, n\} \setminus \{k\}$ (and correspondingly, their schedules) have already been computed. Using this information, we want to find the new initial condition $\mathbf{r_k^*(0)}$ for the kth robot.

Eq.(9) can be re-written as follows:

$$\begin{bmatrix} \mathbf{p_k}^\top \\ \mathbf{p_k A_k}^\top \\ \vdots \\ \mathbf{p_k}^\top \mathbf{A_k}^{T-1} \end{bmatrix} \mathbf{r_k^*(0)} \leq \begin{bmatrix} m \\ m \\ \vdots \\ m \end{bmatrix} - \sum_{\substack{i=1 \\ i \neq k}}^{n} \begin{bmatrix} \mathbf{p_i}^\top \\ \mathbf{p_i A_i}^\top \\ \vdots \\ \mathbf{p_i}^\top \mathbf{A_i}^{T-1} \end{bmatrix} \mathbf{r_i(0)}, \quad (17)$$

$$\mathbf{1}^{T_k} \mathbf{r_k^*(0)} = 1. \quad (18)$$

Algorithm 1 Algorithm to reduce the scheduling horizon

1: **function** DIJKSTRA_LCM(graph, src, target)
2:     *cost* ← array of size $n+2$ filled with ∞
3:     *cost[src]* ← 1
4:     *pq* ← priority queue initialized with $(1, \text{src})$
5:     *visited* ← array of size $n+2$ filled with *false*
6:     *predecessors* ← array of size $n+2$ filled with -1
7:     **while** *pq* is not empty **do**
8:         Sort *pq* in descending order
9:         (min_cost, *u*) ← pop last element from *pq*
10:         **if** *visited[u]* **then**
11:             **continue**
12:         **end if**
13:         **if** *u = target* **then**
14:             *path* ← [*u*]
15:             **while** *predecessors[u]* $\neq -1$ **do**
16:                 Append *predecessors[u]* to *path*
17:                 *u* ← *predecessors[u]*
18:             **end while**
19:             **return** *min_cost, reverse(path)*
20:         **end if**
21:         *visited[u]* ← *true*
22:         **for** each (v, weight) in *graph[u]* **do**
23:             *new_cost* ← lcm(min_cost, weight)
24:             **if** *new_cost < cost[v]* **then**
25:                 *cost[v]* ← *new_cost*
26:                 Push (new_cost, v) to *pq*
27:                 *predecessors[v]* ← *u*
28:             **end if**
29:         **end for**
30:     **end while**
31: **end function**

The above set of inequalities is always feasible, as the original schedule is feasible; in this case, the tardy robot will have to wait until its turn for recharging comes again in the original schedule. In order to minimize the wait time, one can find the set of all feasible $r_k^*(0)$ that satisfy the above inequalities, and pick the one with the smallest wait time.

V. COMPUTATIONAL RESULTS

This section is divided into two parts: In subsection V-A, we compare our proposed resource minimization methodology with existing approaches. In subsection V-B, we present the results of reduced scheduling horizon. The simulations were executed on a computational platform comprising an Ubuntu 20.04 NUC computer, equipped with an Intel i7-6770HQ processor and 32GB of RAM. All simulation data is available at the following GitHub Repository.

A. Minimizing resources

We conducted a comparative analysis between our proposed Integer Linear Programming-based Scheduling (ILPS) method and the Thrift Price Window Scheduling (TPWS) method, as described in [4]. The TPWS algorithm is considered optimal when the charging time c_i (job processing time) and cycle time T_i (window) are powers of 2. For more general cases, TPWS creates a schedule by rounding the charging time C_i up to the nearest power of 2 and the cycle time T_i down to the nearest power of 2. In such scenarios, TPWS is proven to be an 8-approximation algorithm [4].

To evaluate the effectiveness of our proposed ILPS algorithm, we conducted simulations under two scenarios: (a) when all c_i and T_i are powers of 2, and (b) where c_i and T_i deviate from powers of 2. In Case (a), both TPWS and ILPS provide exactly same optimal solution, highlighting ILPS's ability to achieve optimal solutions. Case (b) is created by adding slight perturbations to Case (a) and ideally the results should not deviate drastically between the 2 scenarios. ILPS demonstrates effective results similar to case (a), unlike TPWS which fails to deliver optimal results where c_i and T_i are not powers of 2.

For both cases, we analyze how the number of charging stations required to schedule is affected by the number of robots available for deployment and the scheduling horizon $T = \text{LCM}(T_1, T_2, \ldots, T_n)$.

The simulation employs a set of 10 heterogeneous robots. Fig.4a presents the results of the two algorithms as the scheduling horizon increases. Notably, TPWS and ILPS yield optimal results for instances where the scheduling horizon and charging time are powers of 2. However, as we slightly perturb the parameters from these power-of-2 instances, TPWS deviates from the optimal solution, while ILPS consistently identifies the optimal solution. Fig 4b further demonstrates this comparison as the number of robots increases. For power-of-2 instances, ILPS aligns with TPWS, while the deviation grows in TPWS as the number of robots increases. These findings underscore the robustness of the ILPS algorithm, particularly in scenarios where TPWS encounters challenges in maintaining optimality.

(a) Optimal # of Charging Stations vs Scheduling Horizon

(b) Optimal # of Charging Stations vs # robots

Fig. 4: Performance of the Integer Linear Programming-based Scheduling (ILPS) algorithm and Thrift Price Window Scheduling (TPWS) algorithm.

(a) Computation Time

(b) Charging Stations

Fig. 5: Comparison between Original Operational Time and Reduced Operational Time.

B. Finding a new scheduling horizon with safety margins

In this section, we demonstrate the effectiveness of our proposed method for to reduce the overall scheduling horizon as a function of the safety margin parameter ε. We conducted experiments using a sample of 10 robots with random cycle times. The parameter ε is set at 10% for each robot. The table below presents the results of 15 such instances.

TABLE I: Scheduling Horizon with 10% ε value

Instance	Scheduling Horizon	Modified Scheduling Horizon
1	11592	504
2	4620	504
3	1287	540
4	5148	1287
5	9900	900
6	11310	504
7	5544	504
8	5400	540
9	7920	792
10	2700	540
11	5544	3360
12	3192	2128
13	4200	2970
14	1440	1296
15	2160	1188

As evidenced by the numbers above, scheduling robots with their maximum operational time can lead to a long scheduling horizon, and present computational challenges. However, incorporating a safety margin of up to 10% in the flying time can notably decrease the overall scheduling

Fig. 6: Schedule of the robots for the two cases: (a) when all robots are on time, and (b) when UAV2 is delayed. In both schedules, green represents the charging time and red represents the flying time.

(a) Snapshot at 4 minutes: Robots 1 and 7 continue their missions, while 2 recharging

(b) Snapshot at 11 minutes: Robots 1 and 2 continue their missions, while 7 recharging

(c) Snapshot at 17 minutes: All robots are airborne, maximizing operational efficiency

(d) Snapshot at 30 minutes: Robots 1 continue their missions, while 2 and 7 recharging

Fig. 7: Sequential snapshots from the simulation demonstrate the effectiveness of the scheduling algorithm at different stages of the robot operation cycle. The algorithm optimally selected and scheduled 3 out of 7 available robots (UAV1, UAV2, and UAV7), maximizing total flying time during the scheduling horizon with only 2 charging stations available. The figure on the left shows the simulation based on the original schedule, while the figure on the right corresponds to the simulation with the new schedule, adjusted for a 2-minute delay of UAV2.

horizon and computational time for finding a schedule. Fig.5a illustrates the effectiveness of reducing computation time.

VI. SIMULATION RESULTS

We now present the results of simulations conducted to validate the effectiveness of a) selecting and scheduling robots when charging resources are limited, b) reducing the scheduling horizon along with incorporating a safety margin in flight time, and c) demonstrating robustness when a robot misses its scheduled charging time. These simulations involve selecting among 7 robots with 2 charging stations. Detailed parameters regarding simulations can be found in the GitHub Repository.

The simulations were executed on a computational platform comprising an Ubuntu 22.04 NUC computer, equipped with an Intel i7-6770HQ processor and 32GB of RAM. The simulation environment replicated a realistic operational scenario where multiple robots were required to complete a series of tasks while managing their energy consumption effectively.

- All robots depart from a designated charging spot at coordinates (50,100).
- The total number of available robots is 7, parameters details can be found from the given link: GitHub Repository.
- Robots were tasked with visiting predetermined locations, represented by random 30 vertices, to simulate a

typical surveillance or delivery mission.

- The robots operate at an average speed of 16 m/s, powered by 4000mAh 22.2V Li-po batteries.
- The duration of time slot chosen for scheduling is 1 minute.
- The path planning relied on the meta-heuristic team orienteering problem, which was implemented to determine the feasible sequence of visits. [15]

The algorithm selected 3 robots (UAV1, UAV2, UAV7) from the 7 available robots, with safety margins of 1, 2, and 4 minutes respectively. This selection provided a total flying time of 58 minutes within a cycle time of 36 minutes and effectively scheduled the robots on 2 charging stations. The schedule of the robots is shown in Fig.6. The green shaded region represents the time when the robot needs to secure the charging station, and the red region is the time when the robot is available for the mission. This schedule repeats every 36 minutes.

We also considered the case when a robot misses its deadline to secure the charging station. As seen in Fig.6, the robot (UAV2) needs to secure the charging station at 1 minute. However, if, due to unforeseen circumstances, the robot reaches the charging station at 3 minutes, the situation changes. UAV2 has a safety margin of 2 minutes, so reaching late is not immediately critical. However, providing a charging station at that time might disrupt the schedule of other robots. Using the discussion from Section IV, we found that the earliest available charging station at 3 minutes can be given according to the new schedule in Fig.6, compared to 10 minutes as per the original schedule without disrupting the cycle of other robots.

To visualize the newly computed schedule, Fig.7 displays snapshots from the simulation at key moments: the 4th, 11th, 17th, and 30th slots. The circular icons at the top of each image indicate the status of the charging stations. Initially, at the 4-minute time slot, robot 2 is shown on the charging pads, strategically synchronized with robots 1 and 7, who are in the middle of their missions. As the simulation progresses, it is evident that no more than 2 charging stations are required at any given time to continue the mission. The feasible sequence of visits by the robots is determined using the meta-heuristic team orienteering problem discussed in [15], showcasing the algorithm's adaptability to existing literature.

VII. CONCLUSIONS

This paper presents a framework for efficiently utilizing charging stations by staggering the robots for recharging to achieve long-term autonomy. It discusses the practical implementation of the algorithm by introducing the concept of a reduced scheduling horizon, which also provides a safety margin for the robots. The paper includes simulation results for a persistent surveillance problem to demonstrate the ease of extending this algorithm to different existing works in the literature. Additionally, the paper addresses robustness analysis, highlighting how the system adapts when a robot fails to secure a charging station on time. Overall, this paper covers the entire framework from scheduling to safety

margins to robustness analysis, showing its adaptability and potential for integration into existing literature.

Currently, this method primarily focuses on cases where robots are fully charged and discharged. A natural extension of this work would be minimizing the resources required when partial charging and discharging is allowed.

The general problem involves combining routing of battery-limited robots with scheduling their recharging at the charging stations. The adopted approach decouples the routing problem from the scheduling problem in the following way: the operational time becomes a constraint for routing a robot so that all points of interest are covered within the smallest number of cycles.

REFERENCES

[1] N. Boysen, D. Briskorn, S. Fedtke, and S. Schwerdfeger, "Drone delivery from trucks: Drone scheduling for given truck routes," *Networks*, vol. 72, no. 4, pp. 506–527, 2018.

[2] C. Liu, H. Chen, X. Li, and Z. Liu, "A scheduling decision support model for minimizing the number of drones with dynamic package arrivals and personalized deadlines," *Expert Systems with Applications*, vol. 167, p. 114157, 2021.

[3] "Watts innovations's prsim lite drone," https://wattsinnovations.com/products/prism-lite. [Online]. Available: https://wattsinnovations.com/products/prism-lite

[4] A. Bar-Noy, R. E. Ladner, T. Tamir, and T. VanDeGrift, "Windows scheduling of arbitrary length jobs on parallel machines," in *Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures*, 2005, pp. 56–65.

[5] M. Shin, J. Kim, and M. Levorato, "Auction-based charging scheduling with deep learning framework for multi-drone networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4235–4248, 2019.

[6] V. Hassija, V. Saxena, and V. Chamola, "Scheduling drone charging for multi-drone network based on consensus time-stamp and game theory," *Computer Communications*, vol. 149, pp. 51–61, 2020.

[7] M. Torky, M. El-Dosuky, E. Goda, V. Snášel, and A. E. Hassanien, "Scheduling and securing drone charging system using particle swarm optimization and blockchain technology," *Drones*, vol. 6, no. 9, p. 237, 2022.

[8] K. Yu, A. K. Budhiraja, S. Buebel, and P. Tokekar, "Algorithms and experiments on routing of unmanned aerial vehicles with mobile recharging stations," *Journal of Field Robotics*, vol. 36, no. 3, pp. 602–616, 2019.

[9] S. K. K. Hari, S. Rathinam, S. Darbha, K. Kalyanam, S. G. Manyam, and D. Casbeer, "Optimal uav route planning for persistent monitoring missions," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 550–566, 2021.

[10] S. K. K. Hari, S. Rathinam, S. Darbha, S. G. Manyam, K. Kalyanam, and D. Casbeer, "Bounds on optimal revisit times in persistent monitoring missions with a distinct and remote service station," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1070–1086, 2023.

[11] S. Ahmed, A. Mohamed, K. Harras, M. Kholief, and S. Mesbah, "Energy efficient path planning techniques for uav-based systems with space discretization," in *2016 IEEE wireless communications and networking conference*. IEEE, 2016, pp. 1–6.

[12] J. Kim, B. D. Song, and J. R. Morrison, "On the scheduling of systems of uavs and fuel service stations for long-term mission fulfillment," *Journal of Intelligent & Robotic Systems*, vol. 70, pp. 347–359, 2013.

[13] K. Chour, J.-P. Reddinger, J. Dotterweich, M. Childers, J. Humann, S. Rathinam, and S. Darbha, "An agent-based modeling framework for the multi-uav rendezvous recharging problem," *Robotics and Autonomous Systems*, vol. 166, p. 104442, 2023.

[14] A. Bilal Asghar, S. Sundaram, and S. L. Smith, "Multi-Robot Persistent Monitoring: Minimizing Latency and Number of Robots with Recharging Constraints," *arXiv e-prints*, p. arXiv:2303.08935, Mar. 2023.

[15] J. J. Lee and S. Rathinam, "A meta-heuristic approach for an aerial-ground vehicle path planning problem," in *AIAA SCITECH 2024 Forum*, 2024, p. 0230.