Transportation Research Part E 198 (2025) 104095

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

Data-driven optimization for drone delivery service planning with online demand

Aditya Paul [a], Michael W. Levin [b], S. Travis Waller [c], David Rey [d,*]

[a] School of Civil and Environmental Engineering, UNSW Sydney, NSW, 2052, Sydney, Australia
[b] Department of Civil, Environmental, and Geo-Engineering, University of Minnesota, 55455, Minneapolis, USA
[c] Lighthouse Professorship "Transport Modelling and Simulation", Faculty of Transport and Traffic Sciences, Technische Universität, Dresden, Germany
[d] SKEMA Business School, Université Côte d'Azur, Sophia Antipolis, France

ARTICLE INFO

Keywords:
Data-driven optimization
Online demand
Markov decision process
Drone delivery
Service planning

ABSTRACT

In this study, we develop an innovative data-driven optimization approach to solve the drone delivery service planning problem with online demand. Drone-based logistics are expected to improve operations by enhancing flexibility and reducing congestion effects induced by last-mile deliveries. With rising digitalization and urbanization, however, logistics service providers are constantly grappling with the challenge of uncertain real-time demand. This study investigates the problem of planning drone delivery service through an urban air traffic network to fulfill dynamic and stochastic demand. Customer requests – if accepted – generate profit and are serviced by individual drone flights as per request origins, destinations and time windows. We cast this stochastic optimization problem as a Markov decision process. We present a novel data-driven optimization approach which generates predictive prescriptions of parameters of a surrogate optimization formulation. Our solution method consists of synthesizing training data via lookahead simulations to train a supervised machine learning model for predicting relative link priority based on the state of the network. This knowledge is then leveraged to selectively create weighted reserve capacity in the network and via a surrogate objective function that controls the trade-off between reserve capacity and profit maximization to maximize the cumulative profit earned. Using numerical experiments based on benchmarking transportation networks, the resulting data-driven optimization policy is shown to outperform a myopic policy. Sensitivity analyses on learning parameters reveal insights into the design of efficient policies for drone delivery service planning with online demand.

1. Introduction

Unmanned Aerial Vehicles (UAVs, also known as drones) have been growing in both prevalence and application. UAVs provide the advantage of performing a task aerially with limited or no human supervision. Additionally, most UAV designs are light-weight and can be deployed in large numbers. These features make UAVs highly desirable in many different domains, particularly logistics. UAV applications broadly fall into two categories: (a) data acquisition and transmission (also called monitoring, or reconnaissance), or (b) payload pickup and delivery. Initially developed for the former, advances in UAV technology – in terms of payload capacity,

* Corresponding author.
E-mail addresses: aditya.paul@unsw.edu.au (A. Paul), mlevin@umn.edu (M.W. Levin), steven_travis.waller@tu-dresden.de (S.T. Waller), david.rey@skema.edu (D. Rey).

https://doi.org/10.1016/j.tre.2025.104095
Received 28 July 2024; Received in revised form 17 March 2025; Accepted 19 March 2025
Available online 31 March 2025

flight times and altitude capability – have made them suitable for the latter (Hecken et al. (2022), Kornatowski et al. (2020), Mohsan et al. (2023)).

This study falls into the second category of payload delivery and addresses the drone delivery service planning (DDSP) problem under online demand. This problem framework is motivated by the observation that the widespread application of UAVs coincides with an equally widespread surge in online logistic demand. Online demand, sometimes referred to as on-demand logistics, deals with that aspect of e-commerce which provides an affordable delivery service to satisfy the customer's place and time requirements at short notice. With the consolidation of the requisite digital mechanisms (internet ubiquity, personal computers, mobile phones, online payment apps) and supply chain infrastructure (warehouses, highways, enterprise resource planning, GPS), e-commerce has experienced a consistent surge in online demand in the 21st century. E-commerce is now experiencing over 10% growth per annum worldwide, and B2C e-commerce deliveries rose by nearly 25% in 2020 alone (Lozzi et al., 2022). In addition, the disruption caused by COVID-19 has caused an acceleration in internet usage and a lowering of traditional offline purchases, both factors contributing further to e-commerce home deliveries (Han et al., 2022).

E-commerce and online demand present both opportunities and challenges. While e-commerce signals growth for society, the unpredictable real-time demand it generates can be difficult to meet. Both academic literature and commercial enterprise have delved into the deployment of UAVs to fulfill uncertain real-time demand. Prime Air by Amazon and Project Wing by Alphabet have been pioneering applications of UAVs for home deliveries (Chen et al., 2021). DHL initiated a project named Parcelcopter to deliver medical supplies by UAVs (Benarbia and Kyamakya, 2021). Food delivery has also witnessed UAV incorporation, such as by Ele.me, a subsidiary of Alibaba (Na and Chang, 2024). Most notably, the company Zipline played a crucial role in delivering medical supplies, including but not limited to blood donations, via UAVs during the COVID pandemic in Africa (Ackerman and Koziol, 2019). UAVs are cheaper to maintain, reduce labor and delay costs and have a lower carbon footprint compared to trucks. They have promising potential for logistics, particularly in moderate density urban regions and to deliver time-sensitive packages. Additionally, a majority of customers in urban locations live close to a supermarket, and most delivery packages are relatively light-weight (Moshref-Javadi and Winkenbach, 2021). This includes food, beverages, groceries and also medical and pharmaceutical supplies, all of which have a perishability constraint which makes UAVs an attractive choice.

In light of this, we investigate the DDSP with online demand. The DDSP takes the perspective of a transportation network manager that plans service for drone deliveries. We consider discrete time steps grouped into consecutive time intervals. At each time step, customers may submit delivery requests that are to be served by a fleet of UAVs subject to time windows and link capacity constraints. The probability distributions behind request parameters, and the request arrival process, are unknown to the network manager and estimated from historical data. Decisions are made at every time interval to accept or reject the requests that arrive and to route accepted requests. We explore the potential of data-driven optimization methods for this dynamic stochastic DDSP problem.

In stochastic optimization problems, some or all parameters are subject to uncertainty. These parameters are not deterministic, but instead emerge from probability distributions or uncertainty sets. Traditional solution methods to stochastic problems exploit *a priori* knowledge of these probability distributions. Stochastic problems can further be sub-divided into two categories: (a) static, where all problem parameters are available beforehand, and some problem specific condition determines if and when the stochastic parameters are instantiated; (b) dynamic, where problem parameters arrive sequentially and data is revealed over time. Besides, online optimization makes no assumption on the nature of uncertain data and focuses on the immediacy and real-time nature of incoming problem parameters. Online problems are always dynamic in nature. Nevertheless, in many cases, it is reasonable to presume that the probability distribution, or an estimate thereof, can be extracted either through historical data or predictive models (Van Hentenryck et al., 2010). From a computational standpoint, stochastic and online optimization problems lead to significant challenges. The rise of e-commerce has compelled logistic service providers to tackle online demand. In this study, we consider the setting of the static DDSP in the context of online demand, i.e. user requests that are revealed over time.

Our solution methodology is based on an innovative data-driven optimization and lies at the intersection of predictive analytics (machine learning) and prescriptive analytics (stochastic optimization). The literature on the combination of machine learning (ML) and stochastic optimization reveals two distinct avenues for a successful collaboration:

- ML-enabled optimization: Combinatorial optimization problems can be very hard to solve, and special techniques or methods can be used to solve a particular kind of problem or problem instance. Instead of relying on experts to create such techniques, algorithms and models can be developed to automatically learn from implicit distribution of problem instances and adapt to those instances. ML has been found to be useful here.
- Prediction and optimization: Many real-world practical applications involve uncertain parameters which directly or indirectly affect the objective function, but which must first be predicted before optimization. In such predict-then-optimize situations, ML is being recognized as a vital tool to make accurate data-driven predictions.

Research into transportation service planning under online demand, with a dedicated integration of ML and mathematical programming, is still in its nascent stages. In this study, we incorporate strategies from both these avenues to develop a customized data-driven optimization approach for the DDSP with online demand. This study makes the following contributions to the field.

1. We extend the integer linear programming (ILP) formulation of the deterministic DDSP to incorporate online demand, i.e. delivery requests are revealed over time and routing decisions must be made under uncertainty. We also cast this problem as a Markov Decision Process (MDP).

2. We propose a novel solution method for this dynamic stochastic DDSP where predictive prescriptions are computed over time. Historical demand data including spatial and temporal distributions, as well as the request arrival process are used to determine link-based features that inform a surrogate optimization formulation.
3. We demonstrate through numerical experiments that our data-driven optimization approach outperforms a myopic optimization approach. Sensitivity analysis reveal how learning parameters affect optimal policies for the DDSP with online demand.

The rest of the paper is organized as follows. In Section 2, we review the relevant literature. In Section 3, we present the DDSP with online demand and introduce the MDP framework along with ILP formulations. Section 4 develops the solution methodology and the proposed algorithms. Section 5 reports the numerical experiments and their results. Our findings and research perspectives are discussed in Section 6.

2. Literature review

We examine the state-of-the art relevant to the online DDSP. Section 2.1 reviews recent advances in data-driven optimization, notably the interplay between ML and optimization. Section 2.2 records the advent of stochastic and online demand as well as the latest solution approaches arising in urban logistics. Finally, Section 2.3 outlines the widespread adoption of UAVs in commercial and industrial sectors.

2.1. Data-driven optimization

ML has proved useful for optimization purposes along two lines: to assist directly within the optimization process by exploiting particular problem characteristics, or as a data-driven predictive framework to estimate uncertain parameters that are involved in the optimization formulation. We review these two streams of literature hereafter.

2.1.1. Machine learning-enabled optimization

It is common in optimization that a solution algorithm is tuned to a specific problem scenario by refining its parameters or policy to that scenario. Such tuning (or adaptation) need not come from research expertise and intuition. ML can assist an optimization algorithm to this end (Bengio et al., 2021). This direct assistance provided by ML can be analyzed based on its motivation or its structure. In terms of motivation, ML can either: (a) replace some heavy computation with a fast, reliable approximation or (b) explore the decision space (defined by the optimization model) and discover better policies. As for structure, the collaboration between ML and optimization can be sequential, whereby the ML model supports the optimization algorithm with valuable information. Or it can be parallel, wherein the ML and optimization components work alongside each other. If one of the components works within the framework of the other, then the solutions of the nested component help the external one make better decisions.

Kruber et al. (2017) provide an illustration of the sequential implementation. They use supervised ML to decide if a given Dantzig-Wolf decomposition will be effective in solving instances of a mixed-integer linear programming (ILP) problem. Bonami et al. (2018) address a similar question. They employ supervised ML to determine if linearizing a given instance of a mixed integer quadratic programming problem will reduce solve time. An example of parallel implementation is found in Lodi and Zarpellon (2017). The ML model works within a branch-and-bound framework, and is used to make decisions about branching. Specifically, the goal of the ML component is to learn an efficient policy for selecting the branching variable and branching direction. Yilmaz and Büyüktahtakın (2024) address sequential combinatorial optimization problems in industrial contexts, where problem instances arise with slight parameter variations. A neural network generates a relaxed problem instance by predicting binding constraints, then uses this relaxed instance to eliminate infeasible predictions of decision variables through rapid feasibility checks. This reduces solution time by up to three orders of magnitude, while maintaining an optimality gap below 0.1%. Larsen et al. (2022) introduce a supervised ML model to quickly predict expected tactical descriptions of operational solutions in stochastic optimization. This addresses a common challenge in two-stage stochastic programming, where the second stage is computationally demanding. Their approach aids in solving the overall two-stage problem by circumventing the need for online generation of multiple second stage scenarios and solutions. Another implementation possibility is to leverage an optimization formulation to aid an ML algorithm in the solution process. Tran-Thanh et al. (2012) employ an unbounded knapsack formulation to improve the performance of the ϵ-greedy algorithm for multi-armed bandits, where the knapsack solution governs the probability of pulling a given arm of the bandit. A comprehensive overview of ML-enabled optimization methods is reported in Bengio et al. (2021).

2.1.2. Predict-then-optimize

Many decision-making problems deal with both prediction and optimization (or decision). These two tasks are complex on their own and are hence often treated separately and sequentially. The prediction model predicts the unknown parameters of the optimization model, which the latter uses to perform the optimization. ML has been a popular candidate for the prediction task in recent times. But the conventional approach is to train the ML prediction model to minimize the prediction error. This does not consider the nature and attributes of the downstream optimization problem and may result in sub-optimal decisions (Elmachtoub and Grigas, 2022). As a brief but standard example, consider two possible paths between an origin O and a destination D. Travel times for both paths are unknown to the decision-maker, but may be predicted with contextual data (weather, congestion, etc.).

Suppose the actual travel times are 10 and 20 min each, respectively. ML model 1 predicts travel times on the two paths to be 15 and 13 min (respectively), while ML model 2 predicts the same to be 30 and 40 min. Although model 1 has better accuracy, it causes a worse decision.

Two frameworks have been proposed which integrate prediction with optimization by taking the downstream optimization model into account (Yan and Wang, 2022). The first framework is called smart "predict, then optimize" (SPO), and it evaluates the performance of the ML prediction model based on the final decision error induced by the prediction made, not on the initial prediction error (Elmachtoub and Grigas, 2022). The authors propose a design which applies to the optimization of linear objective functions over a convex feasible region. They introduce the SPO loss function which captures the decision error of a prediction by utilizing the objective function of the downstream optimization model. Since training with the SPO loss function can be cumbersome, a more manageable surrogate loss function called SPO+ loss is also introduced, which is equivalent to the original loss under specific conditions. The second framework, predictive prescriptions, aims to harness the joint distributions between decision variables and auxiliary data, both of which are uncertain Bertsimas and Kallus (2020). The principle of predictive prescriptions is to simulate various scenarios in which the values of the unknown parameters are estimated by the ML prediction model. These estimations lead to a cost in the optimization process, and the costs are weighted by the data-driven exploration of the scenarios. The optimal decision policy then minimizes the weighted sum costs thus generated. Both SPO and predictive prescriptions framework have been increasingly adopted by the scientific community and a survey of contextual optimization methods for decision-making under uncertainty is available in Sadana et al. (2023).

2.2. Stochastic and online demand

As defined in Section 1, online demand refers to the sale of immediately deliverable products via e-commerce, which has grown significantly in recent decades. Early research highlighted the benefits of transitioning from brick-and-mortar to online retail (Kim and Park, 2005), leading to multi-channel and later omni-channel models that integrate physical and online sales (Verhoef et al., 2015). E-commerce investments enabled market expansion, customer loyalty, and data-driven personalization at low capital costs (Fruhling and Digman, 2000). We next discuss the impact of online demand on vehicle routing, both in terms of emerging obstacles and recent solution approaches. We then describe solution strategies which incorporate ML, in line with the methodologies outlined in Section 2.1.

2.2.1. Online demand in vehicle routing

The benefits of e-commerce come with the challenges of online demand. By its dynamic and uncertain nature, online demand is difficult to fulfill. Failed first deliveries can account for up to 60% of all orders for a service provider (Song et al., 2009), the primary reason being the absence of the customer during package handover. This is partially due to modern lifestyle changes such as increase in single person households and flexible (but demanding) work patterns. But a major contributing factor also is that no delivery time slot is explicitly agreed upon between the customer and the e-commerce retailer. This leads to several detrimental effects. If packages are left unattended then lack of security is a concern. Repeated deliveries or customers personally traveling to collection points result in economic losses as well as excess carbon emissions (Edwards et al., 2010). A high rate of unsatisfied or partially satisfied demand also strains driver-customer relationships (Masorgo et al., 2023). However, if a delivery time window is agreed upon through some form of online demand management, then the success of last mile deliveries is shown to increase (Van Duin et al., 2016).

Many studies dealing with online demand employ some form of heuristic approach to modify routes as and when new demand requests emerge. Hong et al. (2019) consider a shipping company that faces online demand, but delivers products to pre-determined collection centers that the customer must visit on their own. They implement an ant colony based heuristic which changes the previously decided collection center for a particular customer in light of incoming demand requests. Mobility services also fall under stochastic and online demand, where the commodity sold is the route itself. Bertsimas et al. (2019) analyze online taxi ride-sharing with pickup time windows for passengers. They employ a cost metric to measure waiting or empty driving time between customers. The network is pruned by keeping, for each customer, the top k subsequent customers with the lowest costs. A maxflow heuristic is applied by reducing pickup time windows to a single randomly selected time instant. The resulting route arcs are retained, and the process is iteratively repeated until a specified number of arcs is reached. Finally, a mixed-integer formulation is applied to the modified network for obtaining results. Consolidation of delivery requests can also save costs by revealing efficient routes. Muñoz-Villamizar et al. (2024) explore stochastic demand where the city of operation is divided into regions, each with an expected customer density, estimated average velocity and travel distance per order. A region-specific postponement cost function is proposed for quantifying the cost of the delay of a request in a given region to a later date. A tabu-search metaheuristic modifies vehicle routes by identifying better (lower cost) routes in their local neighborhood. The neighborhood is traversed by applying insertion, swap and inversion operators that transform the customer sequence of a vehicle route.

2.2.2. ML-based solution methods

Using ML to assist optimization, as discussed in Section 2.1.1, has garnered significant attention in the transport literature. The same-day delivery problem is a prime example of such joint implementations. Chen et al. (2023) study same-day delivery under fairness, dividing the service area into regions, and define service availability as the ratio of expected accepted requests per day to the expected overall number of requests. The objective is to maximize both overall service availability and the minimum service availability across all regions. The learning agent receives rewards for each fulfilled request, improving its policy by considering

changes in both components of the objective function. Feng et al. (2022) apply reinforcement learning to real-time ride-sourcing, incorporating public transport services. For every customer ride request, feasible routes are heuristically generated, and for every driver, the learning agent estimates the perceived value of route-driver pairs. An integer linear program then determines the optimal allocation of drivers to routes. If electric vehicles are involved, charge depletion adds uncertainty. Basso et al. (2022) study the electric vehicle routing problem where customer request arrivals and energy consumption per unit road length are stochastic, while charging locations are deterministic. The reinforcement learning agent estimates the energy cost of moving to possible subsequent states and the probability of battery charge falling below a safety threshold at each state of the MDP formulation. Liu et al. (2022) use deep reinforcement learning to dispatch vehicles for an online ride-hailing platform. They divide the service area into grids and define the supply–demand gap of each grid as the difference between number of ride requests and number of vacant vehicles. They use this supply–demand gap to reallocate vacant vehicles to other grids pre-emptively, preventing drivers from being stuck in low-demand "coldspots" and thereby escaping local optima in the solution space.

The predict-then-optimize frameworks discussed in Section 2.1.2 have also been applied in stochastic transport optimization. Chu et al. (2021) implement the SPO method to solve the last-mile problem for an online food delivery service. They consider driver behavior, traffic conditions and route lengths as input features to predict travel times. Simulated annealing is used to generate feasible routes, which are further updated with customer exchange operators to derive new routes. Baty et al. (2024) design a machine learning pipeline which relies on a downstream optimization component to improve predictions. The authors study the dynamic vehicle routing problem with time windows (VRPTW), where all requests must be fulfilled and fleet size is unlimited. They demonstrate that solving an equivalent prize-collecting VRPTW, where each unfulfilled customer is assigned a prize or profit, addresses this version of the VRP. An ML model determines the optimal allocation of prizes, yielding a near-optimal solution for the original problem. A hybrid genetic search algorithm solves the prize-collecting VRPTW based on input from the ML component. The ML model refines itself through a loss function, which measures the disparity between the target and acquired objective function values. The target objective value is derived by solving a static VRPTW from historical data, utilizing the same hybrid genetic search algorithm.

2.3. UAV for urban logistics

As mentioned in Section 1, UAVs applications are broadly classified into traffic monitoring or reconnaissance, and payload pickup and/or delivery. This paper belongs to the latter category. In payload delivery, there are many UAV models in the literature. UAVs can either be deployed independently, or in association with conventional vehicles such as trucks (with or without synchronization). This paper focuses on the former, i.e. the deployment of a homogeneous fleet of UAVs without ground vehicular support. However, for completeness, we delve into the relevant literature for both drone-only and truck and drone logistics, in that order.

2.3.1. Drone-only routing

Levin and Rey (2023) explore the drone delivery service planning problem and propose a novel branch-and-price algorithm for its solution. They also design a reservation heuristic to generate feasible solutions expediently. Vlahovic et al. (2017) perform a case study of a UAV delivery model for pharmaceutical supplies, exhibiting cost reductions and improvements in service responsiveness. Retail facility placements can be improved when network design is conducted under the availability of a UAV delivery service (Baloch and Gzara, 2020). Liu (2019) propose a mixed integer programming formulation for online meal delivery using UAVs. The objective function penalizes excess battery depletion, motivates fast delivery and minimizes unnecessary movement for re-routing. The model is executed in a progressive rolling fashion across discrete time-steps. Chen et al. (2021) explore key strategic and tactical decisions for retailers implementing drone-based delivery operations, focusing on when to offer drone delivery and the appropriate pricing. Using an MDP framework, they introduce heuristic procedures to obtain near-optimal closed-form solutions efficiently. The aim is to assist online retailers in real-time decision-making regarding the feasibility and extent of offering drone-based delivery for specific product categories in various service zones. Additionally, they identify effective pricing strategies for drone-based deliveries. Payload delivery expands outside the realm of delivery to consumers. For example, UAVs can be used in precision agriculture, to simultaneously monitor crop sites and perform spraying tasks. This increases the efficiency of both pesticide and fertilizer use (Radoglou-Grammatikis et al., 2020).

The study of drone-only logistics also attracts ML-enriched solution strategies. Asadi et al. (2022) study drone delivery of medical supplies to hospitals, categorizing demand based on hospital distance from the drone hub. Each demand class includes a demand distribution and the required battery charge for successful delivery. The problem is formulated as an MDP where the system state is a vector containing the number of batteries at specific charge levels within the hub. The concise state expression allows reinforcement learning where a lookup table suffices for value function approximation. The action space involves charging batteries from lower to higher levels within decision epochs. This model optimizes battery usage, preserving charge for future deliveries and reducing replacement and charging costs. UAVs also have potential as a supplementary resource in two-echelon logistics networks. Li et al. (2024) study the last-mile delivery problem, where trucks transport goods from a primary depot to satellite depots, and UAVs complete the delivery to customers. We categorize this study as drone-only logistics since trucks are treated as a stochastic externality. The stochastic nature of parcel arrival times at satellites is handled through Bayesian estimation, converting continuous arrival scenarios into discrete expected arrival intervals. A two-stage stochastic model is applied: parcel consolidation is first performed to release a subset for immediate delivery, and the remaining parcels are reserved for routing with future arrivals. Next, the route is treated as a particle in a particle swarm optimization problem. Reinforcement learning using search operators incrementally permutes the route in each iteration to discover better routes in the local neighborhood.

2.3.2. Truck and drone logistics

The integration of UAVs into the logistic service fleet (together with conventional vehicles) can result in considerable benefits (Rejeb et al., 2023). Such a multi-modal delivery model decreases delivery times, overcomes traffic congestion and minimizes human participation in the delivery process. Dayarian et al. (2020) investigate a same-day delivery model where a fleet of trucks is assisted by UAVs (through commodity resupply) at mutually determined meeting locations, allowing trucks to make longer trips before returning to the depot. Gu et al. (2023) investigate a single truck-and-drone scenario for online demand, consisting of deterministic delivery requests along with online pickup requests. The drone revisits the truck for resupply when needed, and an insertion heuristic compares the costs of placing pickup customers at various segments of the initial route. A 15% increase in profits is reported, with UAVs contributing a 50% increase in the acceptance rate of online requests. Yang et al. (2023) study a robust drone–truck delivery problem with customer time windows. They leverage the precision of drone routes to mitigate uncertainties in truck travel times caused by ground traffic congestion. Unlike Gu et al. (2023), here the truck remains at a customer location until the drone serves other customers and returns. Numerical experiments show that the drone–truck combination can safe-guard against the risk of failed deliveries due to unmet time windows. Lemardelé et al. (2021) use continuous approximation analysis to compare various last mile delivery services from a strategic lens. One of these services is the deployment of UAVs from heavy commercial vehicles (HCVs), which depart a distribution center equipped with parcels, batteries and drones. The HCVs arrive at a distribution zone, where the UAVs complete the final parcel delivery to drop-boxes or parcel lockers, and customers retrieve their parcels from the lockers as per their convenience. This approach demonstrates viability for large service regions with moderate to low demand density. Rave et al. (2023) expand the truck–drone framework by introducing deployable drone launching stations, known as microdepots. In this setup, trucks depart from a central distribution center (CDC) carrying both onboard drones and microdepots. Along their routes, trucks deploy the microdepots at strategic locations, where they serve as transshipment nodes to reach customers out of range of the CDC. Drones can be launched directly from trucks or from deployed microdepots. Their experiments show that mixed fleets offer cost savings compared to truck-only fleets.

Heterogeneous fleets logistics, such as truck and drone models, also welcome ML-supported solutions. Chen et al. (2022) handle a mixed fleet of vehicles and UAVs to handle online requests. Network evolution is modeled as an MDP. Routing heuristics determine route feasibility by vehicle or drone, and deep reinforcement learning is then used to determine if a vehicle or a drone should be dispatched. Ghiasvand et al. (2024) tackle a two-echelon multi-trip truck–drone routing problem with uncertainty. Trucks make multiple trips from the depot to local distribution centers (LDCs), while UAVs make single trips from LDCs to customers. A mixed-integer linear programming model is formulated to minimize total customer waiting times, managing uncertain customer demand through kernel-function based machine learning algorithms.

It is evident from the literature that the intersection of ML and combinatorial optimization is an active area of research, though applications to dynamic and stochastic scenarios are still relatively new. But existing contributions often depend on specialized models or heuristics that require extensive adaptation for new use cases. Frequently, the application of ML is in the form of deep learning or reinforcement learning, where the ML model operates independently of the underlying structure of the optimization problem. These approaches often prioritize predictive accuracy or decision guidance without explicitly integrating the problem's domain-specific characteristics. Motivated by these observations, this paper introduces optimization policies for the DDSP with online demand, where the network operates under very small decision epochs. This formulation is solved via a novel data-driven solution methodology, which relies on predictive prescriptions of a network resource (i.e., link capacity) to assist the optimization model. Notably, the integration of ML and optimization proposed in this paper explicitly incorporates the problem's domain-specific characteristics. The methodology leverages ML predictions to directly inform the optimization process, ensuring both interpretability and adaptability in model execution.

In urban scenarios, the operation environment of UAVs can mimic the road network. Buildings and other infrastructure impede flight paths or require excessive energy to overcome aerially. But the airspace above roadways is generally free of obstructions. A possible drone flight network can be a map of the road traffic network itself, but projected at a suitable distance above the ground, where links are above roads and nodes are above traffic intersections. Several flight levels may exist (Levin and Rey, 2023). We adopt this context for the DDSP with online demand.

3. Problem description and formulation

We now formally introduce the problem setting of the dynamic and stochastic DDSP. The objective is to maximize profit by accepting and servicing transport requests arriving dynamically over a given time horizon. We describe the problem setup for the DDSP, including the dynamic stochastic customer demand, in Section 3.1. We next present the MDP formulation that captures this problem in Section 3.2. Model constraints and an ILP formulation of the DDSP with online demand is presented in Section 3.3 and a myopic ILP formulation is introduced along with its algorithmic framework.

3.1. Problem description

Let $G = (N, A)$ be a network of N nodes and A links across which customers send transport requests. These requests, if accepted, are fulfilled by a fleet of drones. Every link $a \in A$ has a length ρ_a and capacity C_a. All links are directed. It is assumed that drones can depart only once, and that upon departure, the drone immediately climbs to a suitable flight level, where it traverses the network through these links at a constant velocity v. Each node n has a set of incoming and outgoing links Γ_n^- and Γ_n^+ respectively. The time horizon is discretized into time steps $\{0, 1, \dots, t, \dots, T\}$, where T marks the end of the horizon. Each customer request r consists of (a)

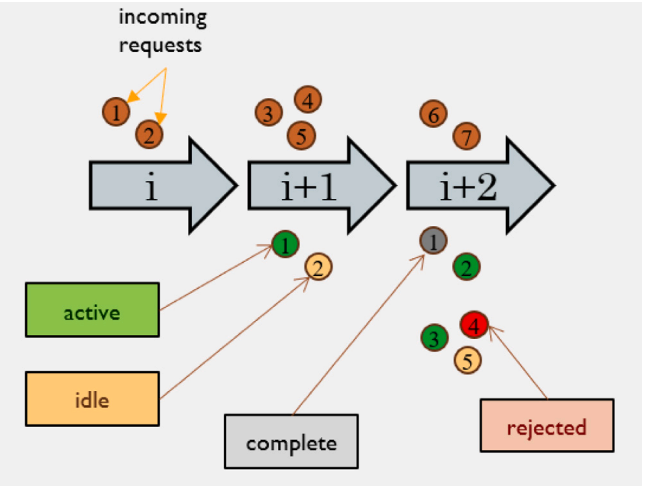


Fig. 1. Online demand model: the circles represent requests whereas $i, i+1, i+2$ are time intervals. The brown circles are incoming requests in an interval, whose status becomes either active, idle, rejected or complete in later intervals.

spatial parameters – the request origin o_r and destination d_r such that $o_r, d_r \in N$, (b) temporal parameters – the earliest permissible time of departure e_r and the arrival time window at the destination $[l_r, u_r]$ and (c) a profit p_r earned upon fulfilling this request. In many logistics service scenarios, such as freight operations, customers have the option to pay extra for expedited delivery services, which grants their requests higher priority. The use of differential pricing in customer requests reflects this varying priority. This approach aligns with the application of drones in delivering not only e-commerce orders, but also critical payloads such as medical supplies, where priority and speed are paramount. The set of all requests that arrive over the entire time horizon is denoted by $\mathcal{R}$.

In the dynamic stochastic demand model, the time horizon is divided into I intervals, each of duration D, such that $DI = T$. The successive execution of all I intervals concludes a single instance (or day) of network operation. The number of requests arriving at the network in each interval is governed by a Poisson process with rate λ. Hence, on average, the expected total number of requests submit to the network during the entire time horizon is λI. Within a specific interval, the submissions of customer requests are assumed to have exponential inter-arrival times. This demand model is illustrated in Fig. 1. Customers submit requests to the network during each interval $i \in I$. These current incoming requests in interval i (denoted by $\mathcal{R}_C^i \subset \mathcal{R}$) can either be accepted or rejected — based on which their status is updated in the next interval. Accepted requests are added to the set $\mathcal{R}_{\checkmark} \subset \mathcal{R}$ and must then be routed through the network. Once accepted (assigned a drone and a planned route), a request cannot later on be rejected or left unfulfilled. As soon as a drone initiates its route within the network, the corresponding request is termed *active*.

If a request is accepted and routed but its planned route begins after the current interval, the request is classified as *idle*. While the proposed route for an idle request is included in the solution, it does not take effect until the corresponding drone starts its flight trajectory. Until the planned route is initiated, the idle request is carried over to subsequent intervals, during which its planned route may be adjusted through re-routing. Once the drone departs and the idle request transitions to active, no further route changes are allowed. Once departed, the idle request becomes active and no route changes are permitted. Let $\mathcal{R}_I \subset \mathcal{R}_{\checkmark}$ and $\mathcal{R}_A \subset \mathcal{R}_{\checkmark}$ be the sets of idle and active requests respectively. Drones are assumed to complete deliveries, return to their origins and drop off the network. Upon route completion the corresponding request is serviced successfully and is removed from $\mathcal{R}_A$ and $\mathcal{R}_{\checkmark}$. The problem objective is to maximize the total profit earned by servicing requests in the given time horizon.

3.2. Formulation as a Markov decision process

The dynamic evolution of the network (arrival and acceptance of requests, followed by drone traffic routing and optimization) can be formulated as a Markov Decision Process (MDP). In an MDP, the problem is decomposed into an *environment* and an *agent*. In the DDSP with online demand, the network is the environment, and the network manager (who accepts and routes requests) is the agent. The environment produces *states*, based on which the agent takes *actions*, to receive *rewards*. The agent decides the best action at a given state. The environment subsequently controls the transition to the next state.

3.2.1. State space

Customer requests arrive dynamically during network operation. The arrival or submission of a request triggers a new state of the MDP. Each such state of the MDP captures all relevant information about the current situation of the network, at the time at which the state is created. When a new request r is submitted to the network, the state s_r is triggered, which consists of the following components:

1. t_r: The time step at which the new request r arrives. This triggers the creation of a new state. Note that two successive requests may arrive several time steps apart i.e., $t_{r+1} - t_r \geq 1$.
2. $\vec{r}$: The request vector $\vec{r}$ contains details of the new request r. These are: the request sequence number, origin node, destination node, earliest departure time, time window of arrival at the destination, and request profit. We write $\vec{r} = (r, o_r, d_r, e_r, [l_r, u_r], p_r)$.
3. $\vec{\Theta}_r^{pre}$: The collective route vector $\vec{\Theta}_r^{pre}$ contains the routes of all requests (both active and idle) which arrived before the new request r and were accepted for service. The subscript r in $\vec{\Theta}_r^{pre}$ indicates that the collective route vector captures the planned routes for all idle requests and the ongoing routes of all active requests which arrived prior to request r (and were accepted). Note that the collective route vector $\vec{\Theta}_r^{pre}$ *does not* contain a route for the new request r, because this request has just arrived and triggered a new state, but no action has yet been taken for this request. Consider a request r' which arrived before request r and was accepted. The individual route for this previous request r' is denoted by $\vec{\theta}_{r'}$. This vector $\vec{\theta}_{r'}$ reveals the request sequence number, the time of drone departure, and sequence of nodes in the route $(r', t_{o_{r'}}, [o_{r'}, \dots, d_{r'}, \dots, o_{r'}])$. For sake of brevity, the sequence of nodes in the route is also denoted by $\mu_{r'} = [o_{r'}, \dots, d_{r'}, \dots, o_{r'}]$. It can be known if the request r' is active or idle based on whether departure from the origin lies within the current interval or without. If a request r' becomes active, its route vector $\vec{\theta}_{r'}$ is immune to any further change. If a request r' is fulfilled by the time the new state is triggered, its corresponding $\vec{\theta}_{r'}$ is removed from $\vec{\Theta}_r^{pre}$.

Hence the state s_r is represented as the tuple $s_r = (t_r, \vec{r}, \vec{\Theta}_r^{pre})$. Every interval witnesses the creation of as many states as there are request arrivals in that interval. Between the acceptance (and subsequent routing) or rejection of a request and the arrival of the successive request, the only change in the network is the movement of active drones as per their routes. Since this is a deterministic change (the routes of active drones are known and fixed), a new state need not be declared until the arrival of the next request. Thus the evolution of drone traffic in the network can be represented as a sequential MDP moving through successive states. The state space in the MDP framework outlines the uncertain input information, capturing the temporal and spatial attributes of incoming requests, their uniformly distributed profit values, and the stochastic arrival process. By delineating these sources of uncertainty, the MDP formulation characterizes the structure of this optimization problem under uncertainty and provides a framework for developing models and solution methods.

3.2.2. Action space

After the arrival of a new request r triggers the creation of state s_r, the agent must take an action a_r from the feasible action space $\mathcal{A}(s_r)$. The action space allows the rejection of request r, but also contains all possible ways in which request r can be routed through the network, provided five conditions are met:

1. The drone must not exceed the capacity of a link pre-occupied by other active drones.
2. The drone must not, at any time-step, utilize a turn junction which is already utilized by an active drone at that time-step.
3. The drone must not depart the request origin before e_r.
4. The drone must reach the request destination within the time window $[l_r, u_r]$.
5. The drone must return to the request origin from where it was launched.

The action space $\mathcal{A}(s_r)$ also permits re-routing idle requests from previous intervals in order to accommodate request r or to achieve a better objective value. The action taken is then represented as the tuple $a_r = (z_r, \vec{\Theta}_r^{post})$ where z_r is a binary decision regarding request r and $\vec{\Theta}_r^{post}$ is the *updated* collective route vector, which would include a route $\vec{\theta}_r$ for request r if $z_r = 1$. At a given state, action selection is done by the agent's **policy**, which maps states to actions. Conventionally, a policy is represented as $\pi(a_r|s_r)$, and governs the probability of taking action a_r when in state s_r.

The action space in the MDP framework circumscribes the set of feasible decisions, inherently defining the structural constraints of the problem. By formalizing the conditions under which requests can be accepted and routed – such as link capacity limits, conflict resolution at junctions, and temporal feasibility – the MDP ensures that subsequent mathematical modeling represents operational constraints.

3.2.3. Reward and value functions

Accepting a request increases the value of the objective function and should therefore indicate a reward. The reward received in state $s_r = (t_r, \vec{r}, \vec{\Theta}_r^{pre})$ with action $a_r = (z_r, \vec{\Theta}_r^{post})$ is simply:

$$R(s_r, a_r) = z_r p_r \tag{1}$$

This gives a reward of p_r for accepting a request and a reward of 0 for rejecting it. Consequently, the reward accumulated over the course of interval $i \in I$ is given by $\sum_{r \in \mathcal{R}_C^i} R(s_r, a_r) = \sum_{r \in \mathcal{R}_C^i} z_r p_r$. The overall cumulative reward is $\sum_{t_r \in [0,T]} z_r p_r$, which is equivalent to $\sum_{r \in \mathcal{R}} z_r p_r$.

The state–action pair (s_r, a_r) captures the process of taking an action a_r while in a state s_r. In the application of ML under the MDP framework, every state–action pair is assigned a relative value in comparison to all other pairs. This value indicates the expected final objective (or reward) that can be achieved given that the agent occupies state s_r and takes action a_r. The learning model presented in this study seeks to estimate the value of taking a certain action while occupying a certain state. In other words, it seeks to discover and exploit the *action value function* $Q(s_r, a_r)$. Thus "learning" is equivalent to improving the accuracy of the value estimations of $Q(s_r, a_r)$ to a satisfactory degree.

3.2.4. Dynamic information evolution and state transition

The density of request arrivals across the time horizon follows a Poisson process with rate λ. The origins and destinations of requests are determined by spatial distributions, while their time windows for arrival at destinations are governed by a temporal distribution. Additionally, request profits are drawn from a profit distribution.

Once an action has been taken in a certain state, the transition to the next state occurs under transition probabilities $P^{a_r}_{s_r \to s'_r}$. The transition probability determines which state the environment jumps to once the agent undertakes a specific action in the current state. There are many application scenarios where the transition probability matrix is either unknown to the agent, or the size of the state space prohibits its use. In such scenarios the agent can only control its policy, and has no prior knowledge of how the environment will react to its actions. Despite this being true for the DDSP, it can still be said that the transition probabilities will be affected both by factors exogenous to the network manager (namely, the spatial and temporal distributions of customer requests, and their dynamic arrival to the system), as well as by the decisions themselves of the network manager. The acceptance or rejection of certain requests will influence the feasibility of acceptance of other requests in the future.

This study assumes the availability of historical demand data, and the epochs over which actions are taken and information evolves are of the order of a few minutes, making the problem online. Even though any available history of demand may provide some stochastic insights into the problem, the size of the state space and the complex interplay between various stochastic parameters make the explicit computation of transition probabilities impractical.

3.2.5. Objective function and optimal policy

The objective is to maximize the cumulative profit earned from requests served over the entire horizon. This is identical to maximizing the cumulative reward acquired in the MDP formulation. In MP terminology, this becomes:

$$\max \sum_{r \in \mathcal{R}} z_r p_r \tag{2}$$

Given an (accurate) action value function $Q(s_r, a_r)$, an optimal policy π^* would aim to choose action a_r in state s_r such that the state–action pair (s_r, a_r) returns the maximum achievable Q-value in that state, i.e.:

$$\pi^*(a_r|s_r) = \begin{cases} 1, & \text{if } a_r = \arg\max_a \ Q(s_r, a) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

This is equivalent to stating that the policy, out of the set of all policies Π, that maximizes the objective function, i.e. the expected total reward gained over the course of the time horizon, is π^*:

$$\pi^* \in \arg\max_{\pi \in \Pi} \ \mathbf{E}\left[\sum_{r=1}^{|\mathcal{R}|} R(s_r, A^{\pi}_r(s_r)) \Big| s_{r-1}\right] \tag{4}$$

Here $A^{\pi}_r(s_r)$ is the action taken by policy π in state s_r. The state s_0 depicts the initial empty network where no request has yet arrived. Note that the limits of summation in (2) and (4) are slightly different because, unlike in (2), in (4) we emphasize operating upon requests sequentially to demonstrate the evolution of the MDP.

Eq. (4) establishes the influence of the reward function on policy design. The reward function is based solely on requests accepted and the corresponding profit acquired at each decision step. Alternative reward formulations, i.e. ones that account for broader network conditions, such as future congestion, capacity allocation, or request deferral, could be used in accordance with the decision maker's preferences.

3.3. Model constraints and ILP formulations

We consider the online demand data arrives in batches and we set out to solve the DDSP with online demand via a rolling horizon approach where decisions are taken periodically over time upon receiving new demand data. The deterministic (or static) version of the DDSP is examined by Levin and Rey (2023), who propose a link-based network formulation and use it to construct an ILP that is solved over the entire time horizon. In an online demand context, problem data must be updated between decision epochs. This requires that, in a given interval, the constraints extend from the beginning of that interval to its end. Hence the limits of t for interval i are given by $[(i-1)D, iD]$. For example, if interval duration $D = 5$ minutes, then the first interval spans from $t = 0$ to $t = 5$ minutes, and so on. In each interval i, the model must handle the current incoming requests of that interval ($\mathcal{R}^i_C \subset \mathcal{R}$), as well as the idle requests from previous intervals ($\mathcal{R}_I \subset \mathcal{R}_{\checkmark} \subset \mathcal{R}$). Let $\mathcal{R}^i_O = \mathcal{R}^i_C \cup \mathcal{R}_I \ \forall i \in I$ denote the union of current incoming requests and idle requests. In each interval i, the model will be applied to the request set $\mathcal{R}^i_O$. All the constraints discussed hereafter in this section are presented for some interval $i \in I$, implying that these constraints apply to the request set $\mathcal{R}^i_O$, for time interval $[(i-1)D, iD]$.

Every link is divided into an upstream and downstream segment, as shown in Fig. 2, which also depicts the set of incoming links to the first node n_1 and the set of outgoing links from the second node n_2. The key decision variable $\chi^{r\uparrow}_a(t) \in \{0, 1\}$ indicates whether the drone carrying request r occupies the upstream portion of link $a \in A$ at time t. The counterpart of $\chi^{r\uparrow}_a(t)$ for the downstream

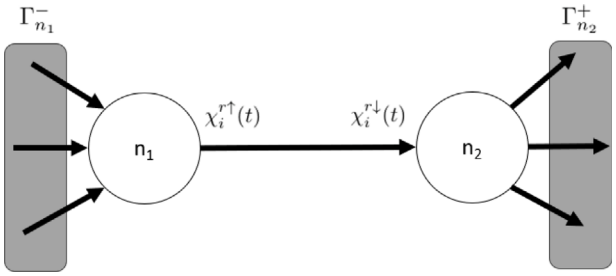


Fig. 2. Link-based model.

segment is $\chi_a^{r\downarrow}(t)$, which indicates whether the drone responsible for request r occupies the downstream portion of link $a \in A$ at time t. The number of drones occupying a link (either upstream or downstream) is limited by the link capacity:

$$\sum_{r \in \mathcal{R}_O^i} \chi_a^{r\uparrow}(t) \leq C_a \qquad \forall a \in A, \forall t \in [(i-1)D, iD] \tag{5a}$$

$$\sum_{r \in \mathcal{R}_O^i} \chi_a^{r\downarrow}(t) \leq C_a \qquad \forall a \in A, \forall t \in [(i-1)D, iD] \tag{5b}$$

For any request, the difference of the upstream and downstream variables $\chi_a^{r\uparrow}(t)$ and $\chi_a^{r\downarrow}(t)$ represents the time taken to traverse the concerned link. Hence, the time at which a drone occupies the downstream portion of a link is greater than the time at which it occupies the upstream portion, by an amount equal to the link traversal time:

$$\chi_a^{r\downarrow}\left(t + \frac{\rho_a}{v}\right) = \chi_a^{r\uparrow}(t) \qquad \forall r \in \mathcal{R}_O^i, \forall a \in A, \forall t \in [(i-1)D, iD] \tag{6}$$

The drone servicing request r must enter a given node n from one if its incoming links Γ_n^- and exit the node from one of its outgoing links Γ_n^+ which translates into the following flow conservation constraints:

$$\sum_{a \in \Gamma_n^-} \chi_a^{r\downarrow}(t) = \sum_{a \in \Gamma_n^+} \chi_a^{r\uparrow}(t) \qquad \forall r \in \mathcal{R}_O^i, \forall n \in N, \forall t \in [(i-1)D, iD] \tag{7}$$

The equalities in Eqs. (6) and (7) require that drones travel precisely at speed v. Waiting or decelerating to allow passage of conflicting traffic is prohibited. Drone routes must be calculated for continuous forward motion. Some types of UAVs, such as fixed-wing ones, would not be able to hover in place or decelerate easily. Moreover, maintaining constant velocities helps save energy by avoiding frequent acceleration and deceleration, which extends battery life and increases operational range (Gong et al., 2023; Shivgan and Dong, 2020).

Two successive links connected by a node is called a *turn*. The decision variable $\gamma_{ab}^r(t) \in \{0,1\}$ indicates whether request r occupies the turn from link a to b at time t. Note that the movement of the drone from the downstream segment of link a to the upstream segment of link b through the connecting node is assumed to be instantaneous. Since $\gamma_{ab}^r(t) \in \{0,1\}$ dictates whether the drone (servicing request r) makes the turn from link a to link b, we have $\gamma_{ab}^r(t) = 1$ only if both $\chi_a^{r\downarrow}(t) = \chi_b^{r\uparrow}(t) = 1$. Conversely, if $\gamma_{ab}^r(t) = 0$, then the turn from a to b is prohibited, meaning both $\chi_a^{r\downarrow}(t)$ and $\chi_b^{r\uparrow}(t)$ cannot together be 1. The following three constraints characterize valid turns:

$$\gamma_{ab}^r(t) \leq \chi_a^{r\downarrow}(t) \qquad \forall r \in \mathcal{R}_O^i, \forall (a,b) \in A^2, \forall t \in [(i-1)D, iD] \tag{8a}$$

$$\gamma_{ab}^r(t) \leq \chi_b^{r\uparrow}(t) \qquad \forall r \in \mathcal{R}_O^i, \forall (a,b) \in A^2, \forall t \in [(i-1)D, iD] \tag{8b}$$

$$\gamma_{ab}^r(t) \geq \chi_a^{r\downarrow}(t) + \chi_b^{r\uparrow}(t) - 1 \qquad \forall r \in \mathcal{R}_O^i, \forall (a,b) \in A^2, \forall t \in [(i-1)D, iD] \tag{8c}$$

A single node can serve as the junction for many turns. For a given turn (a,b), the set of all other turns sharing the same junction node is denoted by C_{ab}. The decision variable $\phi_{ab}(t) \in \{0,1\}$ resolves turn conflicts at every node by indicating which turn is active at a given time t, consequently deactivating all other turns which share that node as a junction. This is captured via the constraints:

$$\phi_{ab}(t) \geq \frac{1}{|\mathcal{R}_O^i|} \sum_{r \in \mathcal{R}_O^i} \gamma_{ab}^r(t) \qquad \forall (a,b) \in A^2, \forall t \in [(i-1)D, iD] \tag{9a}$$

$$\phi_{ab}(t) + \sum_{(a',b') \in C_{ab}} \phi_{a'b'}(t) \leq 1 \qquad \forall (a,b) \in A^2, \forall t \in [(i-1)D, iD] \tag{9b}$$

For any accepted request, the corresponding drone must reach the destination node within the arrival time window. This can only happen if the request is accepted in the first place, and the drone responsible for this request leaves the origin node after the earliest time of departure. Additionally, the drone is required to return to the origin node within a specified time window

$[l_r + STT, u_r + LTT]$. Here STT and LTT are, respectively, the shortest and longest travel times (between request origins and destinations) as estimated from available historical data. Such request tracking is achieved via three constraints. Let the decision variable $z_r \in \{0,1\}$ indicate the acceptance or rejection of request r. The first constraint only allows drone departure from the origin, for a request r, if $z_r = 1$. The second constraint necessitates drone arrival at the destination node if departure from origin had taken place. The third constraint imposes return to the origin if the drone had departed the origin:

$$\sum_{a \in \Gamma^+_{o_r}} \sum_{t=e_r}^{T} \chi_a^{r\uparrow}(t) = z_r \qquad \forall r \in \mathcal{R}^i_O \tag{10a}$$

$$\sum_{a \in \Gamma^+_{o_r}} \sum_{t=e_r}^{T} \chi_a^{r\uparrow}(t) = \sum_{a \in \Gamma^-_{d_r}} \sum_{t=l_r}^{u_r} \chi_a^{r\downarrow}(t) \qquad \forall r \in \mathcal{R}^i_O \tag{10b}$$

$$\sum_{a \in \Gamma^+_{o_r}} \sum_{t=e_r}^{T} \chi_a^{r\uparrow}(t) = \sum_{a \in \Gamma^-_{o_r}} \sum_{t=l_r+STT}^{u_r+LTT} \chi_a^{r\downarrow}(t) \qquad \forall r \in \mathcal{R}^i_O \tag{10c}$$

To facilitate the execution of the rolling horizon, some more constraints pertaining to idle and active requests must be added to the model formulation. Idle requests may potentially be carried over from previous intervals. Their routes may be subject to modification, but requests once accepted must be fulfilled. Thus any change in their z_r values is not permissible. The same is true for active requests from preceding intervals as well. This implies that in each interval, the following constraints must be imposed:

$$z_r = 1 \qquad \forall r \in \mathcal{R}_I \cup \mathcal{R}_A \tag{11a}$$

Since idle requests are still subject to re-routing, their earliest time of departure e_r cannot precede the start of the current interval i. An idle request cannot be re-optimized such that its new route begins at a time-step that has already passed:

$$e_r = \begin{cases} (i-1)D, & \text{if } e_r \le (i-1)D \\ e_r, & \text{otherwise} \end{cases} \qquad \forall r \in \mathcal{R}_I \tag{11b}$$

Finally, the unfinished portion of active request routes, to be undertaken in the current interval, must be preserved. Suppose we are in interval i and the last updated route vector for (currently) active request r is $\vec{\theta}_r = (r, t_{o_r}, [o_r, \dots, d_r, \dots, o_r])$. Let the number of nodes in its path μ_r be P and let $\omega(m,n)$ denote the directed link from node m to node n. Then the definitive path for request r is $\mu_r = [o_r, n_2, \dots, n_k, \dots, n_{P-1}, o_r]$ where $n_1 = n_P = o_r$. The following impositions must be made for all such active requests $r \in \mathcal{R}_A$:

$$\left.\begin{array}{ll} \chi^{r\downarrow}_{\omega(n_{k-1},n_k)}(t^r_{n_k}) = \chi^{r\uparrow}_{\omega(n_{k-1},n_k)}(t^r_{n_k}) = 1, & \text{if } k \ne 1 \\ \gamma_{\omega(n_{k-1},n_k)\omega(n_k,n_{k+1})}(t^r_{n_k}) = 1, & \text{if } n_k \ne o_r \end{array}\right\} \forall\, n_k \in \mu_r \mid (i-1)D \le t^r_{n_k} \le iD \tag{11c}$$

Here $t^r_{n_k}$ denotes the time at which the UAV responsible for request r reaches node n_k. $t^r_{n_k}$ can be determined since t_{o_r}, link lengths and UAV velocity are known. These constraints effectively preserve the link traversals and turns taken by the concerned active requests in the current interval. Since the origin is at the beginning and at the end of a route, there are no turns with the origin as a junction.

The decision variables are collectively mentioned below, along with their respective domains of definition:

$$z_r \in \{0,1\} \qquad \forall r \in \mathcal{R}^i_O \tag{12a}$$

$$\chi_a^{r\uparrow}(t), \chi_a^{r\downarrow}(t) \in \{0,1\} \qquad \forall r \in \mathcal{R}^i_O, \forall a \in A, \forall t \in [(i-1)D, iD] \tag{12b}$$

$$\gamma^r_{ab}(t) \in \{0,1\} \qquad \forall r \in \mathcal{R}^i_O, \forall (a,b) \in A^2, \forall t \in [(i-1)D, iD] \tag{12c}$$

$$\phi_{ab}(t) \in \{0,1\} \qquad \forall (a,b) \in A^2, \forall t \in [(i-1)D, iD] \tag{12d}$$

Let $\mathcal{P}_i$ be the integer polytope representing the constraint set corresponding to interval i, formally represented as:

$$\mathcal{P}_i \equiv \{\boldsymbol{z}, \boldsymbol{\chi}^{\downarrow}, \boldsymbol{\chi}^{\uparrow}, \boldsymbol{\gamma}, \boldsymbol{\phi} \text{ s.t. } (5) - (12)\} \tag{13}$$

The ILP formulation of the DDSP with online demand is:

$$\max \sum_{i \in I} \sum_{r \in \mathcal{R}^i_O} z_r p_r \tag{14a}$$

$$\text{s.t. } \boldsymbol{z}, \boldsymbol{\chi}^{\downarrow}, \boldsymbol{\chi}^{\uparrow}, \boldsymbol{\gamma}, \boldsymbol{\phi} \in \mathcal{P}_i \qquad \forall i \in I \tag{14b}$$

The above ILP formulation (14) captures the ultimate objective of the DDSP with online demand, which is to maximize the cumulative profit acquired by serving requests over time. Recall that this ILP formulation cannot be solved at once since demand data is assumed to be revealed in an online fashion. If all demand data was known *a priori*, the ILP formulation (14) can be used to represent the offline variant of the problem.

A myopic approach to solve the DDSP with online demand consists of restricting the data of the ILP formulation (14) to a single decision epoch i, this yields the following Myopic ILP formulation which will serve as a benchmark:

$$\max \sum_{r \in \mathcal{R}^i_O} z_r p_r \tag{15a}$$

$$\text{s.t. } z, \chi^{\downarrow}, \chi^{\uparrow}, \gamma, \phi \in \mathcal{P}_i \tag{15b}$$

Formulation (15) can readily be solved in an online fashion over a rolling horizon of time intervals $i \in I$, where decisions must be taken at each interval. Since this model does not consider the impact of current decisions on the fate of future requests, and only focuses on the current interval, it is hereafter referred to as the Myopic ILP and its associated policy the Myopic ILP Policy. The Myopic ILP formulation is summarized in its entirety (with the individual constraints of $\mathcal{P}_i$) in Appendix. The pseudo-code of the implementation of the Myopic ILP policy via a rolling horizon is summarized in Algorithm 1.

Algorithm 1: Pseudo-code for Myopic ILP Policy

for $d \leftarrow 1$ **to** N_{days} **do**
  initialize request set $\mathcal{R}_{\checkmark}$ and collective route vector $\vec{\Theta}$
  for $i \leftarrow 1$ **to** I **do**
    initialize request sets $\mathcal{R}_A, \mathcal{R}_I, \mathcal{R}_C^i$
    // request segregation
    for $r \in \mathcal{R}_{\checkmark}$ **do**
      if $t_{o_r} \leq (i-1)D$ **then**
        $\mathcal{R}_A \leftarrow \mathcal{R}_A \cup \{r\}$
      else
        if $e_r \leq (i-1)D$ **then**
          $e_r \leftarrow (i-1)D$
        $\mathcal{R}_I \leftarrow \mathcal{R}_I \cup \{r\}$
    populate $\mathcal{R}_C^i$
    // model formulation
    construct integer polytope $\mathcal{P}_i$ (13)
    for $r \in \mathcal{R}_I \cup \mathcal{R}_A$ **do**
      add constraint $z_r = 1$
    for $r \in \mathcal{R}_A$ **do**
      add constraints

$$\left.\begin{aligned} &\chi^{r\downarrow}_{\omega(n_{k-1},n_k)}(t^r_{n_k}) = \chi^{r\uparrow}_{\omega(n_{k-1},n_k)}(t^r_{n_k}) = 1, \quad \text{if } k \neq 1 \\ &\gamma_{\omega(n_{k-1},n_k)\omega(n_k,n_{k+1})}(t^r_{n_k}) = 1, \quad \text{if } n_k \neq o_r \end{aligned}\right\} \forall\ n_k \in \mu_r \mid (i-1)D \leq t^r_{n_k} \leq iD$$

    solve ILP Formulation (15)
    // update/add drone routes
    for $r \in \mathcal{R}_O^i$ **do**
      if $r \in \mathcal{R}_I$ **then**
        update $\vec{\theta}_r \leftarrow (r, t_{o_r}, \mu_r)$
      if $r \in \mathcal{R}_C^i$ **then**
        if $z_r^{sol} = 1$ **then**
          $\mathcal{R}_{\checkmark} \leftarrow \mathcal{R}_{\checkmark} \cup \{r\}$
          $\vec{\theta}_r \leftarrow (r, t_{o_r}, \mu_r)$
          $\vec{\Theta} \leftarrow \vec{\Theta} \cup \{\theta_r\}$
    // remove completed requests
    for $r \in \mathcal{R}_{\checkmark}$ **do**
      if $t_{o_r} \leq iD$ **then**
        $\mathcal{R}_{\checkmark} \leftarrow \mathcal{R}_{\checkmark} \backslash \{r\}$
        $\vec{\Theta} \leftarrow \vec{\Theta} \backslash \{\theta_r\}$

4. Solution methodology

We present the data-driven optimization approach to solve the DDSP with online demand. Section 4.1 illustrates the motivation and construction of a surrogate objective function for the ILP. The surrogate objective thus created requires predictive prescriptions from an ML component. Section 4.2 details the training scheme of this ML component, which is summarized in Algorithm 3. Once trained, the ML component supplies the predictive prescriptions necessary for implementing the surrogate policy, and this leads to the novel data-driven optimization approach proposed in Section 4.3. The surrogate policy is summarized in Algorithm 4.

4.1. Surrogate objective function

Maximizing profit acquisition at each time interval ignores future demand. Taking future demand into account requires making decisions that are sub-optimal in the current time interval but increase the cumulative reward by contributing to the objective of a later interval. We tolerate a sub-optimal reward per interval $\sum_{r\in\mathcal{R}^i_C} R(s_r, a_r)$ to achieve a greater reward-to-go $\sum_{r\in\mathcal{R}} R(s_r, a_r)$. This sub-optimality comes from allocating some resources to the future, thus preventing their use in the present. In other words, a sacrifice is made in the present to reap greater rewards in the future.

We define available link capacity per unit time as the network resource. This is captured as $C_l - \sum_{r\in\mathcal{R}^i_O} \chi_l^{r\uparrow}(t)$, where the capacity of link occupied by idle and current requests is deducted from its total capacity. The strategy then translates into (intentionally) creating reserve capacity in the network at the current interval, which prevents that capacity from being utilized at the current interval. This may lead to rejection of requests or longer routes in the current interval, but is expected to yield greater profit accumulation in future intervals. We motivate the choice of using link capacity as a resource because: first, it is a quantitative definition, and link capacity at a given time can be easily measured during the execution of the optimization program; second, the spatial distribution of origins and destinations of customer requests, combined with conflict resolution of drone routes, leads to a heterogeneous resource space. In other words, the available capacities on different links have different value to the network. This creates a rich decision set, providing a sizeable margin with which to outperform myopic policies.

The priority of available capacity on link $l \in A$ at time interval i is denoted $\beta_{i,l}$. This indicates the significance of the link in the network. If a link has higher priority, then any reserve capacity on it is relatively more important to network operation and route creation, compared to other links. The formulation of a data-driven prescription of $\beta_{i,l}$ is described in Section 4.2.1. Reserve capacity is generated on network links in proportion to their priority, since slack created on higher priority links will have greater impact on maximizing profit in future intervals.

Indefinite and uncontrolled slack creation will adversely affect network performance and reduce cumulative profit. The reserve capacity being created in the present should at some point be capitalized upon. Hence the balance between profit maximization and ($\beta_{i,l}$ proportionate) slack creation must be controlled. We construct a new objective function to incorporate the sacrificial strategy. Since this objective function is intended to replace the objective function of the Myopic ILP, it is called the surrogate objective function and comprises two components:

$$\max \sum_{r\in\mathcal{R}^i_O} z_r p_r \;+\; \alpha_i \sum_{l\in A} \beta_{i,l} \sum_{t=(i-1)D}^{iD} \left[C_l - \sum_{r\in\mathcal{R}^i_O} \chi_l^{r\uparrow}(t) \right] \tag{16}$$

The DDSP formulation with the surrogate objective function is named the Surrogate ILP. This formulation, combining the surrogate objective function with the constraints defining the integer polytope $\mathcal{P}_i$, is described in Appendix. The first component is profit maximization, and the second is slack creation based on link priority. For every link $l \in A$, we first compute the slack capacity on l during the current time interval, and then multiply it with the relative priority of that link, $\beta_{i,l}$. We also define a parameter α_i which influences the balance between the two components. The value of α_i chosen for this interval determines to what extent reserve capacity will be created on network links (relative to profit maximization). Note that independent of the value of α_i, the slack generated on a link will be proportional to the relative priority of that link. Unless, of course, if $\alpha_i = 0$, in which we case the surrogate objective is reduced to the myopic objective of pure profit maximization and $\beta_{i,l}$ has no role left to play.

In the computational experiments, both components of the surrogate objective function are standardized, so that α_i can effectively control the balance. The first component – which is essentially the myopic objective function – is standardized by division with the expected total profit over the entire time horizon $p_{avg}\mathbf{E}[|\mathcal{R}|]$. This component is hereafter referred to as the Profit Component. The second component is standardized by division with the total available network capacity in that interval $(\sum_{l\in A} C_l)D$. This component is hereafter referred to as the Slack Component.

4.2. Training procedure of relative link priority ($\beta_{i,l}$)

We next explain the training procedure used to determine predictive prescriptions of the relative link priority ($\beta_{i,l}$) which is used in the Surrogate ILP. The training procedure is articulated across two main steps: i) the synthesis of training data via lookahead simulation (Section 4.2.1) and ii) the training of a supervised ML model for predicting relative link priority (Section 4.2.2). To scale up the training process, we use a heuristic algorithm to solve the Myopic ILPs that arise therein (Section 4.2.3).

4.2.1. Training data synthesis via lookahead simulation

We use lookahead simulations to synthesize training data that will be used to train a predictive ML model which itself will be queried during the execution of the Surrogate ILP policy. This procedure is compiled in Algorithm 3.

The priority of a link can be correlated to the frequency of link traversal by drones in a suitable span of time. Given a duration of network operation (e.g. δ intervals), crucial high priority links will experience large throughput and are likely to create bottlenecks which impede request acceptance. The link priority $\beta_{i,l}$ is therefore formally defined as the number of times link l is traversed by incoming requests between the beginning of interval i and the end of interval $i+\delta$.

$$\beta_{i,l} = \sum_{q=i}^{i+\delta} \sum_{r\in\mathcal{R}^q_C} \sum_{t=(q-1)D}^{qD} \chi_l^{r\uparrow,sol}(t) \tag{17}$$

Here $\chi_l^{r\uparrow,sol}(t)$ represents the routing solution value of the upstream-link variable $\chi_l^{r\uparrow}(t)$ for link l and time t at the end of each interval. We also define a vector $\vec{\boldsymbol{B}}$ to store link priority values for all intervals.

$$\vec{\boldsymbol{B}} = \left[\vec{\boldsymbol{\beta}}_i\right]^{i\in\{1,\ldots,I\}} = \left[\beta_{i,l}\right]_{l\in A}^{i\in\{1,\ldots,I\}} \tag{18}$$

Link priority values as defined above are a function of drone traffic evolution in the network, which in turn depends on customer demand. From a snapshot of the network state at a certain time-step, the short-term future of the network may be estimated, provided historical patterns of customer demand are known. The introduction of the ML framework serves this purpose. We develop a supervised ML model which takes as input a snapshot of the network (number of drones occupying each link) and returns as output an estimate of link priorities for the near future. This input snapshot of link occupation is captured for the beginning of every time interval and is denoted by:

$$s_{i,l} = \sum_{r\in\mathcal{R}_A} \sum_{t=(i-1)D-\Delta}^{(i-1)D+\Delta} \chi_l^{r\uparrow,sol}(t)\ \mathbb{1}[t \le (i-1)D < t+\Delta] \tag{19}$$

Here $\Delta = \frac{\rho_l}{v}$ is the time taken by a drone to traverse link l. Thus $s_{i,l}$ denotes the number of UAVs occupying link l in interval i at interval start. Eq. (19) increments the value of $s_{i,l}$ for every active drone occupying link l at the beginning of interval i (i.e., $t = (i-1)D$). We define a collection vector to store link occupancy values for all intervals.

$$\vec{\boldsymbol{S}} = \left[\vec{\boldsymbol{s}}_i\right]^{i\in\{1,\ldots,I\}} = \left[s_{i,l}\right]_{l\in A}^{i\in\{1,\ldots,I\}} \tag{20}$$

$\vec{\boldsymbol{S}}$ serves as the input feature vector dataset and $\vec{\boldsymbol{B}}$ as the target output vector dataset while training the ML model. Once trained, it is deployed in the Surrogate ILP to estimate the values of $\beta_{i,l}$ which are subsequently used in the surrogate objective function (16). Note that in the training phase, all $\beta_{i,l}$ – and hence $\vec{\boldsymbol{B}}$ – are calculated via Eq. (17) to produce the output targets of the training dataset. In the testing and execution of the Surrogate ILP, however, the values of $\beta_{i,l}$ are predicted by the ML model.

In the training scheme, the number of training intervals is designed to be much larger than that in a test instance ($I_{training} \gg I$). This is equivalent to processing several test instances successively without emptying the network between two consecutive test instances. At the beginning of each training interval i, the link occupancy on each link l by currently active requests is calculated and stored in $s_{i,l}$ (lines 8–10 of Algorithm 3). For this we define the function LinkActivity which computes $s_{i,l}$ as per Eq. (19). This function takes two inputs — the route μ_r of an active request and the link occupancy vector $\vec{\boldsymbol{s}}_{\boldsymbol{i}}$ for the current interval. It iterates through the route to determine which link the UAV occupied at the beginning of the interval ($t = (i-1)D$) and increments the corresponding $s_{i,l}$ in $\vec{\boldsymbol{s}}_{\boldsymbol{i}}$. Next, a lookahead is performed by virtually simulating the future of the network for the next $I_{virtual}$ number of intervals (lines 12–13 of Algorithm 3). For these intervals, virtual requests $\mathcal{R}_V$ are created, imitating historical data. These virtual requests are then routed through the network, obeying link capacity and conflict resolution constraints in the presence of active requests. The routes of successfully serviced virtual requests then contribute to the values of $\beta_{i,l}$ for interval i (lines 14–16 of Algorithm 3). For a link l, $\beta_{i,l}$ is equal to the number of times link l was traversed by virtual requests in the lookahead simulation. This captures the perceived priority of link l in the foreseeable near future, from the perspective of the network at interval i, in accordance with Eq. (17), with $\delta = I_{virtual}$. We define the function BetaUpdate, which takes two inputs: the route μ_r of a virtual request and the link priority vector $\vec{\boldsymbol{\beta}}_{\boldsymbol{i}}$ for the current interval, and updates $\beta_{i,l}$ for all the links that this request traverses according to μ_r.

Once the virtual lookahead is complete, all virtual request paths are erased from the network. Thereafter, the process resembles the Myopic ILP. Idle requests from previous intervals and incoming requests of current interval i are routed. Successfully accepted requests are stored in $\mathcal{R}_{\checkmark}$. Successful paths μ_r for a given request r are also stored in a route vector $\theta_r = (r, t_{o_r}, \mu_r)$ and added to the collective route vector $\vec{\Theta}$. Then the next interval $i+1$ is initiated.

4.2.2. *k*-Nearest Neighbors multi-output regressor

We use k-Nearest Neighbors (kNN) regression to train a predictive model for estimating relative link priority based on network snapshots. The kNN regressor is a supervised ML algorithm used for prediction tasks. It predicts the output target value of a new data point by considering the average or weighted average of its k nearest neighbors in the input feature space. The kNN algorithm utilizes a defined distance metric, such as the Euclidean distance, to measure the similarity between data points in the input feature space. By identifying the k training feature vectors closest to a new data point, the algorithm establishes the nearest neighbors of the given data point. The prediction of the target value of the new data point is made by averaging or weighting the target values associated with its k nearest neighbors, leveraging the assumption that closer points in the feature space exhibit similar target values. The kNN multi-output regressor extends the concept of the kNN regressor to handle multiple target values for each data point. Notably, the kNN regressor is non-parametric and makes minimal assumptions about the underlying data distribution, making it more versatile in handling diverse datasets. It can also capture non-linear relationships between input features and output targets. We use the training data synthesized using the lookahead simulation procedure described in Section 4.2.1 to train the kNN regressor.

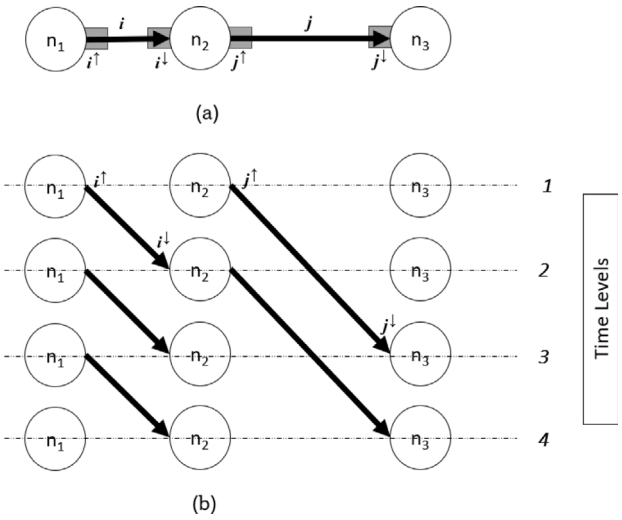


Fig. 3. (a) Original Network (b) Time-expanded Network.

4.2.3. Reservation heuristic

In practice, the training phase has an associated time budget, implying that link priority values must be estimated expediently. Solving the Myopic ILP for $I_{training}$ intervals demands excessive time and memory. Levin and Rey (2023) propose a reservation heuristic which routes incoming customer requests quickly. This heuristic is used within the training phase. We illustrate the reservation heuristic in Algorithm 2 and formalize the training scheme in Algorithm 3.

The original spatial network $G = (N, A)$ can be converted into a spatio-temporal network G^T as demonstrated in Fig. 3. G^T is the time-expanded dual graph of G. To construct G^T, the nodes and links are extended across time through discrete time levels and connections are made on the basis of link travel times. Suppose a link in G connects node n_1 to node n_2 and can be traversed in Δt time steps. Then it connects two nodes Δt time levels apart in G^T. The graph G^T can be viewed as a network of node-time tuples (n, t), where n denotes the node and t is the time level on which it is located. The reservation heuristic works by routing customer requests on G^T using Dijkstra's shortest-path finding algorithm, which finds the shortest path between two specific nodes in a graph with non-negative edge weights. Starting from the source node, this algorithm assigns tentative distances to all other nodes, initially setting the distance to the source as zero and others as infinity. The algorithm then iteratively selects the unvisited node with the smallest tentative distance, updates the distances of its neighboring nodes, and continues this process until the target (destination) node is reached or all nodes have been visited. The final shortest path from the source to the destination is determined once the algorithm terminates. Since G^T is the time-expanded dual graph, nodes are replaced with node-time tuples (n, t). The discovered route is then "reserved" for that request by securing link capacity in space–time, making that capacity unavailable for other requests. Since turn conflict constraints must also be respected in G^T, the heuristic removes connectivity between node-time tuples whenever constraints (5) and (8)–(9) are violated. Algorithm 2 defines the function NewReservation which discovers a path μ_r^T for request r, makes the relevant reservations in G^T and deduces the corresponding path μ_r in G. Two additional functions are also required in the training phase. A request may already have a path, but reservations for this path may either need to be made or erased. We use functions Reserve(μ_r^T, G^T) and EmptyReservation(μ_r^T, G^T) for these purposes respectively. They follow a similar procedure to NewReservation and are hence not illustrated.

4.3. Surrogate ILP policy

The Surrogate ILP policy is executed after completing the training procedure described in Section 4.2. The pseudo-code the Surrogate ILP policy is presented in Algorithm 4. The Surrogate ILP policy iteratively solves the Surrogate ILP where the surrogate objective function is parameterized by the predictive prescriptions for relative link priority $(\beta_{i,l})$ and the balance parameter (α_i). Recall that the ratio of profit maximization to link priority-proportional slack creation is controlled via the balance parameter α_i. The magnitude of this balance parameter may be varied from one time interval to the next. Since α_i is coupled with the slack component, a greater magnitude favors slack creation and a smaller one favors profit maximization. Too large a value of α_i, sustained over many intervals, can impose a vacuum in the network which degrades cumulative profit acquired by injecting excessive reserve capacities. Too small a value can make its contribution insignificant, and a negative value is likely to suffocate the network by overloading link capacities. It follows that there exists a range of desirable values for α_i, which we explore in our numerical experiments. The

Algorithm 2: Pseudo-code for reservation heuristic

Input: r, G_T
Output: μ_r
Function `NewReservation`(r, G_T)**:**
  find space-time path μ_r^T in G_T satisfying
  1: previous reservations for (5) and (8)–(9)
  2: spatial and temporal constraints of request r
  if μ_r^T FOUND **then**
    make reservations (break connectivity) in G_T as per (5) and (8)–(9) for μ_r^T
    create μ_r in G from μ_r^T
    return μ_r
  else
    return FALSE
End Function

Algorithm 3: Training data synthesis for predicting relative link priority

initialize request sets $\mathcal{R}_{\checkmark}$ and collective route vector $\vec{\Theta}$
set $s_{i,l} \leftarrow 0$ and $\beta_{i,l} \leftarrow 0 \; \forall \; l \in A, \forall \; i \in \{1, \ldots, I_{training}\}$
create G^T from G
for $i \leftarrow 1$ **to** $I_{training}$ **do**
  initialize request sets $\mathcal{R}_A, \mathcal{R}_I, \mathcal{R}_C^i$
  `// request segregation as in Algorithm 1 (lines 6 to 12)`
  `// reserve active requests and record link traversals`
  for $r \in \mathcal{R}_A$ **do**
    RESERVE(μ_r^T, G_T)
    LINKACTIVITY$(\mu_r, \vec{s}_i)$
  `// virtual requests`
  for $i_v \leftarrow 1$ **to** $I_{virtual}$ **do**
    create virtual requests for interval i_v and store in $\mathcal{R}_V$
  for $r \in \mathcal{R}_V$ **do**
    if NEWRESERVATION(r, G_T) **then**
      BETAUPDATE$(\mu_r, \vec{\beta}_i)$
  for $r \in \mathcal{R}_V$ **do**
    EMPTYRESERVATION(μ_r^T, G_T)
  `// solve real requests of current interval and update/add drone routes`
  populate $\mathcal{R}_C^i$
  for $r \in \mathcal{R}_O^i$ **do**
    $\mu_r \leftarrow$ NEWRESERVATION(r, G_T)
    if $r \in \mathcal{R}_I$ **then**
      update $\theta_r \leftarrow (r, t_{o_r}, \mu_r)$
    if $r \in \mathcal{R}_C^i$ **then**
      if $\exists \mu_r$ **then**
        $\mathcal{R}_{\checkmark} \leftarrow \mathcal{R}_{\checkmark} \cup \{r\}$
        $\theta_r \leftarrow (r, t_{o_r}, \mu_r)$
        $\vec{\Theta} \leftarrow \vec{\Theta} \cup \{\theta_r\}$
  `// empty all reservations to prepare for next interval`
  for $r \in \mathcal{R}_O^i \cup \mathcal{R}_A$ **do**
    EMPTYRESERVATION(μ_r^T, G_T)
  `// remove completed requests as in Algorithm 1 (lines 33 to 36)`
return $\vec{B}$ and $\vec{S}$

variation (or lack thereof) in the values of α_i, denoted by the sequence $\left[\alpha_i\right]^{i \in \{1,\ldots,I\}}$, is called the *α-profile*. We investigate three types of α-profiles – constant, step-wise and continuous.

Algorithm 4: Surrogate ILP Policy

1 **for** $d \leftarrow 1$ **to** N_{days} **do**
2 initialize request set $\mathcal{R}_{\checkmark}$ and collective route vector $\vec{\Theta}$
3 initialize α-profile $\left[\alpha_i\right]^{i\in\{1,\dots,I\}}$
4 set $s_{i,l} \leftarrow 0$ and $\beta_{i,l} \leftarrow 0 \; \forall \; l \in A, \forall \; i \in \{1,\dots,I\}$
5 **for** $i \leftarrow 1$ **to** I **do**
6 initialize request sets $\mathcal{R}_A, \mathcal{R}_I, \mathcal{R}_C^i$
7 // request segregation as in Algorithm 1 (lines 6 to 12)
8 **for** $r \in \mathcal{R}_A$ **do**
9 LinkActivity$(\mu_r, \vec{s}_i)$
10 // query ML component for link priorities & standardize
11 $\vec{\beta}_i \leftarrow k\text{NN}(\vec{s}_i)$
12 $\beta^{\max} \leftarrow \max(\vec{\beta}_i)$
13 $\vec{\beta}_i \leftarrow \left[\frac{\beta_{i,l}}{\beta^{\max}}\right]_{l\in A}$
14 populate $\mathcal{R}_C^i$
15 // model formulation as in Algorithm 1 (lines 15 to 20)
16 solve Surrogate ILP Formulation: optimize (16) s.t. integer polytope (13)
17 // update/add drone routes as in Algorithm 1 (lines 24 to 31)
18 // remove completed requests as in Algorithm 1 (lines 33 to 36)

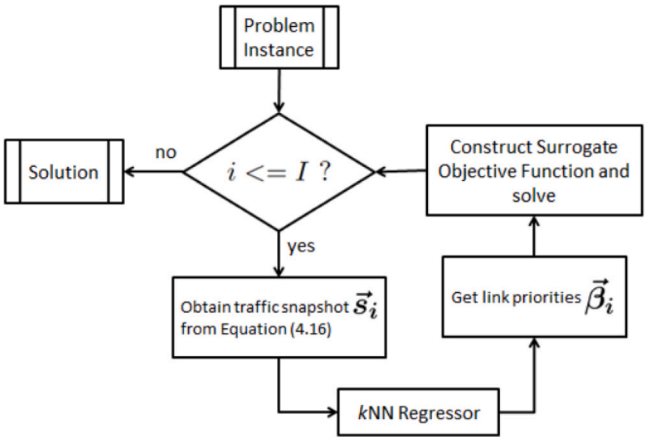


Fig. 4. Flowchart of the Solution Approach.

At the start of every interval, the current active requests (which were accepted in previous intervals) are segregated. The function LinkActivity populates the link occupancy vector $\vec{s}_i$ in accordance with the positions of active requests (lines 8–9 of Algorithm 4). The trained kNN regressor is then queried with $\vec{s}_i$ as input, after which it supplies a predictive prescription of link priorities in the form of $\vec{\beta}_i$ (line 11 of Algorithm 4). Once the link priority vector is standardized, it is then implemented in the parameterized surrogate objective function, along with the chosen α-profile (line 16 of Algorithm 4). This solution approach is illustrated in Fig. 4.

5. Numerical experiments

We conduct numerical experiments to assess the performance of the proposed data-driven optimization approach, i.e. the Surrogate ILP, for the online DDSP. We use the Myopic ILP policy as a baseline for benchmarking and we also conduct sensitivity analyses on learning parameters of the ML-enabled surrogate objective function.

We conduct these experiments on the Sioux Falls network, since it is an established data set for traffic routing problems and has been used in by Levin and Rey (2023) for the deterministic DDSP. There are 24 nodes and 76 links in this network. The length of the time horizon is $T = 60$ min, with 12 intervals of 5 min each ($I = 12$ and $D = 5$ min). For routing and capacity tracking, the time horizon is discretized into time steps of a single minute. After 60 min elapse, the network still remains operational as long as the final accepted request reaches its destination. Thus the network remains active until $\max_{r\in R_{\checkmark}} d_r$. We focus on the proof-of-concept for the proposed novel solution methodology. In line with this goal, we maintain all link capacities at 1 drone per minute, even

Table 1
Summary of α-profiles for the Surrogate ILP Policy.

Name	α-profile	Type
SP_CTE1	$\alpha = 2$	Constant
SP_CTE2	$\alpha = 1.5$	
SP_CTE3	$\alpha = 1$	
SP_CTE4	$\alpha = 0.5$	
SP_CTE5	$\alpha = -1$	
SP_STP1	$\alpha = 1 \rightarrow 0.5 \mid 6{:}6$	Step-wise
SP_STP2	$\alpha = 1 \rightarrow 0.75 \rightarrow 0.5 \rightarrow 0 \mid 3{:}3{:}3{:}3$	

though this is probably more restrictive than is necessary to avoid collisions. The size of the ILPs are a function of the number of variables, and assigning small capacities permits solving the problem on a large network with capacity constraints. The number of customer requests arriving in each interval follows a Poisson process with rate $\lambda = 100$, which amounts to an expected total of 1200 requests over the course of 12 intervals. Within an interval i, the request inter-arrival times are exponentially distributed between the start and end of the interval $[(i-1)D, iD)$.

The spatial distribution of request origins o_r and destinations d_r is exponential with respect to the y-coordinate, with origins dominating the bottom of the network and destinations occupying the top. Let the y-coordinate of a node n be denoted by y_n, and let the maximum y-coordinate in the unidirectional network be y_{max}. The probability that this node can become an origin is given by $10 \cdot \exp\left(-\frac{\sqrt{(y_n^2)\cdot 2}}{25}\right)$. This formula uses an exponential decay function, where the probability decreases as the distance of the node's y-coordinate from the origin (0, 0) increases. The scaling factor of 10 amplifies the probability, and the denominator of 25 in the exponential term controls the rate of decay. The probability that this node can become a destination is given by $10 \cdot \exp\left(-\frac{\sqrt{((y_{max}-y_n)^2)\cdot 2}}{25}\right)$. Here, the distance is measured from y_{max}, which represents the farthest point in the network along the y-axis. The function similarly employs an exponential decay to reduce the probability as the node's y-coordinate approaches the origin point of the destination probabilities. Both the origin and destination probabilities are later normalized to ensure they form valid probabilities. This expression of spatial distributions captures scenarios where spatial clustering of demand is expected, such as in urban delivery environments, where certain areas are often hot-spots for request generation.

For each request, the earliest permissible departure time e_r varies uniformly between $[0, 10]$ minutes of request submission time t_r. The arrival time window at the destination $[l_r, u_r]$ follows a skewed normal distribution between expected travel time and the end of the time horizon, skewed towards expected travel time. This accounts for real-world patterns in demand. This approach introduces asymmetry, ensuring that request arrival times are biased towards the current time rather than being evenly spread throughout the entire time horizon. The skewness parameter controls this asymmetry, favoring earlier arrivals closer to the current time while tapering off as the time horizon progresses, modeling the desire of customers to receive deliveries as early as possible, but with a reasonable tolerance for transport times. The shortest and longest travel times – STT and LTT respectively – are determined to be 4 and 20 minutes respectively. These values are used to construct the time window for return of the drone to the origin, specified as $[l_r + STT, u_r + LTT]$. The price (profit) of each request is uniformly distributed between $[1, 10]$.

With these spatial, temporal and price distributions for customer requests, we generated 5 test instances. We implemented the proposed algorithms in Python on a Windows machine with 3.19 GHz and 64 GB of RAM. We used CPLEX 22.1.0.0 to solve the test instances, with a solve time limit of 300 s for each interval of a given instance.

5.1. α-profiles

We consider two types of α-profiles: constant and step-wise. The performance of constant profiles can be used to determine the desirable range of α_i, after which temporal variations can be introduced to extend the analysis to step-wise profiles. It should be noted that for step-wise profiles, α_i is only evaluated at discrete points in time, marking the beginning of each interval. A summary of the α-profiles experimented with is provided in Table 1.

An intuitive approach consists of starting with a high value of α_i and reduce it over time. This realizes the strategy of preferentially creating link priority-based reserve capacity in the initial intervals and transitioning into profit accumulation towards the end of the time horizon. All step-wise profiles presented are non-increasing for this reason. However, constant α-profiles have significant potential too, since the kNN model is adaptive and returns a different link priority vector at each interval. This implies that the priority-based slack creation is also adaptive and may relieve some of the adjustment burden from α_i.

5.2. Myopic ILP policy

We first examine the performance of the Myopic ILP policy on 5 problem instances based on the Sioux Falls network as described above. Table 2 encapsulates the results. The Myopic ILP achieved an average profit of 1004 across all 5 instances, with the expected maximum possible profit being $p_{avg}\mathbf{E}[|\mathcal{R}|] = 6600$. The average service rate was 12.8%. These performance results serve as a baseline for bench-marking the proposed data-driven optimization approach. The average solve-time for an interval across all 5 instances

Table 2
Performance of the Myopic ILP Policy.

Instance	Total requests	Service rate	Profit
0	1224	12.8%	968
1	1230	10.7%	892
2	1208	12.9%	1045
3	1182	13.7%	1069
4	1210	14.0%	1045

was 278.6 s. It was observed that if an optimal solution was reached, then the solve-time was below 150 s, but for some intervals the solver reached the time limit of 300 s and gave the best feasible solution it could find. The latter case occurred in the initial three to four intervals, presumably because the network traffic was still evolving and had not reached complete saturation. In such a situation, many idle requests have not yet been dispatched and incoming requests continue to arrive at each interval, making it difficult to reach optimality. As idle requests begin to enter the network in greater numbers and network traffic increases, the pending request load subsides because more requests are rejected and hence optimality can be reached more easily.

5.3. Training *k*NN multi-output regressor

The training instance contains 1000 intervals, instead of the usual 12 for test instances. The lookahead simulation contained 3 virtual intervals. The training scheme described in Algorithm 3 was executed on this instance. This supplied the training dataset where the input feature set is $\vec{\boldsymbol{S}}$ and the output target set is $\vec{\boldsymbol{B}}$. The reservation heuristic was employed to handle the size of the training instance. The time-expanded dual graph G^T was constructed up to a time layer $t_{\max} + \Delta$ where $t_{\max} = \max_{r \in \mathcal{R}} d_r$ of the training requests. The purpose of the buffer Δ is to account for the virtual requests being created in the final interval, some of whose arrival windows may exceed $t_{\max}$.

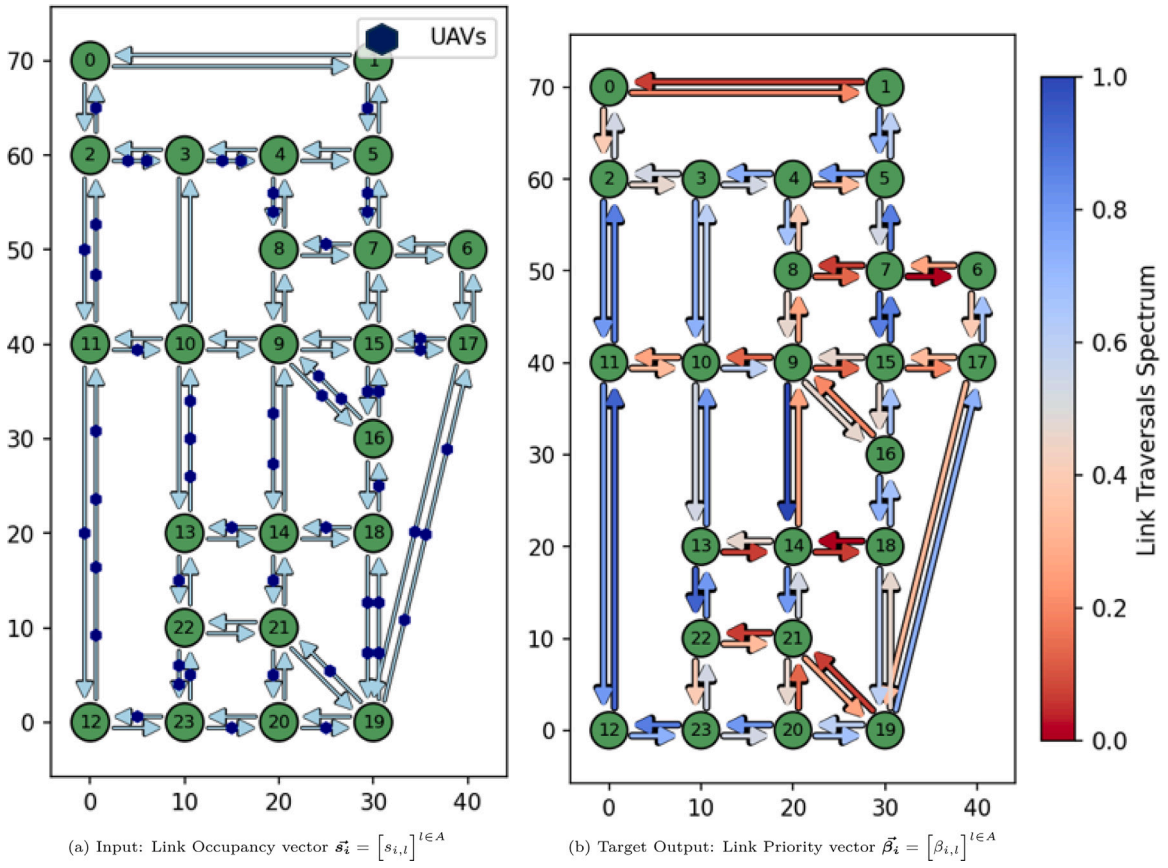
The completed training dataset was passed through a kNN multi-output regressor. kNN was implemented through the Scikit-learn package in Python. As mentioned previously, the collection vectors $\vec{\boldsymbol{S}}$ and $\vec{\boldsymbol{B}}$ contain the link occupancy vectors $\vec{\boldsymbol{s}}_{\boldsymbol{i}} = \left[s_{i,l}\right]^{l \in A}$ and link priority vectors $\vec{\boldsymbol{\beta}}_{\boldsymbol{i}} = \left[\beta_{i,l}\right]^{l \in A}$ respectively for all training intervals. During the execution of the training scheme, the link occupancy vector $\vec{\boldsymbol{s}}_{\boldsymbol{i}}$ was converted into a 2-D node adjacency matrix $\boldsymbol{N}_{\boldsymbol{s}}$ where the row and column indices of the matrix are the nodes of the network. Each entry $s_{m,n}$ of this matrix represents a directed link from node m to node n along link $\omega(m,n)$, and the value of the entry equals the link occupancy. This transformation is useful because it encodes the state of the network in a more information-rich format. Apart from storing link occupancy, the node adjacency matrix also conveys the network topology (for example, two separate non-zero entries in the same row m represent two outgoing links with m as the common source). With $\boldsymbol{N}_{\boldsymbol{s}}$ as input, the kNN regressor is trained to estimate the link priorities as output. The number of neighbors (which is the value of k) considered for each input was set to 60, and the distance metric was set to Euclidean distance.

Fig. 5 demonstrates one pair of input and target output vectors in the training scheme (training interval 500). The input is the snapshot of the network at the beginning of training interval 500, depicting the number of drones (UAVs) on each link (Fig. 5(a)). After simulating 3 virtual intervals, the link traversals of the accepted virtual requests give rise to the desired target output as shown in Fig. 5(b). This kNN model is trained on such input and target output pairs for the entire duration of the training scheme. Note that, although the spatial distribution of request origins and destinations influences the link priorities, so does the imposition of the return to origin. Even though origins dominate the bottom portion of the network and destinations dominate the top, links directed southwards (top to bottom) are of comparable priority. This is because drones have to return to their respective origins. Some links responsible for lateral movement are also of moderate to high importance. Once the kNN model learns these patterns, it helps the Surrogate ILP avoid bottlenecks on high priority links.

5.4. Constant α_i

A sensitivity analysis was conducted to quantify the impact of the α-profile on the performance of the Surrogate ILP policy. Since $\alpha_i = 0 \ \forall i \in \{1, \ldots, I\}$ is equivalent to the Myopic ILP, α_i was perturbed above and below 0 and the effect on cumulative profit acquired was observed. The cumulative profit of the Surrogate ILP is used to determine the Profit Gap % with respect to the Myopic ILP and serves as the main performance metric. Similarly, the cumulative service (total requests accepted) and Service Gap % are recorded. Note that the Service Gap % is calculated with respect to the total number of requests that arrived for an entire instance. The performance of the Surrogate ILP policy with constant α-profiles are displayed in Table 3.

As expected, a fully negative α-profile (SP_CTE5) performed worse than the Myopic ILP, because the surrogate objective seeks to minimize slack and force traffic though the network in a counter-productive manner. The influence of the link priorities $\beta_{i,l}$ further worsened the solution, because high priority links are overloaded more in proportion to their priority. In contrast, a positive alpha profile performed better than the benchmark, but the profit gap decreases and eventually becomes negative if α is exceedingly large (SP_CTE1). This can be explained by observing that too large an alpha maintained over the entire time horizon drains the network excessively and prevents profitable requests from being routed successfully. For the given network and the demand scenario presented, constant α profiles of 1.5, 1 and 0.5 performed the best, with an average gap across the 5 problem instances of 26%, 37.5% and 47.3% respectively.



(a) Input: Link Occupancy vector $\vec{s_i} = \left[s_{i,l}\right]^{l \in A}$ (b) Target Output: Link Priority vector $\vec{\beta_i} = \left[\beta_{i,l}\right]^{l \in A}$

Fig. 5. (a) Input Vector and (b) Target Output Vector for training interval 500 in the training scheme. *Note: drones occupying the same link in the network need not be equidistant but have been shown as such for simplicity*.

It is observed that the Surrogate ILP policy defined by α-profile SP_CTE2 typically served slightly fewer requests than the Myopic ILP policy, but always attained a larger profit. This is likely due to the fact that a high α of 1.5 prioritizes capacity reservation in a manner which serves fewer requests than the Myopic ILP, but the requests accepted by SP_CTE2 are more profitable. However, SP_CTE3 and SP_CTE4 consistently achieved slightly higher service rates and considerably higher profit gaps in comparison to the Myopic ILP. The Myopic ILP behaved greedily – optimal in the current interval (for initial intervals) but negligent of the future – and lost out on potentially profitable requests in later intervals. Instead, the Surrogate ILPs maintained reserve capacity on crucial links which enabled the acceptance of high profit requests no matter in which interval they arrived. It is possible however, to create excessive reserves that begin to hinder traffic through the network, as evidenced by the inferior performance of SP_CTE1.

5.5. Step-wise α_i

Having demonstrated the persistent superiority of SP_CTE2, SP_CTE3 and SP_CTE4, step-wise changes in the α-profile with 1 and 0.5 as pivots were examined. These Surrogate ILPs are called SP_STP1 and SP_STP2. For example, SP_STP1 has the α-profile $1 \rightarrow 0 \mid 6 : 6$, which depicts 6 intervals of $\alpha_i = 1$ followed by 6 intervals of $\alpha_i = 0$. Similarly, SP_STP2 has the α-profile $1 \rightarrow 0.75 \rightarrow 0.5 \rightarrow 0 \mid 3 : 3 : 3 : 3$, which depicts 3 intervals each of $\alpha_i = 1$, $\alpha_i = 0.75$, $\alpha_i = 0.5$ and $\alpha_i = 0$. In these two Surrogate ILP policies, the latter one captures a smoother transition to a lower α_i.

Table 4 records the performance of this class of α-profiles. A higher quality solution from the group of step-wise α-profiles is observed than from the constant profiles. However, there is no clear superiority of one step-wise profile over another. In the first two instances, SP_STP2 is superior to SP_STP1, while the reverse is tru for the remaining three instances. As with SP_CTE3 and SP_CTE4, the service rate is slightly more than that of the Myopic ILP policy but the cumulative profit is significantly greater.

The link priority vector returned by the kNN multi-output regressor is adaptive and different in each interval, since it is extremely unlikely for the input snapshot vector of link occupancy to be identical in two consecutive intervals. For this reason, it is difficult to establish a prevailing improvement over constant α-profiles with step-wise profiles. In Table 5 are summarized the cumulative profits acquired over the entire time horizon for all surrogate objective functions, in the form of a heat map.

Table 3
Surrogate ILP policy with constant α-profile vs. Myopic ILP policy.

Instance	Total arrived	Myopic ILP		α-profile	Service rate	Profit	Profit gap	Profit gap %	Service gap	Service gap %
		Service rate	Profit							
0	1224	12.8%	968	SP_CTE1	7.4%	780	−188	−19.4%	−66	−5.4%
				SP_CTE2	11.8%	1229	261	27.0%	−12	−1.0%
				SP_CTE3	14.7%	1440	472	48.8%	23	1.9%
				SP_CTE4	15.7%	1411	443	45.8%	35	2.9%
				SP_CTE5	1.2%	74	−894	−92.4%	−142	−11.6%
1	1230	10.7%	892	SP_CTE1	8.8%	920	28	3.1%	−24	−1.9%
				SP_CTE2	13.9%	1415	523	58.6%	39	3.2%
				SP_CTE3	12.1%	1173	281	31.5%	17	1.4%
				SP_CTE4	16.5%	1506	614	68.8%	71	5.8%
				SP_CTE5	0.4%	29	−863	−96.7%	−127	−10.3%
2	1208	12.9%	1045	SP_CTE1	9.9%	1042	−3	−0.3%	−37	−3.0%
				SP_CTE2	12.0%	1203	158	15.1%	−11	−0.9%
				SP_CTE3	14.3%	1350	305	29.2%	17	1.4%
				SP_CTE4	15.0%	1312	267	25.6%	25	2.1%
				SP_CTE5	1.2%	83	−962	−92.1%	−142	−11.7%
3	1182	13.7%	1069	SP_CTE1	8.2%	832	−237	−22.2%	−65	−5.5%
				SP_CTE2	12.7%	1261	192	18.0%	−12	−1.0%
				SP_CTE3	14.8%	1447	378	35.4%	13	1.1%
				SP_CTE4	17.8%	1603	534	50.0%	48	4.1%
				SP_CTE5	1.2%	91	−978	−91.5%	−148	−12.5%
4	1210	14.0%	1045	SP_CTE1	7.3%	748	−297	−28.4%	−81	−6.7%
				SP_CTE2	11.5%	1161	116	11.1%	−30	−2.5%
				SP_CTE3	15.8%	1491	446	42.7%	22	1.8%
				SP_CTE4	17.3%	1529	484	46.3%	40	3.3%
				SP_CTE5	0.0%	0	−1045	−100.0%	−169	−14.0%

Table 4
Surrogate ILP policy with step-wise α-profile vs. Myopic ILP policy.

Instance	Total arrived	Myopic ILP		α-profile	Service rate	Profit	Profit gap	Profit gap %	Service gap	Service gap %
		Service rate	Profit							
0	1224	12.8%	968	SP_STP1	15.2%	1442	474	49.0%	29	2.4%
				SP_STP2	16.8%	1533	565	58.4%	49	4.0%
1	1230	10.7%	892	SP_STP1	14.5%	1347	455	51.0%	46	3.7%
				SP_STP2	15.6%	1416	524	58.7%	60	4.9%
2	1208	12.9%	1045	SP_STP1	13.8%	1296	251	24.0%	11	0.9%
				SP_STP2	13.4%	1153	108	10.3%	6	0.5%
3	1182	13.7%	1069	SP_STP1	17.3%	1596	527	49.3%	43	3.6%
				SP_STP2	15.4%	1397	328	30.7%	20	1.7%
4	1210	14.0%	1045	SP_STP1	15.0%	1343	298	28.5%	13	1.1%
				SP_STP2	14.2%	1242	197	18.9%	3	0.2%

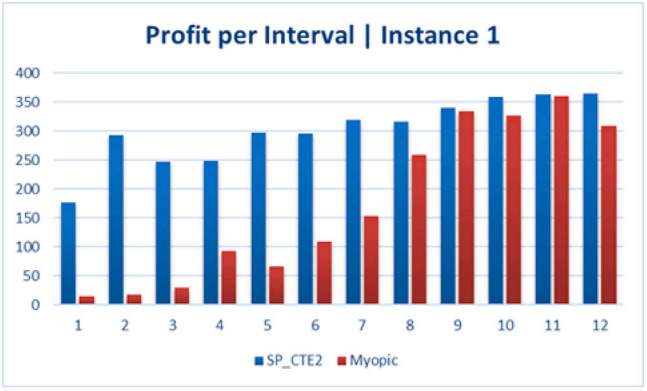
Table 5
Heat map of cumulative profit.

	Profit							
Instance	Myopic	SP_CTE1	SP_CTE2	SP_CTE3	SP_CTE4	SP_CTE5	SP_STP1	SP_STP2
0	968	780	1229	1440	1411	74	1442	1533
1	892	920	1415	1173	1506	29	1347	1416
2	1045	1042	1203	1350	1312	83	1296	1153
3	1069	832	1261	1447	1603	91	1596	1397
4	1045	748	1161	1491	1529	0	1343	1242

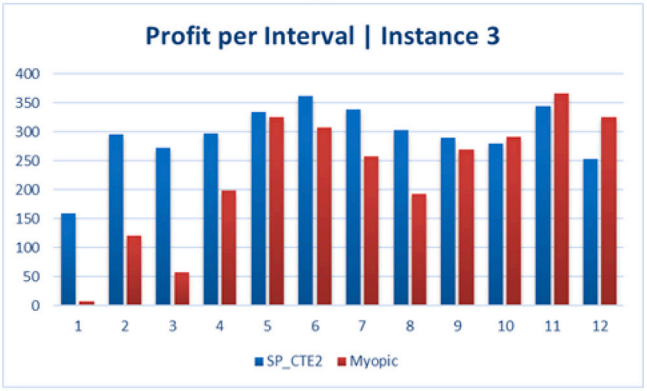
5.6. Profit acquired per interval

Fig. 6 illustrates the profit acquired per interval for test instances 1 and 3 by the Myopic ILP and SP_CTE2 algorithms. The results highlight a consistent performance advantage of SP_CTE2 over the Myopic ILP, even from the earliest intervals. This advantage can be attributed to two primary factors.

First, SP_CTE2 employs an adaptive strategy driven by its predictive kNN component. This enables the algorithm to prioritize and accept high-value requests from the very beginning of the planning horizon. The predictive capacity reservation mechanism



(a)



(b)

Fig. 6. Profit acquired per interval for Myopic ILP and SP_CTE2 in (a) Instance 1 and (b) Instance 3.

plays a critical role here, as it ensures that capacity is strategically reserved for such valuable requests, even while previously routed requests are still in transit, actively returning to their origin nodes. This forward-looking approach contrasts sharply with the Myopic ILP policy, which lacks the ability to anticipate future opportunities and instead focuses on immediate gains. Second, the reliance of the Myopic ILP policy on short-sighted decision-making often results in suboptimal choices, which have prolonged negative effects. Once a request is accepted and routed by the Myopic ILP, it occupies network capacity until the request completes its round trip, returning to its origin node. Poorly informed decisions made early on can therefore constrain the network's ability to accommodate more profitable requests in subsequent intervals, compounding the inefficiency.

In contrast, SP_CTE2 mitigates these issues through its predictive capacity reservation and adaptive decision-making, which collectively enhance its ability to optimize the allocation of network resources over time, leading to sustained superior performance across intervals.

5.7. *Significance of* $\beta_{i,l}$

This section demonstrates the role of using data-driven predictive prescriptions to determine efficient link priority vectors $\vec{\boldsymbol{\beta}}_{\boldsymbol{i}} = \left[\beta_{i,l}\right]^{l\in A}$. The value of these estimations of link priority can be exhibited by comparison to an arbitrary policy which assumes equal importance (or priority) of all links in the network. Such a policy would assign a uniform link priority across the network. It would thereby delegate all responsibility for managing the differential slack in the network to the α-profile. Yet it would suffer from the disadvantage of being unable to create reserve capacity differentiated across space (links in the network), and would only do this across time (varying α_i across intervals). To this end, experiments were conducted using the Surrogate ILP policy SP_CTE3 with a uniform standardized link priority vector $\vec{\boldsymbol{\beta}}_{\boldsymbol{i}} = \left[\frac{1}{|A|}\right]^{l\in A}$ and report the outcome in Table 6. We observed that upon using a uniform link priority vector, the performance dropped sharply and in many cases even fell below the Myopic ILP policy benchmark. This shows that uniform slack creation across all network links fails to improve performance because the reserve capacities are introduced in a manner which cannot be taken advantage of. It is crucial to create reserve capacity differentiated across both space and time to outperform the benchmark of the Myopic ILP.

Table 6
Performance comparison with uniform β.

Instance	Total arrived	Myopic ILP		α-profile	β Application	Service rate	Profit	Profit gap	Profit gap %	Service gap	Service gap %
		Service rate	Profit								
0	1224	12.8%	968	SP_CTE3	kNN	14.7%	1440	472	48.8%	23	1.9%
					uniform	12.3%	952	−16	−1.7%	−7	−0.6%
1	1230	10.7%	892	SP_CTE3	kNN	12.1%	1173	281	31.5%	17	1.4%
					uniform	13.0%	926	34	3.8%	28	2.3%
2	1208	12.9%	1045	SP_CTE3	kNN	14.3%	1350	305	29.2%	17	1.4%
					uniform	11.3%	1089	44	4.2%	-19	−1.6%
3	1182	13.7%	1069	SP_CTE3	kNN	14.8%	1447	378	35.4%	13	1.1%
					uniform	11.4%	1076	7	0.7%	-27	−2.3%
4	1210	14.0%	1045	SP_CTE3	kNN	15.8%	1491	446	42.7%	22	1.8%
					uniform	15.3%	1095	50	4.8%	16	1.3%

Table 7
Performance comparison of training schemes with and without simulation of virtual requests, for SP_CTE3.

Instance	Total arrived	Myopic ILP		Algorithm	Virtual simulation	Service rate	Profit	Profit gap	Profit gap %	Service gap	Service gap %
		Service rate	Profit								
0	1224	0.12827	968	SP_CTE3	yes	14.7%	1440	472	48.8%	23	1.9%
					no	16.5%	1316	1316	35.4%	45	3.7%
1	1230	0.10732	892	SP_CTE3	yes	12.1%	1173	281	31.5%	17	1.4%
					no	14.0%	1031	139	15.6%	40	3.2%
2	1208	0.12914	1045	SP_CTE3	yes	14.3%	1350	305	29.2%	17	1.4%
					no	15.6%	1191	146	14.0%	32	2.7%
3	1182	0.13706	1069	SP_CTE3	yes	14.8%	1447	378	35.4%	13	1.1%
					no	13.4%	1361	292	27.3%	-4	−0.3%
4	1210	0.13967	1045	SP_CTE3	yes	15.8%	1491	446	42.7%	22	1.8%
					no	14.2%	1380	335	32.1%	3	0.2%

5.8. Training scheme without virtual request simulation

This section evaluates the performance of the proposed framework under a modified training approach that excludes the use of simulated virtual requests. Instead, the predictive ML component is trained using a large training instance consisting solely of real requests. The lookahead mechanism is retained but operates exclusively on routing decisions for real requests over a rolling horizon of three intervals. This configuration examines whether the proposed methodology can remain effective without relying on the generation of virtual requests.

The results, summarized in Table 7 indicate that the training scheme that excludes the simulation of virtual requests demonstrates slightly lower performance in terms of profit acquired. This outcome may be attributed to the limited diversity of training data, as virtual requests, to some extent, enrich the dataset with scenarios that might not be naturally present when only real requests are entertained. However, the Surrogate ILP model showcases its robustness by maintaining competitive performance even under this adjusted training procedure, highlighting its adaptability and effectiveness across varying methodologies.

6. Conclusion and perspectives

This study addresses the stochastic online version of the drone delivery service planning problem in urban airspace. The solution methods we explore are executed from the perspective of a network manager. Given an incoming Poisson process of drone delivery trip requests, we aim to find the optimal assignment of drones to space–time trajectories in an urban air traffic network within a predefined time horizon. We assume that drone deliveries must respect delivery time windows and that UAVs must fly at constant speed throughout the network to avoid airborne delays and congestion. The MDP framework plays a crucial role in structuring the problem, providing a preliminary mathematical representation of uncertainty and constraints. By explicitly modeling the reward function's impact on policy design, the MDP reveals how different reward formulations can influence long-term performance, enabling more effective optimization strategies. Our main contribution is to expand the integer linear programming formulation of the deterministic DDSP to the stochastic variant with online demand. We also propose a novel solution methodology which leverages data-driven, machine learning-enabled predictions of network link priorities to construct a modified surrogate objective function. We outline a training scheme to train the machine learning component as well. The numerical results indicate that the Surrogate ILP policy outperforms the Myopic ILP policy, which iteratively solves for the objective of profit maximization. We demonstrate that even though the training procedure might be extensive, it is a one-time investment that returns rewards in the form of a sustained profit margin over the Myopic ILP policy across multiple instances.

kNN was chosen as the ML component for its simplicity, interpretability, and minimal parameter tuning, making it well-suited for this research. Its non-parametric nature allows flexibility in capturing complex relationships without assuming prior data distributions, avoiding over-fitting and extensive hyper-parameter optimization. As a baseline, kNN validates the integration of machine learning into our data-driven optimization framework, enabling interpretable and traceable predictive prescriptions that align with decision-making objectives. Success with this straightforward approach strengthens the framework's robustness and provides a foundation for exploring advanced models like neural networks or gradient boosting in future work. The design of the surrogate objective function used within the surrogate ILP policy also warrants further examination. The original objective function focuses on maximizing total profits and does not directly capture network features such as link occupancy or transportation costs. However, these features implicitly affect service rate since the latter is dependent on network resources. Alternative objective function designs that exploit the inter-dependence of link capacity with service rate may thus be investigated.

The escalating adoption of digital technology by the public and private sectors is producing real-time uncertain demand for logistic service providers globally. This is happening concurrently with the accelerating adoption of UAVs in transport fleets to service regions with varying degrees of urbanization. The utility of the solution methodology proposed in this study lies at the intersection of these two trends. By outperforming conventional online optimization methods our approach unlocks considerable benefits for the service provider and for the customers. Moreover, we successfully establish link priority and available link capacity as pivotal factors for enhancing network performance. This implies that the strategy and procedure of non-myopic anticipatory resource allocation we present in this paper is transferable to other classes of network-based logistic problems such as pickup and delivery logistics and ride-sharing.

Multiple directions for further research can be identified. First, there is an opportunity to extend the DDO framework to learn efficient α-profiles automatically. This requires a training scheme that explores the α-profile space quickly and intelligently under time and computational budgets. Second, modifications can be imposed on the problem definition to make the objective function more realistic. These include transportation costs, battery swaps and charging limitations. The incorporation of transportation cost improves scalability, as with large fleets this cost becomes a considerable factor in service planning. Third, future research could also address various operational strategies, such as multiple deliveries per UAV, parcel exchanges and delivery consolidation via parcel lockers. Lastly, the model constraints can be expanded to include capacity restrictions on each UAV and differential velocities. Additionally, incorporating stochastic link capacities due to weather uncertainties as well as the introduction and placement of charging platforms at selected nodes or a centralized drone hub within the network presents a compelling network-design possibility. While extensively studied in the context of vehicle routing models, few studies have tackled such modifications in an online demand setting with time-dependent routing considerations.

Furthermore, future studies could investigate scenarios involving multiple operators or fleet managers, each employing independent strategies or variants of the online delivery service planning algorithm. This would require exploring mechanisms for inter-operator coordination, such as exchanging request assignments or leveraging centralized airspace management systems. Such research would provide valuable insights into managing airspace congestion, ensuring safety, and addressing operational challenges in multi-operator environments.

CRediT authorship contribution statement

Aditya Paul: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Michael W. Levin:** Writing – review & editing, Validation, Methodology, Conceptualization. **S. Travis Waller:** Writing – review & editing, Validation, Supervision, Project administration, Funding acquisition, Conceptualization. **David Rey:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: David Rey reports financial support was provided by University of New South Wales. Travis Waller reports financial support was provided by University of New South Wales. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Both S. Travis Waller and David Rey have received financial support from the Australian Research Council Discovery Project DP210103138.

Appendix

The Myopic ILP formulation is presented below:

$$\max\ Z = \sum_{r\in\mathcal{R}_O^i} z_r p_r \tag{A.1a}$$

$$\text{s.t.} \sum_{i\in\Gamma_{o_r}^+}\sum_{t=e_r}^{T} \chi_a^{r\uparrow}(t) = z_r \qquad \forall r\in\mathcal{R}_O^i \tag{A.1b}$$

$$\sum_{i\in\Gamma_{d_r}^-}\sum_{t=l_r}^{u_r} \chi_a^{r\downarrow}(t) = \sum_{i\in\Gamma_{o_r}^+}\sum_{t=e_r}^{T} \chi_a^{r\uparrow}(t) \qquad \forall r\in\mathcal{R}_O^i \tag{A.1c}$$

$$\sum_{a\in\Gamma_{o_r}^+}\sum_{t=e_r}^{T} \chi_a^{r\uparrow}(t) = \sum_{a\in\Gamma_{o_r}^-}\sum_{t=l_r+STT}^{u_r+LTT} \chi_a^{r\downarrow}(t) \qquad \forall r\in\mathcal{R}_O^i \tag{A.1d}$$

$$\chi_a^{r\downarrow}\left(t+\frac{\rho_i}{v}\right) = \chi_a^{r\uparrow}(t) \qquad \forall r\in\mathcal{R}_O^i, \forall a\in A, \forall t\in[(i-1)D, iD] \tag{A.1e}$$

$$\sum_{r\in\mathcal{R}_O^i} \chi_a^{r\uparrow}(t) \le C_a \qquad \forall a\in A, \forall t\in[(i-1)D, iD] \tag{A.1f}$$

$$\sum_{r\in\mathcal{R}_O^i} \chi_a^{r\downarrow}(t) \le C_a \qquad \forall a\in A, \forall t\in[(i-1)D, iD] \tag{A.1g}$$

$$\sum_{i\in\Gamma_n^-} \chi_a^{r\downarrow}(t) = \sum_{i\in\Gamma_n^+} \chi_a^{r\uparrow}(t) \qquad \forall n\in N, \forall r\in\mathcal{R}_O^i, \forall t\in[(i-1)D, iD] \tag{A.1h}$$

$$\gamma_{ab}^r(t) \le \chi_a^{r\downarrow}(t) \qquad \forall r\in\mathcal{R}_O^i, \forall (a,b)\in A^2, \forall t\in[(i-1)D, iD] \tag{A.1i}$$

$$\gamma_{ab}^r(t) \le \chi_b^{r\uparrow}(t) \qquad \forall r\in\mathcal{R}_O^i, \forall (a,b)\in A^2, \forall t\in[(i-1)D, iD] \tag{A.1j}$$

$$\gamma_{ab}^r(t) \ge \chi_a^{r\downarrow}(t) + \chi_b^{r\uparrow}(t) - 1 \qquad \forall r\in\mathcal{R}_O^i, \forall (a,b)\in A^2, \forall t\in[(i-1)D, iD] \tag{A.1k}$$

$$\phi_{ab}(t) \ge \frac{1}{|\mathcal{R}_O^i|}\sum_{r\in\mathcal{R}_O^i} \gamma_{ab}^r(t) \qquad \forall (a,b)\in A^2, \forall t\in[(i-1)D, iD] \tag{A.1l}$$

$$\phi_{ab}(t) + \sum_{(i',j')\in C_{ab}} \phi_{i'j'}(t) \le 1 \qquad \forall (a,b)\in A^2, \forall t\in[(i-1)D, iD] \tag{A.1m}$$

$$\gamma_{ab}(t)\in\{0,1\} \qquad \forall r\in\mathcal{R}_O^i, \forall (a,b)\in A^2, \forall t\in[(i-1)D, iD] \tag{A.1n}$$

$$\phi_{ab}^r(t)\in\{0,1\} \qquad \forall (a,b)\in A^2, \forall t\in[(i-1)D, iD] \tag{A.1o}$$

$$z_r = 1 \qquad \forall r\in\mathcal{R}_I\cup\mathcal{R}_A \tag{A.1p}$$

$$\chi_{\omega(n_{k-1},n_k)}^{r\downarrow}(t_{n_k}^r) = 1 \qquad \forall r\in\mathcal{R}_A, \forall\ n_k\in\{\mu_r \mid (i-1)D\le t_{n_k}^r\le iD \wedge k\ne 1\} \tag{A.1q}$$

$$\chi_{\omega(n_{k-1},n_k)}^{r\uparrow}(t_{n_k}^r) = 1 \qquad \forall r\in\mathcal{R}_A, \forall\ n_k\in\{\mu_r \mid (i-1)D\le t_{n_k}^r\le iD \wedge k\ne 1\} \tag{A.1r}$$

$$\gamma_{\omega(n_{k-1},n_k)\omega(n_k,n_{k+1})}(t_{n_k}^r) = 1 \qquad \forall r\in\mathcal{R}_A, \forall\ n_k\in\{\mu_r \mid (i-1)D\le t_{n_k}^r\le iD \wedge n_k\ne o_r\} \tag{A.1s}$$

$$\chi_a^{r\uparrow}(t), \chi_a^{r\downarrow}(t)\in\{0,1\} \qquad \forall r\in\mathcal{R}_O^i, \forall a\in A, \forall t\in[(i-1)D, iD] \tag{A.1t}$$

$$z_r\in\{0,1\} \qquad \forall r\in\mathcal{R}_O^i \tag{A.1u}$$

The Surrogate ILP is presented below:

$$\max\ Z = \sum_{r\in\mathcal{R}_O^i} z_r p_r\ +\ \alpha_i \sum_{l\in A}\beta_{i,l}\sum_{t=(i-1)D}^{iD}\left[C_l - \sum_{r\in\mathcal{R}_O^i}\chi_l^{r\uparrow}(t)\right] \tag{A.2a}$$

$$\text{s.t.} \sum_{i\in\Gamma_{o_r}^+}\sum_{t=e_r}^{T} \chi_a^{r\uparrow}(t) = z_r \qquad \forall r\in\mathcal{R}_O^i \tag{A.2b}$$

$$\sum_{i\in\Gamma_{d_r}^-}\sum_{t=l_r}^{u_r} \chi_a^{r\downarrow}(t) = \sum_{i\in\Gamma_{o_r}^+}\sum_{t=e_r}^{T} \chi_a^{r\uparrow}(t) \qquad \forall r\in\mathcal{R}_O^i \tag{A.2c}$$

$$\sum_{a\in\Gamma_{o_r}^+}\sum_{t=e_r}^{T} \chi_a^{r\uparrow}(t) = \sum_{a\in\Gamma_{o_r}^-}\sum_{t=l_r+STT}^{u_r+LTT} \chi_a^{r\downarrow}(t) \qquad \forall r\in\mathcal{R}_O^i \tag{A.2d}$$

$$\chi_a^{r\downarrow}\left(t+\frac{\rho_i}{v}\right)=\chi_a^{r\uparrow}(t) \qquad \forall r\in\mathcal{R}_O^i,\forall a\in A,\forall t\in[(i-1)D,iD] \tag{A.2e}$$

$$\sum_{r\in\mathcal{R}_O^i}\chi_a^{r\uparrow}(t)\le C_a \qquad \forall a\in A,\forall t\in[(i-1)D,iD] \tag{A.2f}$$

$$\sum_{r\in\mathcal{R}_O^i}\chi_a^{r\downarrow}(t)\le C_a \qquad \forall a\in A,\forall t\in[(i-1)D,iD] \tag{A.2g}$$

$$\sum_{i\in\Gamma_n^-}\chi_a^{r\downarrow}(t)=\sum_{i\in\Gamma_n^+}\chi_a^{r\uparrow}(t) \qquad \forall n\in N,\forall r\in\mathcal{R}_O^i,\forall t\in[(i-1)D,iD] \tag{A.2h}$$

$$\gamma_{ab}^r(t)\le\chi_a^{r\downarrow}(t) \qquad \forall r\in\mathcal{R}_O^i,\forall(a,b)\in A^2,\forall t\in[(i-1)D,iD] \tag{A.2i}$$

$$\gamma_{ab}^r(t)\le\chi_b^{r\uparrow}(t) \qquad \forall r\in\mathcal{R}_O^i,\forall(a,b)\in A^2,\forall t\in[(i-1)D,iD] \tag{A.2j}$$

$$\gamma_{ab}^r(t)\ge\chi_a^{r\downarrow}(t)+\chi_b^{r\uparrow}(t)-1 \qquad \forall r\in\mathcal{R}_O^i,\forall(a,b)\in A^2,\forall t\in[(i-1)D,iD] \tag{A.2k}$$

$$\phi_{ab}(t)\ge\frac{1}{|\mathcal{R}_O^i|}\sum_{r\in\mathcal{R}_O^i}\gamma_{ab}^r(t) \qquad \forall(a,b)\in A^2,\forall t\in[(i-1)D,iD] \tag{A.2l}$$

$$\phi_{ab}(t)+\sum_{(i',j')\in C_{ab}}\phi_{i'j'}(t)\le 1 \qquad \forall(a,b)\in A^2,\forall t\in[(i-1)D,iD] \tag{A.2m}$$

$$\gamma_{ab}(t)\in\{0,1\} \qquad \forall r\in\mathcal{R}_O^i,\forall(a,b)\in A^2,\forall t\in[(i-1)D,iD] \tag{A.2n}$$

$$\phi_{ab}^r(t)\in\{0,1\} \qquad \forall(a,b)\in A^2,\forall t\in[(i-1)D,iD] \tag{A.2o}$$

$$z_r=1 \qquad \forall r\in\mathcal{R}_I\cup\mathcal{R}_A \tag{A.2p}$$

$$\chi_{\omega(n_{k-1},n_k)}^{r\downarrow}(t_{n_k}^r)=1 \qquad \forall r\in\mathcal{R}_A,\forall\, n_k\in\{\mu_r\mid(i-1)D\le t_{n_k}^r\le iD\wedge k\ne 1\} \tag{A.2q}$$

$$\chi_{\omega(n_{k-1},n_k)}^{r\uparrow}(t_{n_k}^r)=1 \qquad \forall r\in\mathcal{R}_A,\forall\, n_k\in\{\mu_r\mid(i-1)D\le t_{n_k}^r\le iD\wedge k\ne 1\} \tag{A.2r}$$

$$\gamma_{\omega(n_{k-1},n_k)\omega(n_k,n_{k+1})}(t_{n_k}^r)=1 \qquad \forall r\in\mathcal{R}_A,\forall\, n_k\in\{\mu_r\mid(i-1)D\le t_{n_k}^r\le iD\wedge n_k\ne o_r\} \tag{A.2s}$$

$$\chi_a^{r\uparrow}(t),\chi_a^{r\downarrow}(t)\in\{0,1\} \qquad \forall r\in\mathcal{R}_O^i,\forall a\in A,\forall t\in[(i-1)D,iD] \tag{A.2t}$$

$$z_r\in\{0,1\} \qquad \forall r\in\mathcal{R}_O^i \tag{A.2u}$$

Data availability

Data will be made available on request.

References

Ackerman, E., Koziol, M., 2019. The blood is here: Zipline's medical delivery drones are changing the game in Rwanda. IEEE Spectr. 56 (5), 24–31.

Asadi, A., Pinkley, S.N., Mes, M., 2022. A Markov decision process approach for managing medical drone deliveries. Expert Syst. Appl. 204, 117490.

Baloch, G., Gzara, F., 2020. Strategic network design for parcel delivery with drones under competition. Transp. Sci. 54 (1), 204–228.

Basso, R., Kulcsár, B., Sanchez-Diaz, I., Qu, X., 2022. Dynamic stochastic electric vehicle routing with safe reinforcement learning. Transp. Res. Part E: Logist. Transp. Rev. 157, 102496.

Baty, L., Jungel, K., Klein, P.S., Parmentier, A., Schiffer, M., 2024. Combinatorial optimization-enriched machine learning to solve the dynamic vehicle routing problem with time windows. Transp. Sci.

Benarbia, T., Kyamakya, K., 2021. A literature review of drone-based package delivery logistics systems and their implementation feasibility. Sustainability 14 (1), 360.

Bengio, Y., Lodi, A., Prouvost, A., 2021. Machine learning for combinatorial optimization: a methodological tour d'horizon. European J. Oper. Res. 290 (2), 405–421.

Bertsimas, D., Jaillet, P., Martin, S., 2019. Online vehicle routing: The edge of optimization in large-scale applications. Oper. Res. 67 (1), 143–162.

Bertsimas, D., Kallus, N., 2020. From predictive to prescriptive analytics. Manag. Sci. 66 (3), 1025–1044.

Bonami, P., Lodi, A., Zarpellon, G., 2018. Learning a classification of mixed-integer quadratic programming problems. In: International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Springer, pp. 595–604.

Chen, H., Hu, Z., Solak, S., 2021. Improved delivery policies for future drone-based delivery systems. European J. Oper. Res. 294 (3), 1181–1201.

Chen, X., Ulmer, M.W., Thomas, B.W., 2022. Deep Q-learning for same-day delivery with vehicles and drones. European J. Oper. Res. 298 (3), 939–952.

Chen, X., Wang, T., Thomas, B.W., Ulmer, M.W., 2023. Same-day delivery with fair customer service. European J. Oper. Res. 308 (2), 738–751.

Chu, H., Zhang, W., Bai, P., Chen, Y., 2021. Data-driven optimization for last-mile delivery. Complex & Intell. Syst. 1–14.

Dayarian, I., Savelsbergh, M., Clarke, J.-P., 2020. Same-day delivery with drone resupply. Transp. Sci. 54 (1), 229–249.

Edwards, J., McKinnon, A., Cherrett, T., McLeod, F., Song, L., 2010. Carbon dioxide benefits of using collection–delivery points for failed home deliveries in the United Kingdom. Transp. Res. Rec. 2191 (1), 136–143.

Elmachtoub, A.N., Grigas, P., 2022. Smart "Predict, then Optimize". Manag. Sci. 68 (1), 9–26.

Feng, S., Duan, P., Ke, J., Yang, H., 2022. Coordinating ride-sourcing and public transport services with a reinforcement learning approach. Transp. Res. Part C: Emerg. Technol. 138, 103611.

Fruhling, A.L., Digman, L.A., 2000. The impact of electronic commerce on business-level strategies. J. Electron. Commer. Res. 1 (1), 13.

Ghiasvand, M.R., Rahmani, D., Moshref-Javadi, M., 2024. Data-driven robust optimization for a multi-trip truck-drone routing problem. Expert Syst. Appl. 241, 122485.

Gong, H., Huang, B., Jia, B., Dai, H., 2023. Modelling power consumptions for multi-rotor UAVs. IEEE Trans. Aerosp. Electron. Syst.

Gu, R., Liu, Y., Poon, M., 2023. Dynamic truck–drone routing problem for scheduled deliveries and on-demand pickups with time-related constraints. Transp. Res. Part C: Emerg. Technol. 151, 104139.
Han, B.R., Sun, T., Chu, L.Y., Wu, L., 2022. COVID-19 and E-commerce operations: Evidence from Alibaba. Manuf. Serv. Oper. Manag. 24 (3), 1388–1405.
Hecken, T., Cumnuantip, S., Klimmek, T., 2022. Structural design of heavy-lift unmanned cargo drones in low altitudes. Autom. Low- Alt. Air Delivery: Towar. Auton. Cargo Transp. Drones 159–183.
Hong, J., Lee, M., Cheong, T., Lee, H.C., 2019. Routing for an on-demand logistics service. Transp. Res. Part C: Emerg. Technol. 103, 328–351.
Kim, J., Park, J., 2005. A consumer shopping channel extension model: attitude shift toward the online store. J. Fash. Mark. Management: An Int. J. 9 (1), 106–121.
Kornatowski, P.M., Feroskhan, M., Stewart, W.J., Floreano, D., 2020. A morphing cargo drone for safe flight in proximity of humans. IEEE Robot. Autom. Lett. 5 (3), 4233–4240.
Kruber, M., Lübbecke, M.E., Parmentier, A., 2017. Learning when to use a decomposition. In: International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Springer, pp. 202–210.
Larsen, E., Lachapelle, S., Bengio, Y., Frejinger, E., Lacoste-Julien, S., Lodi, A., 2022. Predicting tactical solutions to operational planning problems under imperfect information. INFORMS J. Comput. 34 (1), 227–242.
Lemardelé, C., Estrada, M., Pagès, L., Bachofner, M., 2021. Potentialities of drones and ground autonomous delivery devices for last-mile logistics. Transp. Res. Part E: Logist. Transp. Rev. 149, 102325.
Levin, M.W., Rey, D., 2023. Branch-and-price for drone delivery service planning in urban airspace. Transp. Sci. 57 (4), 843–865.
Li, X., Yan, P., Yu, K., Li, P., Liu, Y., 2024. Parcel consolidation approach and routing algorithm for last-mile delivery by unmanned aerial vehicles. Expert Syst. Appl. 238, 122149.
Liu, Y., 2019. An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones. Comput. Oper. Res. 111, 1–20.
Liu, Y., Wu, F., Lyu, C., Li, S., Ye, J., Qu, X., 2022. Deep dispatching: A deep reinforcement learning approach for vehicle dispatching on online ride-hailing platform. Transp. Res. Part E: Logist. Transp. Rev. 161, 102694.
Lodi, A., Zarpellon, G., 2017. On learning and branching: a survey. Trans. Oper. Res. 25 (2), 207–236.
Lozzi, G., Iannaccone, G., Maltese, I., Gatta, V., Marcucci, E., Lozzi, R., 2022. On-demand logistics: Solutions, barriers, and enablers. Sustainability 14 (15), 9465.
Masorgo, N., Mir, S., Hofer, A.R., 2023. You're driving me crazy! how emotions elicited by negative driver behaviors impact customer outcomes in last mile delivery. J. Bus. Logist.
Mohsan, S.A.H., Othman, N.Q.H., Li, Y., Alsharif, M.H., Khan, M.A., 2023. Unmanned aerial vehicles (UAVs): Practical aspects, applications, open challenges, security issues, and future trends. Intell. Serv. Robot. 16 (1), 109–137.
Moshref-Javadi, M., Winkenbach, M., 2021. Applications and research avenues for drone-based models in logistics: A classification and review. Expert Syst. Appl. 177, 114854.
Muñoz-Villamizar, A., Velazquez-Martínez, J.C., Caballero-Caballero, S., 2024. A large-scale last-mile consolidation model for e-commerce home delivery. Expert Syst. Appl. 235, 121200.
Na, H., Chang, Y.S., 2024. Study on the policies, regulations of drone-based logistics. In: 2024 2nd International Conference on Cyber Resilience. ICCR, IEEE, pp. 1–4.
Radoglou-Grammatikis, P., Sarigiannidis, P., Lagkas, T., Moscholios, I., 2020. A compilation of UAV applications for precision agriculture. Comput. Netw. 172, 107148.
Rave, A., Fontaine, P., Kuhn, H., 2023. Drone location and vehicle fleet planning with trucks and aerial drones. European J. Oper. Res. 308 (1), 113–130.
Rejeb, A., Rejeb, K., Simske, S.J., Treiblmaier, H., 2023. Drones for supply chain management and logistics: a review and research agenda. Int. J. Logist. Res. Appl. 26 (6), 708–731.
Sadana, U., Chenreddy, A., Delage, E., Forel, A., Frejinger, E., Vidal, T., 2023. A survey of contextual optimization methods for decision making under uncertainty. arXiv preprint arXiv:2306.10374.
Shivgan, R., Dong, Z., 2020. Energy-efficient drone coverage path planning using genetic algorithm. In: 2020 IEEE 21st International Conference on High Performance Switching and Routing. HPSR, IEEE, pp. 1–6.
Song, L., Cherrett, T., McLeod, F., Guan, W., 2009. Addressing the last mile problem: transport impacts of collection and delivery points. Transp. Res. Rec. 2097 (1), 9–18.
Tran-Thanh, L., Chapman, A., Rogers, A., Jennings, N., 2012. Knapsack based optimal policies for budget–limited multi–armed bandits. In: Proceedings of the AAAI Conference on Artificial Intelligence. 26, pp. 1134–1140.
Van Duin, J., de Goffau, W., Wiegmans, B., Tavasszy, L., Saes, M., 2016. Improving home delivery efficiency by using principles of address intelligence for B2C deliveries. Transp. Res. Procedia 12, 14–25.
Van Hentenryck, P., Bent, R., Upfal, E., 2010. Online stochastic optimization under time constraints. Ann. Oper. Res. 177, 151–183.
Verhoef, P.C., Kannan, P.K., Inman, J.J., 2015. From multi-channel retailing to omni-channel retailing: introduction to the special issue on multi-channel retailing. J. Retail. 91 (2), 174–181.
Vlahovic, N., Knezevic, B., Batalic, P., 2017. Implementing delivery drones in logistics business process: Case of pharmaceutical industry. Int. J. Mech. Ind. Eng. 10 (12), 4026–4031.
Yan, R., Wang, S., 2022. Integrating prediction with optimization: Models and applications in transportation management. Multimodal Transp. 1 (3), 100018.
Yang, Y., Yan, C., Cao, Y., Roberti, R., 2023. Planning robust drone-truck delivery routes under road traffic uncertainty. European J. Oper. Res. 309 (3), 1145–1160.
Yilmaz, D., Büyüktahtakın, İ.E., 2024. An expandable machine learning-optimization framework to sequential decision-making. European J. Oper. Res. 314 (1), 280–296.