IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 21 March 2024, accepted 22 April 2024, date of publication 27 May 2024, date of current version 3 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3405796

RESEARCH ARTICLE

On the Scheduling of Spatio-Temporal Charging Windows for Autonomous Drone Fleets

KASPAR HAGEMAN AND RUNE HYLSBERG JACOBSEN, (Senior Member, IEEE)
Department of Electrical and Computer Engineering, Aarhus University, 8000 Aarhus, Denmark
Corresponding author: Kaspar Hageman (kh@ece.au.dk)

This work was supported by the European Union's Horizon 2020 Research and Innovation Project Drones4Safety under Grant 861111.

ABSTRACT The availability of low-cost unmanned aerial vehicles (UAVs), or drones, has made their organisation in fleets more feasible. The required coordination for managing these fleets comes with an increased complexity. When used for long-durability, autonomous inspection missions, it is necessary to recharge the drones due to their limited battery capacity. By providing a set of nearby charging stations, the fleets can autonomously recharge and sustain indefinite missions. In order to reduce congestion at these charging stations, effective scheduling of charging cycles can have a significant impact on the mission execution time. In this paper, we propose a novel centralized method for scheduling charging time windows, taking into account the travel distances and occupation of charging stations. We formulate a mixed-integer linear program (MILP) model with two extensions to reduce the computational complexity. The solution to this problem assigns a set of charging windows to each drone, minimizing the mission execution time and ensuring batteries will not fully deplete. The performance of our proposed method is evaluated through a series of experiments, based on a discrete-event simulator. Our results reveal a clear benefit over a greedy approach, reducing the mission execution time by up to 39.8%. Through careful parameter selection, a trade-off between mission execution time and scheduling time can be found.

INDEX TERMS Charging, discrete-event simulation, drone fleets, MILP, scheduling, time windows.

I. INTRODUCTION

In recent years, the reduction of hardware cost and increase in computation power has made small unmanned aerial vehicles (UAVs), or drones, more suitable for new application areas. When equipped with additional sensors, they have immense potential to execute tasks that are currently being done by humans, such as "last mile" parcel delivery [1], search and rescue operations [2], and safety-critical infrastructure inspection [3], [4], [5], [6]. The presence of lightweight and powerful onboard computers allows drones to autonomously make decisions, removing the reliance on a human operator. Autonomous missions can benefit from organising multiple drones in *fleets* to complete tasks in a coordinated fashion, which reduces the task completion time and increases overall resiliency [7]. These benefits come with an increased complexity of coordination and involve communication, task allocation, collision avoidance, and path planning.

The typical battery capacity of a drone lasts for tens of minutes [8]. Depending on the application, this might not be sufficient, and recharging or replacement facilities must be available. This paper considers a specific use case, in which an ordered set of waypoints, or points in a three-dimensional space, is allocated to each drone prior to a mission, and a set of charging stations is positioned near these waypoints. By interrupting the mission execution (i.e., visiting these waypoints in sequence) to charge their batteries at one of the charging stations, each drone can operate indefinitely. In the presence of a large fleet, a strategy is required to determine the order in which drones should be served by each of the stations (i.e., a temporal aspect). Similarly, a spatial strategy is required to determine which drone should be served at which charging stations. The spatial strategy determines the path that a drone takes during its mission, thereby affecting its energy consumption and arrival time at charging stations.

The associate editor coordinating the review of this manuscript and approving it for publication was Jie Gao.

As a result, spatial and temporal scheduling are naturally intertwined.

In this paper, we seek to obtain the optimal time window scheduling method for a given scenario, composed of waypoint and charging station locations, taking into account relevant characteristics of the drones, such as their speed, minimum required battery level, and charging and depletion rates. A schedule consists of a set of time windows, each of which indicates the start time and duration of a particular charge station visit for a drone, and guarantees that the battery levels of the drone remain above a minimum threshold. The optimal schedule is the composition of time windows such that the maximum mission execution time across drones is minimized. This particular scheduling problem has not been addressed by the research community, and yet naturally fits concrete drone fleet applications, such as disaster relief or the inspection of infrastructure. We recognize the following design principles that the optimal schedule must adhere to and are particularly relevant to our use case:

- The optimal schedule is continuous in time, as opposed to the discretization of time into slots.
- The charging duration of a drone is relatively long to its depletion duration [8]. As a consequence, each charging window has a variable duration and its duration directly affects the charged battery level.
- It is not always beneficial for a drone to fully charge its battery and instead might terminate charging preemptively.
- Charging stations can only serve a maximum number of drones simultaneously, and to avoid queuing and crowding around a charging station, drones should be able to be routed to other charging stations.

We propose a centralized method for obtaining the optimal charging schedule by formulating a mixed-integer linear programming (MILP) model aimed to minimize the mission execution time, that can be solved by off-the-shelve commercial tools. The model addresses the four aforementioned principles and is parameterized by the positioning of charging stations and waypoints, velocity of drones, depletion and charging rates, and required battery levels of the drones. Furthermore, we extend this model to address the computational complexity and uncertainty of the operation environment by repeatedly formulating and solving smaller, less complex models. We evaluate the efficacy of our proposed methods and the effect of the parameterization by comparing them to a greedy approach. For this evaluation, we implement a discrete-event simulator and design a set of experiments on a bridge inspection use case. We share the source code of our simulator, experiments, and data processing.[1] Our major simulation findings are as follows:

- Our optimal approach consistently outperforms a greedy baseline approach for a variety of parameters and configurations.
- This overall optimality is achieved by sacrificing the mission execution time of faster drones for the reduction of the execution time of slower drones.
- The parameterization of our MILP problem allows operators to make a trade-off between the optimality of the mission execution time and the time used for scheduling.

The related work is described in Section II. We introduce the mathematical formulation of the charge scheduling problem in Section III and introduce the extensions in Section IV. This is followed by an in-depth evaluation in Section V. We conclude the paper with a discussion in Section VI and conclusions in Section VII.

II. RELATED WORK

The state-of-the-art related work differs from our work in two significant ways: the target use case is different or they tackle only a subset of the aforementioned principles that are key to our use case (i.e., continuous time, variable charging duration, charge interruption, and queuing avoidance at charging stations). We present an overview of the significant related works and denote the key differences to our work, summarizing these findings in Table 1. In cases where the authors did not explicitly mention a principle, we assume it is not considered in their design. We acknowledge the comprehensive literature on optimal electric vehicle charging addressing related issues. However, this falls outside the scope of our analysis

As a first remark, the scheduling problem we address in this paper is orthogonal to combinatorial problems such as the vehicle routing problem (VRP) [26] and the job-shop scheduling problem (JSP) [14]. Our proposed approach assumes ordered sequences of waypoints, differentiating our work from the VRP. Similar to the JSP, a series of ordered jobs – in our case a variable number of charging operations per drone – need to be processed by a set of machines, the charging stations. However, in our use case, the number and duration of jobs are variable and dependent on how previous jobs are scheduled, differentiating our problem definition from the JSP.

Pasha et al. [27] provide a state-of-the-art overview of charging-related drone scheduling problems. It coins the term Drone Scheduling Problem with Recharging Considerations (DSPRC), an optimization problem formulation originating from Kim et al. [9]. The use case of this work involves employing a drone fleet to track the path of an object such that the object is constantly and perpetually tracked by at least one drone, charging drones along the tracking operation. This work is further refined to reduce the scheduling time [10] and to reduce the complexity of the problem through scheduling for a short, receding horizon [11], [28]. In contrast to our work, these MILP formulations assume a fixed charging time, as opposed to one that depends on the battery state and varying charging rates of the drones. Kim et al. [14] propose a centralized path finding and charge scheduling system for drone fleets, which must navigate from a source

[1] https://github.com/kdhageman/spatiotemporal_charge_scheduling

TABLE 1. Overview of relevant related work.

Year	Author	Method	Continuous time	Variable charging time	Charge interruption	Charging queueing avoidance	Contribution
2013	Kim *et al.* [9]	MILP	✗	✗	✗	✗	Constant following of trajectories, split jobs
2014	Song *et al.* [10]	MILP	✗	✗	✗	✗	Efficiency improvement over [9]
2014	Kim *et al.* [11]	MILP, RH	✗	✗	✗	✗	Receding horizon formulation of [9], [10]
2018	Song *et al.* [12]	MILP, RH	✗	✗	✗	✗	Distance-constrained vehicle routing problem
2018	Ghazzai *et al.* [13]	MI(N)LP	✗[1]	✓	✓	✗	Event coverage and charge scheduling
2019	Kim *et al.* [14]	SP	✓	✓	✗	✓	Drone routing, congestion avoidance
2019	Shin *et al.* [15]	Auction, ML	✗	✓	✓	✓	Drone charge scheduling for single station
2019	Ghazzai *et al.* [16]	MI(N)LP, PSO	✗[1]	✓	✓	✗	Charging station placement, timeslot-based scheduling
2020	Hassija *et al.* [17]	Auction, GT	✗	✓	✓	✓	Drone charge scheduling, decentralization
2020	Liu *et al.* [18]	RL	✗	✓	✓	✗	Route planning including charge station visits
2020	Ribeiro *et al.* [19]	MILP	✗	✗	✗	✗	Optimal route planning with charging scheduling
2020	Hassija *et al.* [20]	GT, DLT	✗	✓	✓	✗	Energy trading among UAVs and charging stations
2021	Ahani *et al.* [21]	GLA	✗	✓	✓	✗	Age-optimal UAV scheduling
2022	Arafat *et al.* [22]	Clustering, MILP	✓	✓	✗	✗	Drone-delivery routing problem
2022	Torky *et al.* [23]	PSO, GT, DLT	✗	✗	✗	✓	Transactive charging scheduling. Improvement over [17]
2022	Pinto *et al.* [24]	MILP, SP	✗	✗	✗	✗	Optimal charging network infrastructure design
2023	Wang *et al.* [25]	MILP, MDP	✗	✓	✗	✓	Spatio-temporal path planning optimization
2024	This work	MILP, RH	✓	✓	✓	✓	

MI(N)LP = Mixed-Integer (Non)Linear Program, **RH** = Receding Horizon, **SP** = Shortest Path, **ML** = Machine Learning, **PSO** = Particle Swarm Optimization, **GT** = Game Theory, **RL** = Reinforcement Learning, **GLA** = Graph Labeling Algorithm, **MDP** = Markov Decision Process, **DLT** = Distributed Ledger Technology.

[1] Time is discretized in heterogeneous time slots based on the events that must be handled, preserving significant time slot boundaries.

to a destination. This work considers relevant aspects, such as continuous time scheduling, avoiding simultaneous charging, travel time prediction (both waiting, charging, and flying time) and battery depletion rates. Their shortest path-based method navigates drones through a network of charging stations between a source and target location – a different use case from ours where there is a sequence of target locations – and drones are assumed to always fully charge. Similarly, Song et al. [12] propose a MILP formulation for solving a distance-constrained version of the vehicle routing problem, combining it with a receding horizon to address the NP-hardness of the problem. In contrast to our work, they do not address the potential conflict of multiple drones trying to charge simultaneously which our approach does resolve. In [24], the authors addressed the problem of extending the drones operating range from a network design perspective. The authors used a mixed-integer optimization model and a heuristics method to find the optimal number and location of charging stations with respect to a set of delivery points. In contrast to our use case, the set of delivery points can potentially be significantly spread out whereas our use case considers drone inspection points that are bound to the infrastructure under inspection. Ribeiro et al. [19] used a MILP formulation to tackle the routing problem in inspection and charging planning. In contrast our work the study focused on optimal route planning and did not consider temporal aspects. The authored compared different solvers to evaluate the feasibility of handling the problem complexity.

Auction-based methods allow drones with competing interests – their desire to charge interferes with the desire of other drones – to agree on an order in which drones can charge at a station. Shin et al. [15] propose a centralized and machine learning-based charge scheduling system, in which a single charging station collects bids from drones and assigns charging slots to the winner of an auction. The output of this proposed method is effectively an ordering of drone schedules, as only a single charging station is involved. Alternatively, Hassija et al. [17] propose a decentralized, blockchain-based strategy, in which drones join and leave a distributed ledger network with charging stations and agree on schedules and prices. The concept was further explored in [20], where UAVs were equipped with tokens to buy energy from charging points using a tangle data structure. An improvement of the work in [17] was later presented by Torky et al. [23]. The improvement resulted from the combination of the auction based method with particle swarm optimization (PSO). None of these methods plan in advance or anticipate future events.

A generic drone swarming scheduling problem is addressed by Ghazzai et al. [13], in which a drone fleet is assigned to cover events, taking into account their location, and starting and end times. The drones can charge at a single charging station. This work is followed by [16] in which the placement of charging stations is optimized and multiple charging stations are introduced. The authors discretize time into timeslots of different lengths, to align time slots with the start and end times of the events that must be handled. In the latter, the authors do not explicitly prohibit drones from charging at the same station simultaneously. Wang et al. [25] studied a related problem analyzing cooperative path planning algorithms of a UAV swarm for optimally servicing many spatial locations with dynamically released demands.

Even though our work differs from the VRP, our work shares similarities with battery-constrained routing planning methods. Liu et al. [18] introduce a reinforcement learning-based method for route planning and deciding when to charge a swarm of drones, focusing on visiting a set of sensing nodes for data collection. In their approach, congestion at charging stations is not considered. Similarly, Ahani et al. [21] tackle a variant of the VRP, in which a UAV must collect information from sensor nodes to deliver to a base station. The time since the last collection of data, referred to as the age of information, acts as an important metric for defining which nodes to visit per iteration, to optimize the freshness of the sensor data at the base station. In their graph labeling approach, time is discretized and partial charging is allowed but it does not address multiple drones nor multiple charging stations, thereby not addressing scheduling conflicts. Arafat and Moh [22] focus on the drone delivery routing problem (DDRP), proposing a three-phase process (preprocessing, flight segment construction, delivery route construction). As such, they consider both the routing problem and the charge scheduling simultaneously. Their work does not address the prevention of charge schedule collisions.

In conclusion, none of the state of the art takes the four design principles into account. They are typically concerned with slightly different use cases (e.g., routing, covering time-sensitive events, or congestion avoidance only). Notably, the long charge duration of a battery is typically not mentioned in these works, possibly a reason why few works include charge interruption, and is a significant consideration in our work.

III. PROBLEM FORMULATION

In this section, we introduce the general context in which we operate. Consider N_d drones $\mathbb{D} = \{D_1, D_2, \ldots, D_{N_d}\}$. The drones are equipped with wireless communication hardware that allows them to communicate with a ground control station. This station provides each drone D_d with N_w waypoints (i.e., three-dimensional points as part of the inspection mission) to visit in a predetermined order $\mathbb{W}_d = \{W_{d,0}, W_{d,1}, W_{d,2}, \ldots, W_{d,N_w}\}$, beginning at a starting position $W_{d,0}$. When a drone is allocated fewer waypoints in reality, the list of waypoints can be padded to ensure the presence of N_w waypoints. We omit the dynamics of the vehicle and assume the drones can change their direction instantly. In the vicinity of the waypoints, there are N_s charging stations $\mathbb{S} = \{S_1, S_2, \ldots, S_{N_s}\}$ with fixed positions deployed, which can be used by drones to dock to and to recharge their batteries [29]. Each charging station can only serve a single drone at a time. When navigating between waypoints, drones deplete their battery and this battery level is not allowed to fall below a minimum threshold in order to guarantee a successful mission execution. As such, after each *source* waypoint W_{d,w_s}, i.e., a non-terminating waypoint ($0 \leq w_s \leq N_w - 1$), a drone D_d has the option to visit either of the charging stations or to directly visit the next waypoint, resulting in the path node options $\mathbb{N}_{d,w_s} = \mathbb{S} \cup \{W_{d,w_s+1}\} = \{N_{d,1,w_s}, N_{d,2,w_s}, \ldots, N_{d,N_s+1,w_s}\}$ (see Fig. 1). At the charging station, drones optionally wait for another drone to finish charging, charge until a desired battery level has been reached, and continue to the next waypoint. During these activities, the battery percentage of a drone increases, remains the same, or decreases when charging, waiting, and flying between nodes respectively. Our goal is to find an optimal charging schedule such that the mission execution time is minimized, under the constraints that drone batteries are never depleted and that charging stations are never allocated to more than one drone simultaneously. We formulate a mathematical optimization problem for this purpose. The indices used so far have been summarized on Table 2 and will continue to be used throughout the paper in the mathematical notation.

TABLE 2. List of indices and their ranges.

Name	Index	Range of values
Drone	d	$\{1, 2, \ldots, N_d\}$
Waypoint	w	$\{0, 1, \ldots, N_w\}$
Source waypoint	w_s	$\{0, 1, \ldots, N_w - 1\}$
Charging station	s	$\{1, 2, \ldots, N_s\}$
Path node	n	$\{1, 2, \ldots N_s + 1\}$

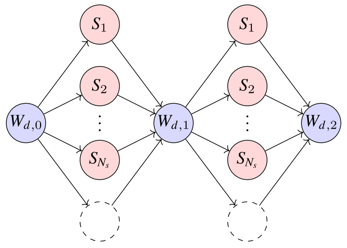


FIGURE 1. After all non-terminating waypoints ($W_{d,0}$ and $W_{d,1}$), the drone must visit a *path node*: any charging station (red nodes) or move directly to the next waypoint (the dotted nodes).

A. PARAMETERS

Our optimization problem is parameterized by a set of values that are provided prior to the mission execution. We convert the three-dimensional positions of the waypoints and charging stations into two distance matrices $\overrightarrow{\Delta}_{d,n,w_s}$ and $\overleftarrow{\Delta}_{d,n,w_s}$ (measured in meters). The former represents the distance from a waypoint W_{d,w_s} to the next path node N_{d,n,w_s}, whereas the latter represents the distance from the path node to the next waypoint. Typically, the Euclidean distance between these points can be used, but this is not necessary. For path nodes that indicate moving to the next waypoint directly (i.e., $n = N_s + 1$) the entries in $\overleftarrow{\Delta}$ are set to zero. We assume a linear depletion and charging profile for a battery [30], and as such the parameters r_d^+ and r_d^- denote the battery percentage that is depleted and charged per second for drone D_d respectively. The choice of this linearity allows us to express the optimization problem as a (more easily solvable)

TABLE 3. List of parameters, with variables introduced by the extensions in the bottom section.

Parameters	Description
$\overrightarrow{\Delta}_{d,n,w_s}$	Distance between waypoint W_{d,w_s} and path node N_{d,n,w_s}
$\overleftarrow{\Delta}_{d,n,w_s}$	Distance between path node N_{d,n,w_s} and waypoint $W_{d,w_{s+1}}$
r_d^+	Charging rate (% per second) for drone d
r_d^-	Depletion rate (% per second) for drone d
v_d	Speed of drone d
β_d^{start}	Start battery percentage for drone d
$\beta_{d,w}^{min}$	Minimum allowed battery percentage for drone d after waypoint w
$\beta_d^{default}$	Default minimum battery percentage for drone d
β_d^{max}	Maximum allowed battery percentage for drone d
ϵ	Time window (in seconds) between consecutive charging time slots at the same station
I_d^{max}	Maximum waiting time for drone d
C_d^{max}	Maximum charging time for drone d
$\omega_{d,s}$	Minimum waiting time of drone d at charging station s after the first scheduled waypoint
ρ_d	Estimated remaining distance in mission after the horizon, for drone d
$\hat{N}_w$	Horizon size (in # of waypoints)
π	Rescheduling frequency (in # of waypoints)
σ	Anchor stride

linear problem and can be seen as the assumed average depletion over time. Alternatively, more complex depletion profiles could be injected in the optimization problem [22], [31]. In our case, we consider the computational complexity of the problem more important than the accuracy of the battery profile. On top, each drone has a speed v_d (meters per second), a starting battery percentage β_d^{start}, a lower battery percentage threshold $\beta_{d,w}^{min}$ at a given waypoint $W_{d,w}$ and an upper battery percentage threshold β_d^{max}. Note that the differentiation of minimum battery levels at different waypoints is necessary later on in Section IV. For now, this value can be assumed to be a default constant $\beta_d^{default}$ across waypoints ($\beta_{d,w}^{min} = \beta_d^{default}$). Furthermore, we require a separation window between the charging windows of drones at the same charging station to give the drones time to (un)dock. This separation window ϵ is measured in seconds. An overview of the parameters is provided in Table 3.

B. OBJECTIVE

To schedule the fleet of drones, we introduce the decision variables P, C and I, which determine the path the drones take, the time the drones charge and the time the drones wait (or idle) for another drone to finish charging respectively (see the top section in Table 4). For each drone, its execution time (E_d) consists of the sum of the charging (C_{d,w_s}), waiting (I_{d,w_s}), and moving time (t_{d,w_s}) across all of its visited waypoints.

$$t_{d,w_s} = \frac{1}{v_d} \sum_n \left[P_{d,n,w_s} (\overrightarrow{\Delta}_{d,n,w_s} + \overleftarrow{\Delta}_{d,n,w_s}) \right] \tag{1}$$

TABLE 4. List of variables, with decision variables in the top section and computed variables in the bottom section.

Variables	Description
P_{d,n,w_s}	$\begin{cases} 1 & \text{if drone } D_d \text{ moves via path node } N_{d,n,w_s} \text{ from waypoint } W_{d,w_s} \\ 0 & \text{otherwise} \end{cases}$
C_{d,w_s}	charge duration of drone D_d after waypoint W_{d,w_s}
I_{d,w_s}	waiting duration of drone D_d after waypoint W_{d,w_s} prior to charging
$\Gamma_{d,d',w_s,w_s'}$	$\begin{cases} 1 & \text{if drone } D_d \text{ charges before drone } D_{d'} \text{ at their respective waypoints } W_{d,w_s} \text{ and } W_{d',w_s'} \\ 0 & \text{otherwise} \end{cases}$
$b_{d,w}^*$	battery charge of drone D_d when arriving at waypoint $W_{d,w}$
b_{d,w_s}^-	battery charge of drone D_d when arriving at path node after waypoint W_{d,w_s}
b_{d,w_s}^+	battery charge of drone D_d after charging at the path node after waypoint W_{d,w_s}
$\overset{s}{\tau}_{d,w_s}$	start timestamp of charging window of drone D_d after waypoint W_{d,w_s}
$\overset{e}{\tau}_{d,w_s}$	end timestamp of charging window of drone D_d after waypoint W_{d,w_s}
$\theta_{d,d',w_s,w_s'}$	$\begin{cases} 1 & \text{if drone } D_d \text{ and } D_{d'} \text{ occupy the same charging station at their respective waypoints } W_{d,w_s} \text{ and } W_{d',w_s'} \\ 0 & \text{otherwise} \end{cases}$

$$E_d = \sum_{w_s} \left[C_{d,w_s} + I_{d,w_s} + t_{d,w_s}) \right] \tag{2}$$

Note that the drone speed v_d is assumed to be constant. The objective is to minimize the maximum mission execution time across all drones:

$$\textbf{Minimize} \quad \max_d E_d \quad \text{(objective)}$$

C. CONSTRAINTS

We impose several constraints on the objective to ensure the battery level requirements and non-overlapping property of the charging windows of the drones. To do so, we must introduce variables related to the battery state of each drone (b^*, b^-, b^+) and the window overlap ($\overset{s}{\tau}, \overset{e}{\tau}, \Gamma, \theta$) (see the bottom section in Table 4). The former three capture the battery level of a drone D_d at all moments of the mission execution (at arrival at a waypoint, at arrival at a path node, and after charging at a path node respectively). At each waypoint W_{d,w_s}, the *charging window* of a drone D_d is bounded by a starting ($\overset{s}{\tau}_{d,w_s}$) and ending ($\overset{e}{\tau}_{d,w_s}$) timestamp. The variables Γ and θ are used later on in the constraint definition to ensure the windows of different drones do not overlap when charging at the same station.

First, we introduce several constraints that calculate the state of several variables:

$$b_{d,0}^* = \beta_d^{start} \tag{3}$$

$$b_{d,w_s}^- = b_{d,w_s}^* - \frac{r_d^-}{v_d} \sum_n \left[P_{d,n,w_s} \overrightarrow{\Delta}_{d,n,w_s} \right] \tag{4}$$

$$b_{d,w_s}^+ = b_{d,w_s}^- + r_d^+ C_{d,w_s} \tag{5}$$

$$b^*_{d,w_s+1} = b^+_{d,w_s} - \frac{r^-_d}{v_d} \sum_n \left[P_{d,n,w_s} \overleftarrow{\Delta}_{d,n,w_s} \right] \tag{6}$$

$$\theta_{d,d',w_s,w'_s} = \sum_s P_{d,s,w} P_{d',s,w'_s} \tag{7}$$

$$\overset{s}{\tau}_{d,w_s} = \sum_{w_p=1}^{w_s-1} \left[C_{d,w_p} + I_{d,w_p} + t_{d,w_p} \right] + \frac{1}{v_d} \sum_n \left[P_{d,n,w_s} \overrightarrow{\Delta}_{d,n,w_s} \right] + I_{d,w_s} \tag{8}$$

$$\overset{e}{\tau}_{d,w_s} = \overset{s}{\tau}_{d,w_s} + C_{d,w_s} \tag{9}$$

Equations 3 - 6 express the state of the battery. Equation 7 is used to express whether two drones D_d and $D_{d'}$ overlap at their respective waypoints W_{d,w_s} and W_{d,w'_s}. Equations 8 and 9 establish the start and end points of the charging window. We then impose limits on the values of the decision variables as follows:

$$\sum_n P_{d,n,w_s} = 1 \tag{10}$$

$$0 \leq C_{d,w_s} \leq (1 - P_{d,N_s+1,w_s}) C^{max}_d \tag{11}$$

$$0 \leq I_{d,w_s} \leq (1 - P_{d,N_s+1,w_s}) \sum_{d' \neq d} I^{max}_{d'} \tag{12}$$

$$b^*_{d,w_d} \geq \beta^{min}_{d,w_d} \tag{13}$$

$$b^-_{d,w_s} \geq \beta^{min}_{d,w_s} \tag{14}$$

$$b^+_{d,w_s} \leq \beta^{max}_d \tag{15}$$

$$\overset{e}{\tau}_{d,w_s} \leq \overset{s}{\tau}_{d',w'_s} - \epsilon + M(1 + \Gamma_{d,d',w_s,w'_s} - \theta_{d,d',w_s,w'_s}) \tag{16}$$

$$\overset{s}{\tau}_{d',w'_s} \leq \overset{s}{\tau}_{d,w_s} - \epsilon + M(2 - \Gamma_{d,d',w_s,w'_s} - \theta_{d,d',w_s,w'_s}) \tag{17}$$

Equation 10 forces each drone to visit exactly one path node after each source waypoint. Equations 11 and 12 prevent drones from waiting or charging whenever they move directly between consecutive waypoints without a charging station visit. The values for C^{max} and I^{max} must be higher than any potential charging and waiting time respectively (see Appendix for their calculation). Equations 13 - 15 force the lower and upper limits of the battery charge at any point during the mission. Equations 16 and 17 ensure the charging windows of drones D_d and $D_{d'}$ are not overlapping at their respective waypoints $W_{d,w}$ and $W_{d',w'}$. The ϵ serves two purposes: (1) to prevent edge cases in which two charging windows overlap precisely and (2) to force an arbitrary distance between charge windows to emulate docking/undocking behaviour. The term on the second line of both equations allows for *both* constraints to become non-binding whenever the two drones are not charging at the same time (i.e., when the θ term equals one) or *either* constraint to become non-binding through Γ. The M must be sufficiently large to exceed any possible value for $\overset{s}{\tau}$ (see Appendix VII for their calculation). Note that Equations 16 and 17 are not exact opposites of one another; instead they impose constraints on the start of the time window of D_d ($\overset{s}{\tau}_{d,w_s}$ and $\overset{e}{\tau}_{d,w_s}$) in relation to the time window of D'_d. For each pair of drones D_d and D'_d, there is a symmetrical set of constraints that impose constraints on the starting of the time window of D'_d, in which the d and d' and waypoint indices w_s and w'_s indices are swapped.

The objective function and Equation 7 are non-linear, and we rely on linearization techniques [32] to retain a Mixed Integer Linear Problem (MILP).

IV. EXTENSIONS

The number of variables and constraints grows quadratically with both the number of waypoints and the number of drones (due to the Γ and θ decision variables). As a result, solving medium-sized scenarios using conventional off-the-shelve solvers becomes computationally infeasible and there is a need to reduce the computational complexity. A second limitation of the MILP definition is that the full charging schedule is computed before the mission execution and can therefore not take into account any uncertainty in the mission execution environment. Unforeseen events, such as obstacles or unexpected wind and temperature conditions, can impact the distance traveled and battery depletion rate of the drones. We propose two improvements to address these two limitations.

A. WAYPOINT HORIZON SCHEDULING

Firstly, we take inspiration from *receding horizon control* [33]. Instead of solving a problem for all N_w waypoints, we solve a reduced problem with the first $\hat{N}_w$ waypoints, or the *horizon size*, for each drone instead, where $\hat{N}_w \leq N_w$. This resulting schedule will be followed by the drone fleet for a given number of waypoints, controlled by a rescheduling frequency parameter π. The first drone that reaches π waypoints triggers a centralized rescheduling process, in which a new MILP is constructed given the current state of the drone fleet (positions and battery statuses). The current positions of the drones serve as starting positions in the MILP parameterization. The solution of this MILP is once again followed by all drones in the fleet for π waypoints. We repeat this process until the mission is completed. Whenever the remaining set of waypoints is smaller than the horizon size, we add auxiliary waypoints as padding – with the same position as the last 'real' waypoint – to ensure each drone has an equal number of waypoints.

B. ANCHOR WAYPOINTS

Increasing the waypoint horizon negatively affects the computation time of the scheduling while positively affecting the overall performance of the generated schedule. To circumvent this trade-off, we down-sample the waypoints that are eligible to be followed by a visit to a charging station. A subset of the waypoints, refered to as *anchor* waypoints are considered, whereas the remaining waypoints must be visited without a charging station visit. This sacrifices optimality of the calculated schedule, but allows scheduling efficiently with

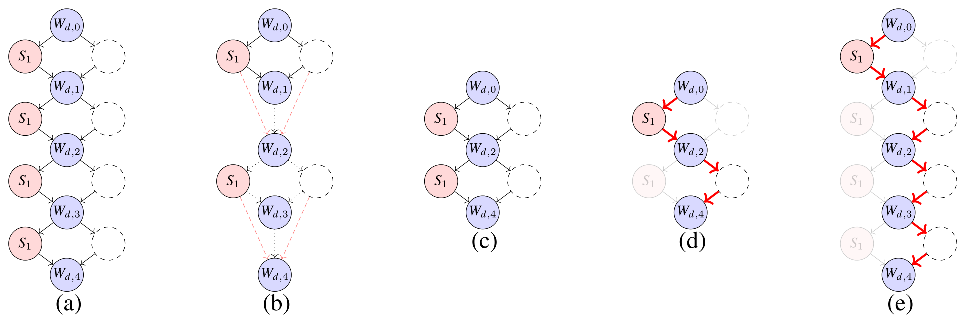


FIGURE 2. Collapsing of a scenario: (a) original problem definition, (b) removal of choices at non-anchor waypoints ($W_{d,1}$ and $W_{d,3}$), (c) collapsed problem, (d) solution for collapsed problem, (e) expansion to original problem.

a relatively large waypoint horizon. A stride parameter σ defines at which interval waypoints are marked as anchors. For example, for $\sigma = 3$, the anchor waypoints indices would be $\{0, 3, 6, 9, \ldots\}$. This approach is suitable for real-world scenarios where subsequent waypoints are positioned closely together, and the optimality of a solution does not affect the total mission execution time significantly.

By reframing the initial MILP in smaller, easier-to-solve MILPs, the computational complexity, and by extension the solving time, is reduced. The selection of the horizon size and rescheduling frequency impact the complexity and the number of smaller MILPs to solve respectively. In fact, our original problem definition is a specific case in which $\hat{N}_w = N_w$, $\sigma = 1$ and $\pi = \infty$. Furthermore, since the receding horizon MILPs are defined online during the mission execution, they take into account the effect of the environment up to that point, making the system more resistant to the dynamic environment.

C. MIXED-INTEGER LINEAR PROGRAM

The MILP definition from Section III, hereafter referred to as *base* model, can be used to express the two extensions with several modifications. We discuss the changes in the MILP formulation to accommodate the two extensions separately below, but these are combined with the base model in practise. For additional parameters introduced in this section, we refer the reader to the bottom section in Table 3 for an overview.

1) WAYPOINT HORIZON SCHEDULING

During the mission execution, it is necessary to keep track of which waypoints have been visited by which drone. Whenever a rescheduling is triggered, the positioning and battery statuses of the drones serve as the starting position ($W_{d,0}$) and the starting battery (β_d^{start}) for the drone. The waypoints used for the MILP construction are dependent on the progress of the drones along their route so far.

In the base model, the objective does not take into account the remaining route that needs to be traversed by the drones after the horizon. If not accounted for, an optimal solution for a given horizon problem may produce suboptimal results from a mission-wide point of view. The base objective aims to reduce the maximum execution time (Equation 2), which may not be relevant when one drone is fully done, whereas another drone still has many waypoints to go. We redefine the execution time (and objective) as follows:

$$\hat{E}_d = E_d + \lambda_d^{move} + \lambda_d^{charge}$$
$$\textbf{Minimize} \quad \max_d \hat{E}_d \quad \text{(Objective)} \tag{18}$$

where λ_d^{move} and λ_d^{charge} are estimations for the time that a drone d takes for moving and charging for the remainder of the mission. We estimate the remaining distance ρ_d as the sum of Euclidean distances of all consecutive waypoints starting from the last scheduled waypoint up to the last waypoint in the mission. This distance is considered a constant in solving the optimization problem. Note that this estimation excludes any extra distance added by diverging from this path to visit any charging station. We use this definition to calculate the expected remaining travel distance:

$$\lambda_d^{move} = \rho_d / v_d \tag{19}$$

The remaining estimated charging time is composed of the estimated depletion amount (erd_d) and the battery level that the drone has more than is required at the end of this horizon (oc_d):

$$erd_d = \lambda_d^{move} r_d^- \tag{20}$$
$$oc_d = b^*_{d,\hat{N}_w} - \beta^{min}_{d,\hat{N}_w} \tag{21}$$

The computation of λ_d^{charge} is as follows:

$$\lambda_d^{charge} = \max\{0, erd_d - oc_d\} \tag{22}$$

The rationale behind including the *overcharge* component oc_d is that without its occurrence, an optimal solution for the problem could (erroneously) result in drones ending with a fully charged battery, at the expense of a longer mission execution time.

Secondly, the base model does not take the distance to any charging stations after the last waypoint scheduled in the horizon into account. In the base model, it is acceptable that a drone arrives at the last waypoint with a minimal battery charge. For a limited horizon, this is not the case, since the drone must travel further after the horizon. Therefore, the minimum battery charge that a drone is allowed to carry at the last waypoint must be enough to reach a charging station afterward. For drones that do not finish their mission after a particular horizon, we add the following term to the minimum battery charge to the (previously) default minimum battery charge $\beta_d^{default}$ for the last scheduled waypoint ($\beta_{d,\hat{N}_w}^{min}$):

$$\beta_{d,\hat{N}_w}^{min} = \beta_d^{default} + \frac{\bar{r_d}}{v_d} \min_s \overrightarrow{\Delta}_{d,w,\hat{N}_w} \tag{23}$$

Thirdly, since rescheduling can interrupt drones at any point (including during charging), our model should preserve the imposed buffer between charging windows. As such, we introduce a parameter $\omega_{d,s}$ to define how long a drone D_d should wait *at least* at charging station S_s after the first waypoint (with $0 \leq \omega_{d,s} \leq \epsilon$). Cells in this matrix are only non-zero for drone D_d and charging stations S_s combinations for which the charging station was occupied by another drone $D_{d'} \neq D_d$ at the time of rescheduling. The values of these non-zero cells are equal to the separation window ϵ minus the time that the drone $D_{d'}$ had been charging there already. We introduce the following constraint:

$$I_{d,0} + \frac{1}{v_d} \sum_s \left[P_{d,s,0} \cdot \overrightarrow{\Delta}_{d,s,0} \right] \geq \sum_s \left[P_{d,s,0} \cdot \omega_{d,s} \right] \tag{24}$$

The left side of the inequality represents the time it takes drone D_d to start charging after the first waypoint and the right side the time required to wait.

2) ANCHOR WAYPOINTS

To facilitate anchor waypoints, we rely on the conversion of an existing MILP problem (either the base model or extended with the horizon changes) to another problem. Since non-anchor waypoints remove the option to charge from those waypoints, they can be fully encapsulated in a scenario with the non-anchor waypoints completely removed, and adding distances to the distance matrices $\overrightarrow{\Delta}_{d,n,w_s}$ and $\overleftarrow{\Delta}_{d,n,w_s}$. Fig. 2 shows the process of how an original scenario (2a) is collapsed into a smaller problem (2c). For example, the distance associated with the arrow between S_1 and $W_{d,2}$ in 2c represents the cumulative distance of S_2 to $W_{d,1}$ and $W_{d,1}$ to $W_{d,2}$ (red line in 2b). Besides adjusting the distance matrices (and the reduced number of waypoints due to the removal of the non-anchor waypoints), no further parameter changes to the collapsed MILP are necessary. After the collapsed problem is solved (2d), its solution is expanded back to the original problem (2e). For non-anchor waypoints, the drone moves directly to the next waypoints.

D. COMPLEXITY

From Table 4 and Equation 1 to 17 we can infer that the number of variables and constraints of the base model grows proportional to $\mathcal{O}({N_d}^2{N_w}^2 + N_dN_sN_w)$ respectively. The first component ($N_d^2{N_w}^2$) comes from the prevention of charging window overlap, whereas the second component ($N_dN_sN_w$) comes from the path decision variable P. The extensions to this base model turn a single optimization problem into multiple ones. The combination of horizon $\hat{N}_w$ and stride σ results in a reduction of the complexity as the variables and constraints do not grow proportional to the number of waypoint N_w, but rather with constant $\lceil \frac{\hat{N}_w}{\sigma} \rceil$. As a consequence, this turns the complexity into $\mathcal{O}({N_d}^2 + N_dN_s)$.

V. EVALUATION

We implemented our MILP model in Pyomo [34] and built a discrete-event simulator based on SimPy [35] around it for the evaluation of our proposed approach. A scenario (consisting of a series of waypoints and the positions of charging stations) and a set of parameters (i.e., the input for the MILP) serve as the configuration for the simulator. Based on these inputs, the behaviour of a centralized scheduler and the drones are simulated by generating discrete events. A scheduling *strategy* determines at which point in the simulation the scheduler produces a new schedule of charging station visits, which are distributed among the drones. An example strategy could be triggering a rescheduling process every time a drone reaches a waypoint or after a certain amount of simulated time. We use Gurobi 9.5.1 [36] to solve the individual optimization problems. The drones follow these schedules and generate events when (among others) they reach a new position (i.e., waypoint or charging stations), finish their waiting for another drone, start charging, etc. The behaviour of the simulator follows the configuration of the simulation perfectly; it does not deviate and is fully deterministic. By monitoring the behaviour of the scheduler, strategy, and drone, we extract relevant performance metrics from the simulator, most importantly the mission execution time, the time each schedule iteration has taken, and the number of schedule iterations. All simulations were performed on a 12-core i7-6800K CPU with 50 GB of RAM available.

Besides the implementation of our MILP formulation, we implemented a greedy scheduler (and strategy) as well, whose behaviour is highly short-term and non-coordinated (see Algo. 1); each drone will visit the closest charging station when it cannot reach a charging station after visiting the next waypoint. Line 7 results in the drone charging fully during the mission, except when it is about to reach the end of the mission (where it charges the minimum required charge to reach it). This prevents the batteries from 'overcharging' and unnecessarily increasing the mission execution time, thereby providing a fair comparison between scheduling strategies.

To illustrate the differences between the greedy and optimal scheduler, Fig. 3 shows a scenario (in 3a) in which two drones fly along a parallel path of four waypoints each, with three charging stations located in between them. Figs. 3b and 3c show the resulting greedy and optimal schedules respectively. The figure illustrates the precise timestamps of waypoint visits, charging window sizes, and how long the drones wait, as well as the development of the battery over time (the black lines). The greedy scheduler sends both

Algorithm 1 Greedy Scheduler

```
1: for $w_s \in \mathbb{W}_d$ do
2:     if any station reachable from $W_{d,w_{s+1}}$ after moving directly there from $W_{d,w_s}$ then
3:         move directly to next waypoint
4:     else
5:         move to closest station from $W_{d,w_s}$
6:         wait until available
7:         charge until end reachable or until battery is full
8:         move to next waypoint
9:     end if
10: end for
```

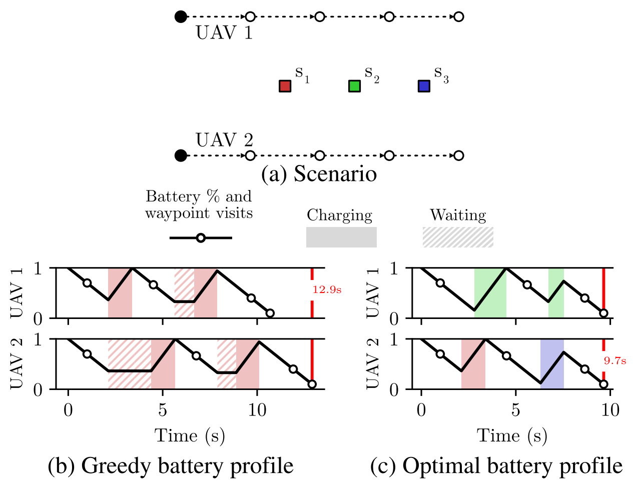


FIGURE 3. For the scenario in (a), the difference in scheduling between greedy (b) and optimal scheduling (c).

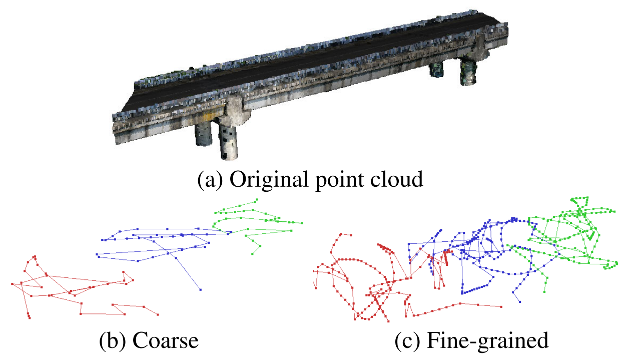


FIGURE 4. Conversion of a point cloud (a) to coarse paths (b) and fine-grained paths (c) of waypoints.

drones to the red charging station after both their first and second waypoint visit, resulting in the underutilization of the green and blue stations and congestion around the red station. The optimal scheduler instead, allocates the (middle) green station to one drone, and the red and blue charging station to the other, preventing any congestion. As a result, the second drone spends less time waiting for the drone to finish charging and reduces the total mission execution time by 3.2 seconds.

Simulation Setting: We use a point cloud obtained from a bridge segment and rely on prior work by Shi et al. [37] to extract realistic waypoint sequences. Their approach produces a series of waypoints from a point cloud by (1) representing a down-sampled version of the point cloud as a graph, (2) finding an efficient path in six subgraphs of the larger graph, and (3) optimizing the paths to produce a smooth trajectory. In order to accommodate a configurable number of drones (instead of a fixed six), we rely on an alternative partitioning process. Multilevel k-way hypergraph partitioning [38] is used to extract an arbitrary number of subgraphs of approximately equal size. The original point cloud used for this extraction and an example sequence of waypoints for three drones is depicted in Fig. 4. The coarseness of the down-sampling determines the number of waypoints to be inspected, as illustrated by the difference between (b) and (c), which contain 99 and 488 waypoints in total respectively.

Simulator Parameters: In the experiments, we resort to a set of default parameters that are used, unless the specific experiment denotes a different value. These values are as follows:

1) We use three drones for the inspection and have two charging stations positioned at ground level, at the side of the bridge, and in the middle of the bridge segment length-wise.
2) The speed of all drones is 0.15 m/s.
3) The battery levels are enforced between 20% and 100%.
4) The charge and depletion rates of the drones are set such that a full battery charges in 60 minutes and depletes in 10 minutes.
5) The separation between consecutive charging windows is set to 10 seconds.

The speed and minimum battery level are not necessarily representative of a realistic scenario, but these parameters force the drones to undergo at least one charging cycle in the mission execution which results in experiments in which the charge scheduling is evaluated fairly.

A. EXPERIMENTS

We conduct three sets of experiments to investigate different aspects of our scheduling approach. Firstly, we conduct a performance evaluation of our optimal, sub-optimal, and greedy schedulers. We then assess to what extent the utilization of charging stations impacts the performance gain between an optimal and greedy scheduler. Lastly, we turn to experimenting with different configurations of scheduling parameters to evaluate how these affect the performance of our proposed MILP model. The three experiments are geared towards providing insight into the capabilities and limitations of our approach for different configurations.

[I] How does an optimal solution compare with a suboptimal or greedy approach? As a first evaluation of our model, we assess the trade-off between mission

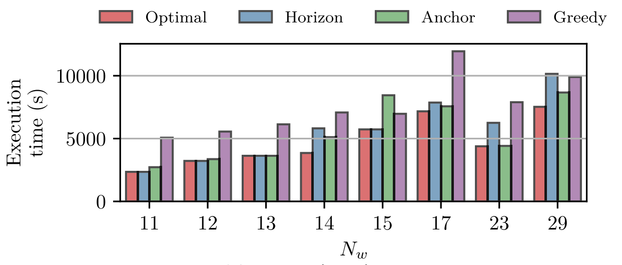
execution time (i.e., the optimality of the solution) and the required computation time. For this evaluation, we conduct a series of simulations for different complexities of scheduling problems. The complexity of the scheduling problem is determined by the number of scheduled waypoints, which in turn is a result of the coarseness of the down-sampling process. For each scenario, we run an experiment with four schedulers:

1) An optimal scheduler (i.e., our initial proposed solution)
2) A waypoint horizon scheduler ($\hat{N}_w = 15, \pi = 13$)
3) An anchor scheduler ($\sigma = 2$)
4) A greedy scheduler

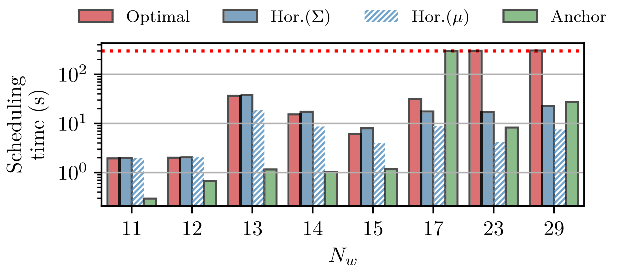
We vary the number of waypoints between 11 and 29, as the optimal solution becomes infeasible to calculate within a reasonable time for higher waypoint values. Fig. 5 illustrates the performance, measured as the mission execution time (a) and the corresponding time spent on scheduling (b).

Fig. 5a shows that the optimal scheduler outperforms the greedy scheduler in all cases, completing missions between 17.8% and 53.6% faster. Since the horizon scheduler uses a rescheduling frequency of 13, its performance up to $N_w = 13$ is guaranteed to be identical to the optimal solution. The horizon and anchor scheduler outperform the greedy scheduler in most cases, both performing worse in one experiment each. The occasional poor performance of the horizon scheduler can be attributed to its inability to take into account the scenario after the horizon, which may cause it to produce suboptimal short-term schedules in the context of the full mission. The anchor scheduler can be negatively affected in configurations where the exclusion of a particular waypoint from the charging scheduling process has a large impact (e.g., cases where this omitted waypoint is located close to a charging station in relation to other waypoints).

Fig. 5b depicts the scheduling time for the different schedulers for different complexities of inspection missions. The figure excludes the greedy scheduler, as the scheduling time is negligible. It further differentiates between the total scheduling time and average scheduling time per optimization problem for the horizon scheduler. Note that the y-axis is logarithmic. For both the optimal and anchor scheduler, we observe a general increasing trend of the scheduling time when the number of waypoints increases, and both hit the scheduling time limit at least once. This is not the case for the horizon scheduler, whose mean scheduling time remains fairly constant. For larger horizon sizes, the performance of both the optimal and horizon scheduler is expected to increase even more, making both of them infeasible. The horizon scheduler is expected to retain a low mean scheduling time, with the total scheduling time growing linearly over time. Since the scheduling can be performed online during the mission execution time, an increased total scheduling time will not affect the operations negatively. A realistic bridge inspection use case may consist of hundreds of waypoints to be visited and our results suggest that a



(a) Execution time



(b) Scheduling time. The dotted red line represents the solving time limit of 5 minutes. Σ = cumulative time spent scheduling during the mission, μ = average time spent per optimization problem.

FIGURE 5. Comparison of different scheduling strategies.

combination of the horizon and anchor scheduler is required for a well-performing strategy, balancing mission execution time and scheduling time.

[II] Charging station utilization We further investigate the impact of the percentage of time that a charging station is occupied, i.e., their *utilization*, on the performance gain from our proposed model compared to a greedy approach. The rationale behind this experiment is that the gains are expected to be only marginal whenever the utilization is either very low or very high. For the former, drones relying on a greedy approach are less likely to encounter an occupied charging station that requires them to wait. In the latter case, optimal scheduling is unlikely to be more effective than a greedy approach as drones are expected to wait on one another. The utilization of the charging station can be affected by (1) the ratio of the number of drones and charging stations, and (2) the ratio of the depletion and charging rate. We assess the difference in the mission execution time of our proposed approach and the greedy approach. We keep the number of drones and the depletion constant (3 and $\frac{1}{600}$, or a full depletion in 10 minutes, respectively) and vary the number of charging stations and charging rates to achieve a variation in the aforementioned ratios that affect the charging station utilization. The charging station count varies between one and three, and the charging rate between $\frac{1}{5400}$ (or fully charging in 1.5 hours) and $\frac{1}{300}$ (or fully charging in 5 minutes), corresponding a charging ratio of 9 and 0.5 respectively. For this experiment, we use a relatively coarse scenario, with 17 waypoints to be visited per drone, to allow the optimal solver to find a solution within a reasonable time. We further adjust the drone speed to 0.13 m/s to enforce at least two

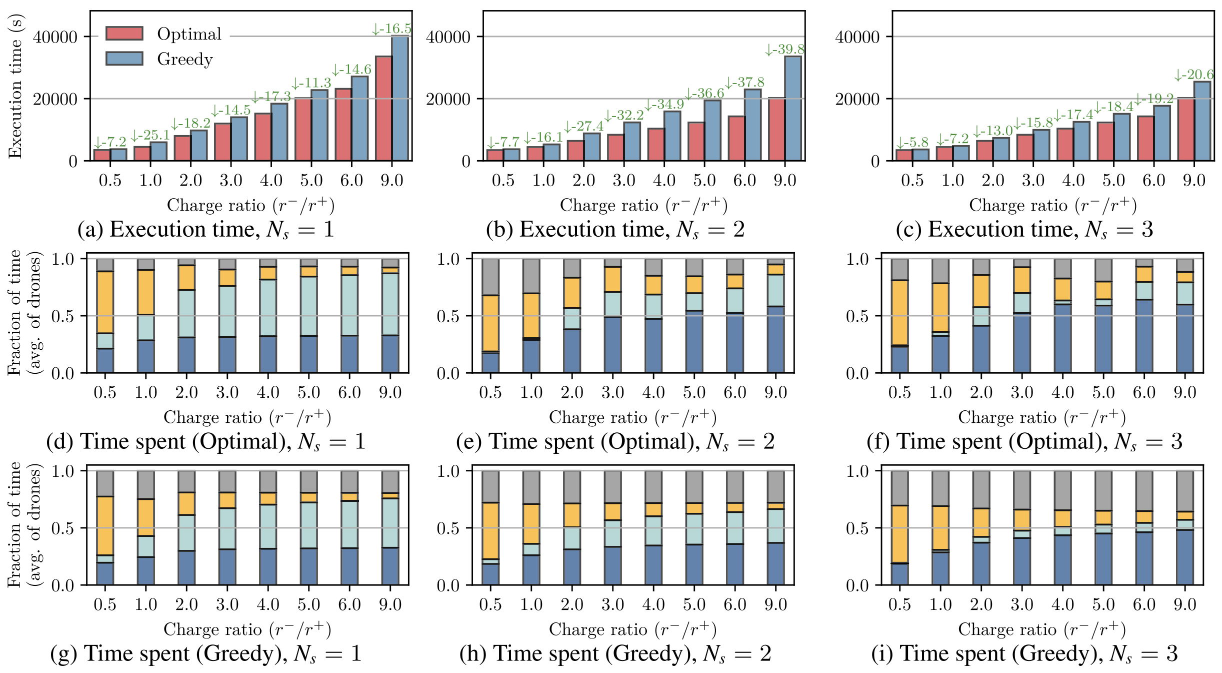


FIGURE 6. **Impact of charge ratio and the number of available charging station on the performance of the optimal and greedy scheduler. (a)–(c) = execution time, (d)–(f) = average fraction of time spent of optimal scheduler, (g)–(i) = average fraction of time spent of greedy scheduler. 🟦 = charging, ⬜ = waiting, 🟧 = moving, ⬛ = idle.**

charging cycles for each drone, to evaluate the impact of the charging station selection.

Fig. 6 shows the results of these experiments, depicting the execution times (6a–6c) and the average time spent (across the three drones) on different activities of the optimal (6d–6f) and greedy (6g–6i) scheduler for the various parameter combinations. The activities are divided into charging, waiting, moving, and idling (i.e., waiting for the mission to be completed by another drone). The first row shows that the optimal scheduler outperforms the greedy approach in all cases, ranging from a 5.8% to a 39.8% performance gain. Generally, a larger gain is achieved at higher charge ratios. For either scheduler, the behaviour of the drone with the longest individual execution time seems to be relatively unaffected by changes in the charge ratio, besides the duration of its activity. The optimal scheduler adapts the behaviour of the other drones instead to facilitate the slower drone at higher charge ratios, whereas the greedy scheduler does not.

The most significant gains are generated by the results for two charging stations. In these cases, the optimal scheduler takes advantage of leveraging both charging stations, whereas the greedy scheduler utilizes primarily one station only. The activity distribution differences (between the second and third row) reveal that the optimal scheduler results in far less time spent on idling than the greedy scheduler. By routing drones to charging stations that are farther away, the optimal scheduler avoids congestion at charging stations. As a result, it sacrifices the mission execution time of drones that finish earlier for the benefit of drones with a longer execution time, equalizing their individual execution time and thereby shortening the total mission execution time. This advantage diminishes with three charging stations, where the simple rules for the greedy scheduler end up distributing the three drones across three charging stations.

[III] What are ‘good’ scheduling parameters? So far, we have seen that a suboptimal solution has a larger potential of approximating the optimal solution than a greedy approach, but the effectiveness depends on the initial conditions of the problem configuration and the parameters. In this experiment, we evaluate a scheduler that combines both proposed extensions (waypoint and anchor scheduling). We run several simulations to identify how the optimality of the solution is sacrificed in favor of a faster scheduling time, for a variety of parameter combinations that affect the scheduling time. For these simulations, we use three drones and two charging stations where each drone is allocated 75 waypoints to follow. Given this configuration, we use the following sets of parameters, and run the experiment for each unique combination:

- $\sigma \in \{1, 2, 3, 4\}$
- $\hat{N}_w \in \{15, 30, 45, 60, 75\}$.
- $\pi \in \{8, 16, 31, 46, 61, \infty\}$

We excluded experiments with (1) an impossible combination of parameters i.e., where the rescheduling frequency π is larger than the horizon size $\hat{N}_w$ (except non-horizon schedulers with $\hat{N}_w = 75, \pi = \infty$) and (2) parameter combinations that result in too computationally expensive schedulers. We limit the stride σ to four as larger values end up with infeasible optimization problems, where at

TABLE 5. Performance metrics for different combinations of $\hat{N}_w$, σ and π. The upper section denotes the relative execution time compared to the best seen value (in %), the middle section the total scheduling time (in seconds) and the bottom section the average rescheduling time (in seconds).

		$\sigma = 1$	$\sigma = 2$			$\sigma = 3$					$\sigma = 4$				
	$\hat{N}_w$ / π	**15**	**15**	**30**	**45**	**15**	**30**	**45**	**60**	**75**	**15**	**30**	**45**	**60**	**75**
Relative Execution time (%)	**8**	16.5	**20.6**	**6.8**	**0.2**	**33.7**	14.2	7.0	7.6	7.0	**23.2**	16.3	9.9	9.9	9.9
	16			10.8	19.2		7.2	7.0	7.0	7.0		15.5	9.9	9.9	9.9
	31				**0.0**			7.0	7.0	7.0			9.9	9.9	9.9
	46								7.0	7.0				9.9	9.9
	61									7.0					9.9
	∞									7.0					9.9
Total scheduling time (s)	**8**	104	53	330	**2912**	**7**	460	777	**1708**	1594	**6**	331	**1604**	878	452
	16			78	644		311	130	520	669		310	636	650	260
	31				320			80	143	223			24	36	107
	46								88	162				**22**	67
	61									157					60
	∞									132					56
Mean rescheduling time (s)	**8**	6	**3**	18	**150**	**0**	24	43	**96**	91	**0**	17	76	46	22
	16			9	84		52	17	63	77		39	63	91	29
	31				**132**			23	22	30			5	8	13
	46								12	30				11	11
	61									25					4
	∞									–					–

least one drone ends up with two consecutive anchor waypoints whose distance is too large to cover with the given battery constraints. The range of horizon sizes covers short to mission-wide scheduling scenarios. The rescheduling frequencies cover a similar path, in most cases being one higher than the values of the horizon size to ensure the horizon size and rescheduling frequency are not too close to one another. For each of the parameter combinations, we collect the execution time and the scheduling time (total across the full mission and the average for each rescheduling after the initial) for the simulation. Table 5 shows the result of the simulations. The columns denote the combination of the horizon size $\hat{N}_w$ and stride σ parameters and the rows show the different rescheduling frequencies π and the three metrics. Excluded experiments have empty fields in the table. For each section in the table – corresponding to the three metrics – we denote the three worst and best-performing parameter configurations in red and green respectively.

The upper part of the table shows the percentual difference compared to the fastest-seen execution time. The best value is found for $\hat{N}_w = 45$, $\sigma = 2$, and $\pi = 31$, and the worst configuration is 33.7% slower. For stride values of 3 or 4 it appears that an optimum (for the given stride value) has been found with several of the configurations (which are 7.0% and 9.9% slower than the fastest execution time respectively). It suggests that a small stride value of 2 gives the scheduler access to better-positioned waypoints to follow up with a charging station visit. Varying the scheduling frequency π or horizon $\hat{N}_w$ can have an unpredictable impact. For $\sigma = 2$ and $\hat{N}_w = 45$, changing the frequency from the best performing $\pi = 31$ to $\pi = 16$ results in an execution time increase of 19.2%, whereas going down to $\pi = 8$ results only in a 0.2% slower execution time. In the MILP formulation of the horizon scheduling extension, an estimated depletion is taken into account, which cannot take into account future charging station visits. Configurations with different rescheduling policies or horizon size values will - throughout a mission - have access to a different view of the mission after their horizon and over time their generated schedules can diverge over time.

The middle section of the table denotes the total scheduling time in seconds. A lower total scheduling time can be achieved by either lowering the horizon (i.e., decreasing $\hat{N}_w$), increasing the stride σ or reducing the number of reschedule instances (i.e., increasing π). An alternative perspective is the time that is required for rescheduling *during* a mission, or all optimization problems except the initial one. These are namely the schedules that must be generated ad hoc during the mission and are time-critical. We report the average time for these values in the bottom section of the table. For $\pi = \infty$, no rescheduling is performed, so these fields in the table are denoted by '–'. These numbers reveal that the lowest execution time comes at the cost of one of the slowest rescheduling times. A trade-off of a low computation time and a decent execution time can be found, for example where $\sigma = 4$, $\hat{N}_w = 45$, and $\pi = 31$. In that situation, the rescheduling time on average is 5 seconds, while still having a 9.9% slower execution time.

VI. DISCUSSION

Our three experiments have shown the potential of our approach over a greedy approach. By tweaking the relevant scheduling parameters of the model, one can find a balance between optimality and scheduling time. The performance is highly dependent on the initial configuration of the parameters and charging station positions and parameters. Moderately large problems take tens of seconds to schedule, making it non-practical to run the scheduling on the drone hardware itself and therefore there is a need for scheduling on a more powerful ground control station. We relied on the stride parameter to select waypoints that served as anchors in the scheduling. Alternatively, other methods could be far more effective, for example prioritizing waypoints close to a charging station as anchor waypoints. Although we compensate in the redefined objective (Equation 18) for the remainder of the mission, our current approach might not be sufficient. It does not take into account any extra moving time (and its associated charging time) which can be substantial when charging stations are not readily available. In our experiments, we provided a comparison of the scheduling time between our base model and the extensions. One supposed benefit of our proposed horizon extension is that it handles a dynamic environment that deviates from the configuration and parameters of the MILP model, but this benefit remains unexplored.

We consider this work as one important step in drone fleet design, one that allows drones to operate indefinitely and significantly contributes to future fully autonomous systems. Even though we present our results through the lens of a bridge inspection use case, there are other promising use cases. Our proposed method can be translated to other types of structures, such as railways or building inspection. For each structure, the positioning of waypoints and charging stations will be unique, so we recommend the research community assess the efficacy of our method for other use cases. Our method can be adapted to other similar scheduling problems, such as the allocation of a charging network across a fleet of electric vehicles in a country. Furthermore, repeated experiments under different parameter configurations can be used to find appropriate values for the number of drones to use or the number and positioning of charging stations. However, more efficient solving techniques may be necessary to combat the computational complexity that a fleet of even ten vehicles would impose.

For future work, we intend to resort to alternative methods for solving optimization problems, such as genetic algorithms [39], particle swarm optimization [40], or reinforcement learning [41], as they have been demonstrated to efficiently solve optimization problems in similar application areas. Furthermore, we plan to implement our method in drone hardware and evaluate its feasibility and performance in the much more dynamic and volatile real world. We also intend to address the unreliable nature of a communication network that allows the drone and mission control to communicate.

VII. CONCLUSION

The organisation of multiple drones into autonomous fleets comes with an increased complexity and challenges. One of these challenges is the charging strategy for the battery-limited drones, when provided with a limited number of battery recharging station. In this paper, we propose a mathematical formulation for a centralized charge scheduling optimization problem. Solving the model produces a set of allocated charging windows that specify when a given drone is supposed to charge at a given charging station. We tackle the computational complexity by proposing two extensions to the problem. Through experimentation with a discrete-event simulator on a real-world bridge inspection example, we show that our proposed method significantly outperforms a greedy scheduler for a variety of parameter configurations. Our efforts to combat the computational complexity of solving the model result in two extensions, which prove to provide a balance between scheduling time and minimizing the mission execution time. We believe that our model has more far-reaching use cases than the evaluated bridge inspection mission, including other drone swarming application but also scheduling problems for fleets of electrical vehicles.

APPENDIX
PARAMETER DEFINITIONS

We define C_d^{max} in Equation 11 as:

$$C_d^{max} = \frac{\beta_d^{max} - \beta_{d,w}^{min}}{v_d} \tag{25}$$

We define I_d^{max} in Equation 12 as:

$$I_d^{max} = \sum_{d' \neq d} \Big[\frac{1}{v_d} \sum_{w_s} \left(\max_s \left[\overrightarrow{\Delta}_{d',s,w_s} + \overleftarrow{\Delta}_{d',s,w_s} \right] \right) + (N_w - 1)(C_{d'}^{max} + \epsilon) \Big] \tag{26}$$

The first and second term in Equation 26 represent the longest possible travel time and longest charging duration for all drones for each of their waypoints respectively. We define the 'sufficiently large' M (used in Equations 16 and 17) as:

$$M = \sum_d I_d^{max} \tag{27}$$

REFERENCES

[1] T. Benarbia and K. Kyamakya, "A literature review of drone-based package delivery logistics systems and their implementation feasibility," *Sustainability*, vol. 14, no. 1, p. 360, Dec. 2021.

[2] Y. Karaca, M. Cicek, O. Tatli, A. Sahin, S. Pasli, M. F. Beser, and S. Turedi, "The potential use of unmanned aircraft systems (drones) in mountain search and rescue operations," *Amer. J. Emergency Med.*, vol. 36, no. 4, pp. 583–588, Apr. 2018.

[3] R. H. Jacobsen, L. Matlekovic, L. Shi, N. Malle, N. Ayoub, K. Hageman, S. Hansen, F. F. Nyboe, and E. Ebeid, "Design of an autonomous cooperative drone swarm for inspections of safety critical infrastructure," *Appl. Sci.*, vol. 13, no. 3, p. 1256, Jan. 2023.

[4] K. Máthé and L. Buşoniu, "Vision and control for UAVs: A survey of general methods and of inexpensive platforms for infrastructure inspection," *Sensors*, vol. 15, no. 7, pp. 14887–14916, Jun. 2015.

[5] B. Chen and X. Miao, "Distribution line pole detection and counting based on YOLO using UAV inspection line video," *J. Electr. Eng. Technol.*, vol. 15, no. 1, pp. 441–448, Jan. 2020.

[6] Y. Wu, Y. Qin, Z. Wang, and L. Jia, "A UAV-based visual inspection method for rail surface defects," *Appl. Sci.*, vol. 8, no. 7, p. 1028, Jun. 2018.

[7] A. Phadke and F. A. Medrano, "Towards resilient UAV swarms—A breakdown of resiliency requirements in UAV swarms," *Drones*, vol. 6, no. 11, p. 340, Nov. 2022.

[8] (2022). *Mavic 3—Specs*. [Online]. Available: https://www.dji.com/dk/mavic-3/specs
[9] 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," *J. Intell. Robotic Syst.*, vol. 70, nos. 1–4, pp. 347–359, Apr. 2013.
[10] B. D. Song, J. Kim, J. Kim, H. Park, J. R. Morrison, and D. H. Shim, "Persistent UAV service: An improved scheduling formulation and prototypes of system components," *J. Intell. Robotic Syst.*, vol. 74, nos. 1–2, pp. 221–232, Apr. 2014.
[11] J. Kim and J. R. Morrison, "On the concerted design and scheduling of multiple resources for persistent UAV operations," *J. Intell. Robotic Syst.*, vol. 74, nos. 1–2, pp. 479–498, Apr. 2014.
[12] B. D. Song, K. Park, and J. Kim, "Persistent UAV delivery logistics: MILP formulation and efficient heuristic," *Comput. Ind. Eng.*, vol. 120, pp. 418–428, Jun. 2018.
[13] H. Ghazzai, A. Kadri, M. Ben Ghorbel, H. Menouar, and Y. Massoud, "A generic spatiotemporal UAV scheduling framework for multi-event applications," *IEEE Access*, vol. 7, pp. 215–229, 2019.
[14] J. Kim, S. Kim, J. Jeong, H. Kim, J.-S. Park, and T. Kim, "CBDN: Cloud-based drone navigation for efficient battery charging in drone networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 11, pp. 4174–4191, Nov. 2019.
[15] M. Shin, J. Kim, and M. Levorato, "Auction-based charging scheduling with deep learning framework for multi-drone networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4235–4248, May 2019.
[16] H. Ghazzai, H. Menouar, A. Kadri, and Y. Massoud, "Future UAV-based ITS: A comprehensive scheduling framework," *IEEE Access*, vol. 7, pp. 75678–75695, 2019.
[17] V. Hassija, V. Saxena, and V. Chamola, "Scheduling drone charging for multi-drone network based on consensus time-stamp and game theory," *Comput. Commun.*, vol. 149, pp. 51–61, Jan. 2020.
[18] C. H. Liu, C. Piao, and J. Tang, "Energy-efficient UAV crowdsensing with multiple charging stations by deep learning," in *Proc. IEEE IEEE Conf. Comput. Commun. (INFOCOM)*. Toronto, ON, Canada: IEEE, Jul. 2020, pp. 199–208. [Online]. Available: https://ieeexplore.ieee.org/document/9155535/
[19] R. G. Ribeiro, J. R. C. Júnior, L. P. Cota, T. A. M. Euzébio, and F. G. Guimarães, "Unmanned aerial vehicle location routing problem with charging stations for belt conveyor inspection system in the mining industry," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 10, pp. 4186–4195, Oct. 2020.
[20] V. Hassija, V. Chamola, D. N. G. Krishna, and M. Guizani, "A distributed framework for energy trading between UAVs and charging stations for critical applications," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5391–5402, May 2020.
[21] G. Ahani, D. Yuan, and Y. Zhao, "Age-optimal UAV scheduling for data collection with battery recharging," *IEEE Commun. Lett.*, vol. 25, no. 4, pp. 1254–1258, Apr. 2021.
[22] M. Y. Arafat and S. Moh, "JRCS: Joint routing and charging strategy for logistics drones," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21751–21764, Nov. 2022.
[23] M. Torky, M. El-Dosuky, E. Goda, V. Snásel, 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. [Online]. Available: https://www.mdpi.com/2504-446X/6/9/237
[24] R. Pinto and A. Lagorio, "Point-to-point drone-based delivery network design with intermediate charging stations," *Transp. Res. C, Emerg. Technol.*, vol. 135, Feb. 2022, Art. no. 103506. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X21004927
[25] K. Wang, X. Zhang, L. Duan, and J. Tie, "Multi-UAV cooperative trajectory for servicing dynamic demands and charging battery," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1599–1614, Mar. 2023.
[26] D. Schermer, M. Moeini, and O. Wendt, "Algorithms for solving the vehicle routing problem with drones," in *Proc. Asian Conf. Intell. Inf. Database Syst.* Cham, Switzerland: Springer, 2018, pp. 352–361.
[27] J. Pasha, Z. Elmi, S. Purkayastha, A. M. Fathollahi-Fard, Y.-E. Ge, Y.-Y. Lau, and M. A. Dulebenets, "The drone scheduling problem: A systematic state-of-the-art review," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 14224–14247, Sep. 2022.
[28] M. Alighanbari. (2004). *Task Assignment Algorithms for Teams of UAVs in Dynamic Environments*. [Online]. Available: http://hdl.handle.net/1721.1/17754
[29] C. G. Grlj, N. Krznar, and M. Pranjić, "A decade of UAV docking stations: A brief overview of mobile and fixed landing platforms," *Drones*, vol. 6, no. 1, p. 17, Jan. 2022.
[30] A. Boggio-Dandry and T. Soyata, "Perpetual flight for UAV drone swarms using continuous energy replenishment," in *Proc. 9th IEEE Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Nov. 2018, pp. 478–484.
[31] J. Meng, G. Luo, and F. Gao, "Lithium polymer battery state-of-charge estimation based on adaptive unscented Kalman filter and support vector machine," *IEEE Trans. Power Electron.*, vol. 31, no. 3, pp. 2226–2238, Mar. 2016.
[32] M. Asghari, A. M. Fathollahi-Fard, S. M. J. Mirzapour Al-e-hashem, and M. A. Dulebenets, "Transformation and linearization techniques in optimization: A state-of-the-art survey," *Mathematics*, vol. 10, no. 2, p. 283, Jan. 2022.
[33] W. H. Kwon and S. H. Han, *Receding Horizon Control: Model Predictive Control for State Models*. London, U.K.: Springer, 2006.
[34] W. E. Hart, J.-P. Watson, and D. L. Woodruff, "Pyomo: Modeling and solving mathematical programs in Python," *Math. Program. Comput.*, vol. 3, no. 3, pp. 219–260, Sep. 2011.
[35] Team SimPy. (2022). *SimPy—Discrete Event Simulation for Python*. [Online]. Available: https://simpy.readthedocs.io/en/latest/
[36] Gurobi Optimization. (2023). *Gurobi Optimizer Reference Manual*. [Online]. Available: https://www.gurobi.com
[37] L. Shi, G. Mehrooz, and R. H. Jacobsen, "Inspection path planning for aerial vehicles via sampling-based sequential optimization," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2021, pp. 679–687.
[38] G. Karypis and V. Kumar, "Multilevel k-way hypergraph partitioning," *VLSI Design*, vol. 11, no. 3, pp. 285–300, Jan. 2000.
[39] M. K. Amjad, S. I. Butt, R. Kousar, R. Ahmad, M. H. Agha, Z. Faping, N. Anjum, and U. Asgher, "Recent research trends in genetic algorithm based flexible job shop scheduling problems," *Math. Problems Eng.*, vol. 2018, pp. 1–32, 2018.
[40] T.-L. Lin, S.-J. Horng, T.-W. Kao, Y.-H. Chen, R.-S. Run, R.-J. Chen, J.-L. Lai, and I.-H. Kuo, "An efficient job-shop scheduling algorithm based on particle swarm optimization," *Expert Syst. Appl.*, vol. 37, no. 3, pp. 2629–2636, 2010.
[41] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takáč, "Reinforcement learning for solving the vehicle routing problem," 2018, *arXiv:1802.04240*.

KASPAR HAGEMAN received the Ph.D. degree in analysis of the domain name system, digital certificates and network traffic for security from Aalborg University, Denmark, in 2021. He is currently a Postdoctoral Researcher with Aarhus University. His research interests include linear optimization and reinforcement learning for the coordination and scheduling of drone fleets. His work was awarded the IETF/IRTF Applied Networking Research Award in 2017.

RUNE HYLSBERG JACOBSEN (Senior Member, IEEE) received the master's degree in science (physics and chemistry) and the Ph.D. degree in laser physics and optoelectronics from Aarhus University, Denmark, in 1997. He is a currently a Professor with the Department of Electrical and Computer Engineering, Aarhus University. His professional career spans more than 14 years in the telecommunications and IT industry, where he has been responsible for the management of research and development. He has published more than 85 peer-reviewed journal and conference papers in areas of communications and its applications. His research interests include computer networking, wireless communications, network security, cooperative intelligent systems, and satellite communications.

• • •