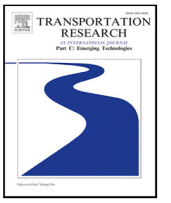
Transportation Research Part C 152 (2023) 104147

Contents lists available at ScienceDirect

Transportation Research Part C

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



# Routing battery-constrained delivery drones in a depot network: A business model and its optimization–simulation assessment

Yanchao Liu[1]

*Department of Industrial and Systems Engineering, Wayne State University, Detroit, MI, USA*

ARTICLE INFO

*Keywords:*
Drone delivery
Column generation
Dynamic vehicle routing
Optimization

ABSTRACT

This paper proposes a novel business model for on-demand package shipment services using drones, and evaluates different modeling and solution approaches for the drone routing problem that underpins the service operation. In the proposed service, customers' shipment orders of arbitrary origins and destinations, payload weights and bid values are collected every five minutes, and available drones from multiple depots are then dispatched to fulfill a subset of these orders in a way to maximize profit. A drone path starts from a depot, serves one or more customer orders in sequence, and ends at a depot for battery recharging, which incurs a fixed cost. Two mixed integer programming (MIP) formulations are presented to model the drone dispatch and routing problem. To improve solution efficiency, three computational approaches, including two column generation based algorithms and a brute-force path enumeration algorithm, are developed and compared. Computational experiments suggest, somewhat surprisingly, that the brute-force approach is the most effective and most scalable one, outperforming other alternatives by a substantial margin in both computing time and solution quality. Furthermore, an optimization–simulation framework is proposed to assess the system performance over a long horizon that spans multiple dispatch periods without complicating the optimization model. Using the simulation framework, useful managerial insights including the effects of battery capacity, wind condition and computing capability on the fleet dispatch operation, are generated, which will guide real-world implementations of the new business model.

## 1. Introduction

Unmanned aircraft (UA), or drones, are able to transport light-weight goods for a short distance faster and more economically than other means. Drone-based on-demand delivery is envisioned by many to be a promising new business model. Example services include delivering restaurant meals to customers, shipping urgently needed items from warehouses to shops, and transporting blood samples from clinics to test labs, etc.

Through a National Science Foundation (NSF) supported Innovation Corps (I-Corps) project, our team conducted extensive market readiness research in 2020 with an attempt to commercialize the drone routing technologies developed earlier, see Liu (2019b,a, 2020). We interviewed more than 100 people that were hypothesized to play key roles in the drone delivery ecosystem, if one was established. In the project, we have identified a number of barriers to a profitable business model, one of which was the relatively high cost of Lithium Polymer (LiPo) battery. LiPo battery is the most widely used power source for small civilian drones. While the cost of electric energy consumed in a short trip is almost negligible, the trip's share of the battery's acquisition and maintenance

*E-mail address:* yanchaoliu@wayne.edu.
[1] Mailing: 4815 4th Street Rm 2169, Detroit, MI 48201, USA.

https://doi.org/10.1016/j.trc.2023.104147
Received 28 December 2021; Received in revised form 11 July 2022; Accepted 18 April 2023
Available online 6 May 2023

costs is not. Barring a new breakthrough in battery technology (or alternative power source technology), efficient fleet-level battery asset management will be the key factor that determines the bottom line of a drone-based delivery business.

In this paper, we propose a novel operations model for an on-demand package shipment service using drones, in which the batteries' capacity and life cycle costs are explicitly considered. The key operational decisions, i.e., which shipment orders to accept, which drones to dispatch, and what route for each drone to follow, are formulated into a mixed integer programming (MIP) model. The model is to be solved periodically in short intervals of time to support continuous operations. With effective preprocessing and transformation of input data, the model formulation is compact and intuitive, yet, even the state-of-the-art commercial software such as CPLEX is unable to solve reasonably sized instances within the desired time frame. To overcome this challenge, we investigate the path-based formulation along with different solution algorithms, including a parallel path enumeration algorithm, the column generation approach and a heuristic improvement of the column generation approach. While all these methods are able to substantially enhance the solvability of the problem as compared to the original compact formulation, the approach based on the parallel, exhaustive path enumeration algorithm is shown to be the most computationally viable. To aid the validation of the business model, we further integrate the routing optimization model in a simulation framework with which interested entrepreneurs and investors can simulate the daily operations and conduct what-if analyses with customized design parameters. As usage examples, the effects of battery capacity and wind speed and direction on the operational efficiency are explored in a simulation case study.

The remainder of the paper is organized as follows. In Section 2, the related literature is reviewed and our contributions are highlighted. Section 3 describes the business model and operation specifications. In Section 4, the battery consumption model and drone routing model are formulated. Section 5 develops the path-based formulation, the column enumeration algorithm and two column generation based algorithms for solving the drone routing problem. Section 6 proposes a simulation approach for addressing multi-period, continuous operation scenarios. Section 7 presents validation experiments and a simulation case study. Section 8 concludes the paper with pointers for future research.

## 2. Literature review

Drone-enabled delivery operations mainly fall into two categories, the drone-only model and the truck–drone tandem model. In the drone-only model, drones are the sole transporting vehicles operated by the service provider. In the truck–drone tandem model, drones are launched from conventional delivery trucks, and the two types of vehicles coordinate their routes to deliver the truckload of packages to geographically dispersed customers. Both models have company-backed operational prototypes in reality. For instance, Alphabet Wing has been running drone delivery trials in Christiansburg, Virginia, since 2019, and UPS, in partnership with a startup called Workhorse, has demonstrated the concept of launching a small package-delivery drone from the roof of a conventional delivery truck. However, most prototypes were only demonstrated in small-scale trial runs, and the actual commercial value gleaned from these trials has been scantly disclosed.

The truck–drone tandem model has attracted much interest from the operations research community, partly because the coordination and synchronization between the two types of vehicles bring about additional challenges upon the classic, yet $\mathcal{NP}$-hard, traveling salesman problem (TSP) and vehicle routing problem (VRP). The pioneering work along this line includes Murray and Chu (2015) and Agatz et al. (2018), which first proposed the concepts of flying sidekick traveling salesman problem (FSTSP) and of the traveling salesman problem with drones (TSP-D), respectively. Subsequently, a large volume of follow-on research has emerged. Prominent works include Yurek and Ozmutlu (2018), which decomposed the TSP-D into two stages and presented an iterative algorithm that solved the problem faster than CPLEX, Dell'Amico et al. (2019), which improved the solvability of FSTSP via a better formulation and heuristics, Roberti and Ruthmair (2021) which improved the solvability of TSP-D via a compact formulation and novel algorithm design, and Murray and Raj (2020), which extended the FSTSP by using multiple drones to aid the truck to complete the TSP tour. Jeong et al. (2019) incorporated the drone's energy consumption and no-fly zone considerations into the FSTSP framework and developed a novel search heuristic that performed competitively against several well-known metaheuristics. Truck–drone tandem has also been modeled as VRP variants. Wang and Sheu (2019) provided a path-based VRP model for a delivery system consisting of multiple trucks, drones, and docking nodes, and proposed a branch-and-price (i.e., column generation) algorithm for its solution. Wang et al. (2017) developed theoretical bounds for ratios of the total delivery time among different truck–drone delivery options and the insights were subsequently extended in Poikonen et al. (2017). Wang et al. (2020) proposed a bi-objective variant of the TSP-D problem to attain a compromise between the operational cost and completion time, and developed an improved non-dominated sorting genetic algorithm to solve the problem. Another form of truck–drone coordination was modeled through operating a mixed fleet — the drones would not be carried by trucks and both types of vehicles would deliver packages separately. Ulmer and Thomas (2018) proposed a same-day delivery service using a mixed fleet of trucks and drones. For each randomly arriving customer order, the service provider must decide whether or not the order can be served and whether a truck or a drone should perform the delivery. A Markov decision process was employed to find the optimal policy that maximizes the expected number of orders served in a day. Khoufi et al. (2019) provided a survey of recent TSP variants involving drones, and Chung et al. (2020) reviewed optimization models for a broader scope of applications that involved drone-truck tandem operation.

The drone-only model has been frequently formulated as variants of the vehicle routing problem. Most works acknowledged the carrying capacity and flight distance limitation, and featured innovative treatments. Dorling et al. (2017) formulated the drone delivery problem as a multi-trip vehicle routing problem (MTVRP) in which a single drone at a depot makes multiple trips to deliver to a set of customer sites. A highlight of this formulation was its explicit treatment for the relationship between the drone's energy consumption and the total takeoff weight. The resulting MIP with big-M constraints was challenging to solve, and simulated annealing algorithms were implemented for approximate solutions. Cheng et al. (2020) extended the MTVRP model by retaining

the more descriptive nonlinear relationship between a trip's energy consumption and its payload and travel distance. The nonlinear constraints were then approximated in an iterative solution process by linear cuts. Apart from the cost-saving potential, service agility has also been emphasized as an advantage of drone-enabled deliveries. The latter aspect is typically manifested through various forms of dynamism and stochasticity in real-world scenarios, such as the on-demand meal delivery services, see, e.g., Reyes et al. (2018), Liu (2019a). In particular, Reyes et al. (2018) characterized the meal delivery problem as having structural features such as "multiple pick-up points (restaurants)" and "dynamic order arrivals", among other features pertinent mainly to manned delivery systems (e.g., varying capacity throughout the day in the form of courier shifts). Both of the mentioned features are handled in this paper. Liu (2019a) used a comprehensive MIP to characterize the on-demand meal delivery service, and proposed to drive the practical operations by solving the MIP in a rolling-horizon framework. In that model, time was discretized while spatial quantities (e.g., the 2D velocity and location of drones) were treated as continuous variables. The author suggested as future work an alternative approach for addressing the multiperiod dynamic vehicle routing problem, i.e., "in each optimization period, dynamically construct a new graph on the Euclidean plane with nodes being the exact coordinates of relevant entities (i.e., orders and vehicles) at the time and arcs representing the straight-line distances between pairs of entities". The current work implements that idea. Poikonen and Campbell (2020) in their "future directions" article called for new business models of drone delivery that are potentially more fruitful than the widely studied warehouse-to-customer model. The point-to-point 3rd-party logistics (3PL) model proposed herein to some extent answers the call.

As with many other research works in the emerging field, we are simultaneously innovating a new business model and developing a mathematical model to characterize it, thus there is not a clear-cut lineage for the present work. Being an application of the mixed integer programming technique to a specific business case, our model contains features from several well-known problem categories. We highlight three points here. (1) Each shipment request comes in the form of an origin–destination segment that the assigned drone needs to follow, drawing resemblance to the arc routing problem (ARP), see Corberán et al. (2020) for a latest survey and Campbell et al. (2018) for an example of ARP with drones. However, in our model we do not need to treat the origin and destination of a shipment request as separate nodes in the network. Instead, an arc-like request can be collapsed into a single node via data preprocessing, which simplifies the mathematical program. A similar treatment can be found in Riera-Ledesma and Salazar-González (2017) with a more elaborate explanation. (2) As with any other service system that requires dynamic matching between limited supply and random demand (e.g., the taxi market), there is no guarantee that all demands can be satisfied within the desired time frame. Some rules (or modeling assumptions) are needed to handle demand overflow. To this end, we propose a demand bidding mechanism in which each customer makes an offer for her shipment request, and it is up for the supply side, i.e., the fleet manager, to decide which requests to fulfill via solving a profit maximization problem. The fleet routing aspect of the problem therefore resembles a team orienteering problem (TOP), a VRP variant that is concerned with routing a team of vehicles in a network to collect the maximum amount of rewards from demand nodes, see Golden et al. (1987), Chao et al. (1996), Archetti et al. (2014), Gunawan et al. (2016). But different from a canonical TOP, our problem has the team of vehicles initially located at multiple depots and the ending depot of a vehicle trip is unspecified; furthermore, the objective function in our model is augmented with a bin packing type of cost term to model the battery life cycle cost. (3) Our model treats the costs of battery differently from the literature. Specifically, in addition to capacity constraints, we also explicitly minimize the number of battery charging cycles. This treatment introduces a bin packing objective term and constraints into the problem. The bin packing problem is a classic $\mathcal{NP}$-hard combinatorial problem (Garey and Johnson, 1978), thus its addition to the formulation calls solution approaches that are not readily available in the TOP literature.

Column generation (CG) is a well-studied approach for solving linear programs that contain a large number of variables compared to the number of constraints (Nemhauser, 2012). Such linear programs typically arise as the LP relaxation of certain MIP problems (Barnhart et al., 1998; Lübbecke and Desrosiers, 2005), including the bin packing problem (Vanderbeck, 1999; Aydın et al., 2020) and the path-based formulation for VRP (Desrochers et al., 1992; Lozano et al., 2016). The application of column generation has been limited in drone-related optimization problems. For evidence, Otto et al. (2018) surveyed more than 200 articles (published up to early 2018) on optimization approaches to civil applications of drones, among which only one article, i.e., Mufalli et al. (2012), was identified to have employed the column generation (or branch-and-price) technique. The problem addressed in that paper was assigning reconnaissance sensors to different military drones and simultaneously routing the drones to maximize the mission's intelligence gain. The role of column generation was to decompose the problem by UAVs and provide candidate sensor and route combinations on a per UAV basis. Another related work is Riera-Ledesma and Salazar-González (2017), which used column generation to solve a team orienteering arc routing problem. The CG algorithm started with an initial set of candidate routes restricted to visit $l$ customers, and the authors particularly investigated the effect of $l$ on the algorithmic performance, based on benchmark TOP instances (Archetti et al., 2014). Computational results suggested that a more relaxed (i.e., larger) $l$ was preferable when the instance size was manageable, whereas a smaller $l$ was more suitable for larger instances. Given that the pricing problem is solved many times in the CG algorithm, accelerating the solution of the pricing problem is a critical aspect for improving the performance of the CG algorithm (Costa et al., 2019; Desrosiers J., 2005). In the VRP with Time Window (VRPTW) context, the pricing problem is typically an elementary shortest path problem with resource constraints (ESPPRC), which is NP-hard (Dror, 1994). Well-known exact algorithms include the labeling algorithm (Feillet et al., 2004) and the pulse algorithm (Lozano et al., 2016). Recently, Yu et al. (2021) proposed a new enhancement for the pulse algorithm as well as a random coloring method suitable for parallel implementation. However, the path generation subproblem encountered in our context is different from ESPPRC, thus the aforementioned algorithms are not applicable. Instead, we develop a randomized greedy procedure for solving it approximately to expedite the convergence of the column generation algorithm. Despite the algorithmic enhancements and the care taken in the implementation, the CG-based approaches have been shown to be inferior to the more direct, complete enumeration based approach

in our computational assessment. While this observation is made only in the application scenario considered in this paper, it may suggest that the direct approach (i.e., systematically enumerate all columns up front and then solve the set-partitioning or set-packing MIP model) is worth benchmarking within other work where column generation is focused.

In view of the existing literature, the contribution of the paper consists of (1) a new business model for drone-based shipment service considering customer bidding, battery capacity and battery life cycle costs, (2) two mixed integer programming formulations and three different algorithms under the path-based formulation, along with comprehensive computational comparison of these algorithms, (3) an optimization–simulation framework for assessing continuous operations under the proposed business model, (4) insights gained on the service operation from the simulated scenarios.

## 3. Problem description

We consider the following business model in this paper. A drone delivery company aims to perform within-the-hour shipment of light-weight goods, such as meals, drugs, groceries and other small items from any origin to any destination within its service region. The service region may span a city, a suburb or an industrial park, etc. Customers can initiate shipment requests at any time. A shipment request will consist of an origin (pickup) location, a destination (drop-off) location, the estimated weight of the shipping item and the amount of money the customer would pay for the shipment service. It is assumed that the weight, dimension and other aspects of the shipping item meet certain criteria for airborne carriage. Within a set amount of time, e.g., five minutes, of its initiation, each shipment request will be responded with either acceptance or declination. If a request is accepted, a drone will be dispatched to perform the shipment in the next hour. If a request is declined, then some other form of shipment may be offered to fulfill it which cannot guarantee the within-the-hour shipment; in this case, the customer may choose to re-submit the request with an increased bid value in order to be accepted in the next five-minute dispatch interval.

To support the logistics operation, the company has established $p$ drone depots located across the service region and will operate a homogeneous fleet of $K$ drones in the depot network. The depots provide battery swaps and parking spots for stand-by drones. We assume that there is unlimited parking space for drones as well as an adequate supply of fully-charged batteries at each depot. The $p$ depot locations are chosen so as to maximize coverage of the area to enable safe and efficient fleet operations. The depot location problems are addressed in separate papers, e.g., Liu (2021b, 2022).

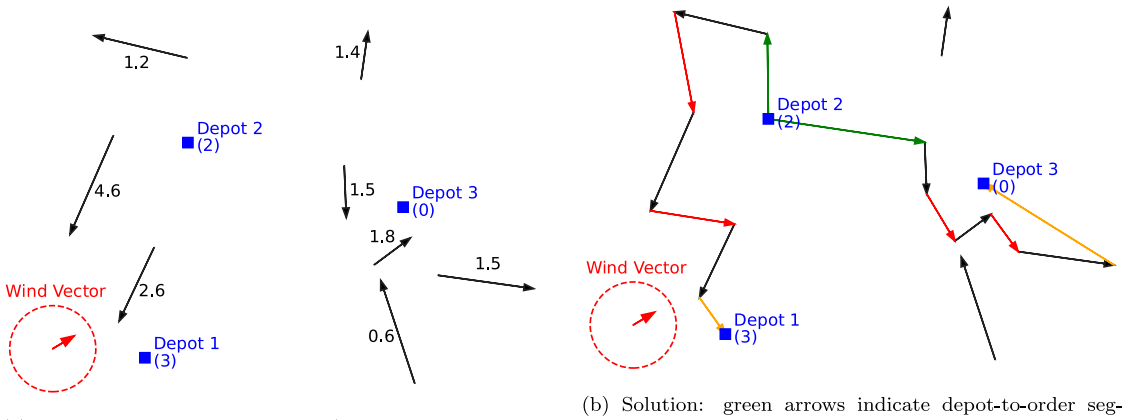
In brief, Liu (2021b) aimed to place a given number of depots in a service region so that the distance between any point in the service region and its nearest depot is minimized; therefore, the battery power reserved for emergency landing would be minimized, meaning that the proportion of a battery's capacity available for value-adding operations would be maximized. In a similar vein, Liu (2022) aimed to place a given number of depots to minimize the maximum distance of an economical delivery trip in which a delivery vehicle would start from the nearest depot to a demand point, visit the demand point and then return to the second nearest depot to that demand point. The assurance that the vehicle was able to return to not only the nearest depot but also the second nearest depot would increase service quality under demand uncertainty. In this paper, we assume that the depot locations have been determined and fixed.

The most critical resource in drone operations is the battery. Most small and medium sized civilian drones are powered by LiPo battery packs. Compared to other types of batteries, the LiPo batteries are able to sustain a high current draw needed by the high-speed electric motors while keeping the voltage relatively stable. However, the amount of energy stored in a battery pack of a given size and weight is quite limited. As a result, most drones are configured with a battery pack that can sustain no more than one hour of airborne time. The operational implication is that drones have to repeatedly return to a depot for battery swaps, i.e., replacing the discharged battery with a fully charged one before embarking on the next flight mission. The swapped-out battery will be charged at the depot and be swapped onto some other drone in need when it is fully charged.

In addition to having limited capacity, each LiPo battery can withstand only a few hundred charge–discharge cycles (French, 2021; Flynt, 2019), making it a consumable component in the system. While the cost of electric energy is negligible compared to the energy cost for other means of transportation, the battery units can cost an appreciable amount of money. For instance, a 32 Ah 12-cell LiPo battery pack costs about $1200 (Adorama, 2021). Suppose the battery is able to sustain 500 charge–discharge cycles in its usable lifetime, then each recharge would account for about $2.4 in the battery's acquisition cost. In operations, it is therefore important to have a battery sustain as much value-adding flight distance as possible before it is recharged. On the other hand, given that a battery was to be recharged, the exact amount of remaining charge in the battery, as long as it is above some minimum level for safety reserve, would have negligible impact on a LiPo battery's life cycle cost. Therefore, the operational cost should be modeled as a function of the *number* of battery recharges, instead of the *amount* of energy consumed or the total distance traveled as in a conventional VRP.

The fleet manager[2] needs to periodically (e.g., once every five minutes) commit available drones to outstanding requests, and plan routes for the committed drones so that they fulfill the assigned requests in the most economical sequence. Specifically, the inputs to the problem consist of (1) the GPS coordinates of the depots as well as the number of stand-by drones with fully-charged battery at each depot; (2) the GPS coordinates of the origin and destination of each shipment request, the weight of the corresponding payload, and the revenue to be earned by performing the requested shipment; (3) the battery capacity and the battery charging cost, which are constant parameters; (4) the wind speed and direction at the time of operation, which will affect battery consumption of different flight segments. We enforce that each time a drone lands at a depot, the battery will be swapped and recharged. This

[2] The *fleet manager* represents the decision-making entity, which can be either a human or a computer program responsible for fleet management.




(a) Input: three depots, five drones (parked at different depots) and eight orders (each represented by a black arrow).

(b) Solution: green arrows indicate depot-to-order segments, red arrows indicate order-to-order segments, orange arrows indicate order-to-depot segments.

**Fig. 1.** Demonstration of the fleet manager's fleet routing problem.

effectively simplifies the ground support operation — just swap the battery for every returning drone. This practice also ameliorates the interdependence between successive dispatch cycles. The problem facing the fleet manager is illustrated in Fig. 1(a), a uniformly-scaled map view of the service region. In the figure, depot locations are marked by blue square markers and the parenthesized number at each depot represents the number of available drones at the depot. Each shipment order is represented by a black arrow, originating from the pickup location and pointing to the dropoff location. The number by each arrow represents the shipping fee (revenue) to be collected from the corresponding customer. The wind vector, i.e., wind speed and direction, is illustrated at the lower right corner of the map. In this example, there are three depots, five drones (three at Depot 1 and two at Depot 2), and eight outstanding shipment orders.

The fleet manager is not obliged to fulfill all shipment requests, nor does she have to utilize all available drones. A drone will be dispatched only when the trip's revenue (i.e., shipping fees collected from all orders fulfilled in the trip) exceeds its cost; if multiple plans are available to serve the requests, the one that maximizes the total profit should be chosen. Therefore, it is legitimate to simultaneously have both unfulfilled requests and unused drones in a dispatch solution. Furthermore, the following assumptions are made: (a) A drone cannot carry more than one shipping item at a time. In other words, once a shipping item is picked up, the drone must fly directly to its drop-off location, before returning to depot or flying to the next pick-up location; (b) A drone cannot park at the drop-off location to wait for order assignment in the next scheduling cycle. Each drone tour must eventually end at a depot; (c) The battery consumption during takeoff and landing can be location dependent and payload dependent, but it must be known or calculable at the time of dispatch. Fig. 1(b) demonstrates the optimal solution of the example shown in Fig. 1(a). In the solution plot, a drone's trip is illustrated by a series of arrows connecting one another. The following color-code is used to distinguish flight segments: a green arrow represents a depot-to-order segment, a red arrow represents an order-to-order segment, and an orange arrow represents an order-to-depot segment. In this example, two drones are dispatched from Depot 2, each serving three orders and returning to different depots. Two orders are left unfulfilled due to the battery constraints and the profit maximization principle in the decision-making process, which will be modeled in detail in the next section.

## 4. Mathematical formulation

### 4.1. Battery consumption model

The energy required for a drone to fly from point A to point B (denoted below by $E_{\overrightarrow{AB}}$) is largely determined by the distance, the wind speed and direction and the payload weight. Zhang et al. (2021) provides an excellent overview of different energy consumption models for delivery drones. Since estimating battery consumption is not the focus of this paper, we make the following assumptions to simplify calculation: the same airspeed is used in all flights; vertical ascend/descend flight stages are ignored; the wind speed and direction is uniform across the service region. Under these assumptions, the energy consumption to fly any trip segment $\overrightarrow{AB}$ on a Euclidean plane can be estimated using formula (1).

$$E_{\overrightarrow{AB}} = W_{\text{Payload}} \cdot \frac{\|\overrightarrow{AB}\|}{(\text{AS}^2 - (\text{WS} \cdot \sin(\angle(\overrightarrow{AB}, \text{WD})))^2)^{1/2} + \text{WS} \cdot \cos(\angle(\overrightarrow{AB}, \text{WD}))} \tag{1}$$

where the function $W_{\text{Payload}}$ represents the power (in Watt) required for transporting a given Payload weight at a constant airspeed (denoted by AS), WS is the wind speed, WD is the unit vector representing the wind direction, and $\|\overrightarrow{AB}\|$ is the Euclidean distance between $A$ and $B$. The effect of wind on energy consumption in this formula is illustrated in Fig. 2. For instance, when the wind speed is small (e.g., 1 m/s) flying in all directions has a similar rate of energy consumption; in contrast, when the wind speed is high (e.g., 5 m/s), flying up wind, in this case southwest, costs more energy than flying down wind.

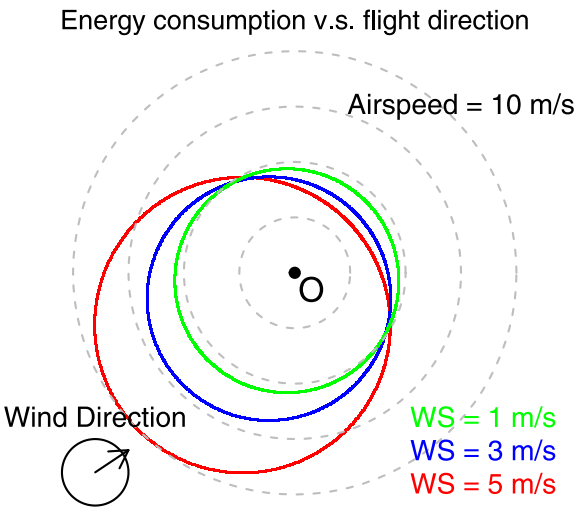



**Fig. 2.** The effects of wind on energy consumption demonstrated in a radial plot. The lengths of the vector pointing from the origin $O$ to a point on the colored contour represents the power consumption for a flight along the vector direction.

Furthermore, we know that each deployed drone must fly a tour that looks like this:

*Depot → Pickup 1 → Dropoff 1 → … → Pickup m → Dropoff m → Depot* in which at least one Pickup and Dropoff pair (i.e., one shipment request) must be performed between visits to the starting and ending depots. Therefore, any drone tour must consist of one Depot-to-Order segment, zero, one or multiple Order-to-Order segment(s), and one Order-to-Depot segment. Let $P_j$ denote the location of depot $j$ and $O_i$ and $D_i$ denote the pickup and dropoff point of the order $i$, the three types of trip segments and their battery consumption calculations are listed below.

| Type | Trip segment | Battery consumption |
|---|---|---|
| Depot-to-Order | $P_j \to O_i \to D_i$ | $c_{ji} := E_{\overrightarrow{P_j O_i}} + E_{\overrightarrow{O_i D_i}}$ |
| Order-to-Order | $D_i \to O_{i'} \to D_{i'}$ | $c_{ii'} := E_{\overrightarrow{D_i O_{i'}}} + E_{\overrightarrow{O_{i'} D_{i'}}}$ |
| Order-to-Depot | $D_i \to P_j$ | $c_{ij} := E_{\overrightarrow{D_i P_j}}$ |

By the time routing optimization is to be performed, the battery consumption to fly each possible trip segment is calculable, thus the relevant $c_{ij}$ values are input data to the optimization problem. In this way, we have transformed a seemingly arc routing problem to a node routing problem.

### 4.2. Drone routing model: a compact formulation

We formulate the fleet manager's problem into a mixed integer program by leveraging a conceptual network. The network nodes are the $p$ depots and the $n$ shipment orders at the time. Each depot node has zero, one or multiple available drone(s) ready for dispatch; each order node is associated with a revenue to be collected; and the arc between each pair of nodes has a travel cost in terms of the battery consumption.

Index sets, parameters and decision variables are defined in Table 1. Note that the integer constants $K$, $p$ and $n$, which represent the number of drones, the number of depots and the number of orders, respectively, are used in defining the index sets in the table.

The fleet manager's drone routing (DR) problem is formulated as follows.

(**DR**):

$$\text{Maximize} \quad \sum_{i\in\mathcal{O}}\sum_{k\in\mathcal{K}} v_i z_{ik} - C\sum_{k\in\mathcal{K}} y_k \tag{2}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{N}}\sum_{j\in\mathcal{N},j\neq i} c_{ij}x_{ijk} \le B y_k, \qquad \forall k\in\mathcal{K} \tag{3}$$

$$\sum_{i\in\mathcal{O}} x_{jik} = w_{jk} y_k, \qquad \forall j\in\mathcal{D}, k\in\mathcal{K} \tag{4}$$

$$\sum_{i\in\mathcal{O}}\sum_{j\in\mathcal{D}} x_{ijk} = y_k, \qquad \forall k\in\mathcal{K} \tag{5}$$

$$\sum_{j\in\mathcal{N},j\neq i} x_{jik} = z_{ik}, \qquad \forall i\in\mathcal{O}, k\in\mathcal{K} \tag{6}$$

**Table 1**
Symbol definition for the drone routing model.

| | |
|---|---|
| Sets: | |
| $\mathcal{K}$ | Index set of drones, $\{1, \ldots, K\}$ |
| $\mathcal{D}$ | Index set of depots, $\{1, \ldots, p\}$ |
| $\mathcal{O}$ | Index set of orders, $\{p+1, \ldots, p+n\}$ |
| $\mathcal{N}$ | Set of network nodes, $\mathcal{N} = \mathcal{D} \cup \mathcal{O}$ |
| Parameters: | |
| $v_i$ | Revenue earned via serving order $i$, $i \in O$ |
| $c_{ij}$ | Energy consumption to go from node $i$ to node $j$, $i, j \in \mathcal{N}$ |
| $w_{jk}$ | $= 1$ if drone $k$ is at depot $j$ in the beginning; $= 0$ otherwise |
| $B$ | Battery's energy capacity per charge |
| $C$ | Cost per battery charge |
| Binary variables: | |
| $x_{ijk}$ | $= 1$ if drone $k$ visits node $i$ then immediately $j$ in its tour |
| $y_k$ | $= 1$ if drone $k$ is used |
| $z_{ik}$ | $= 1$ if order $i$ is served by drone $k$ |
| Positive variables: | |
| $t_i$ | The precedence rank of order $i$'s visitation (if it is served) among all served orders on the same tour |

$$\sum_{j \in \mathcal{N}, j \neq i} x_{ijk} = z_{ik}, \qquad \forall i \in \mathcal{O}, k \in \mathcal{K} \tag{7}$$

$$\sum_{j \in \mathcal{N}, j \neq i} \sum_{k \in \mathcal{K}} x_{ijk} \leq 1, \qquad \forall i \in \mathcal{O} \tag{8}$$

$$t_j - t_i \geq 1 - (n+1)(1 - \sum_{k \in \mathcal{K}} x_{ijk}), \qquad \forall i, j \in \mathcal{O} \tag{9}$$

$$0 \leq t_j \leq n, \qquad \forall j \in \mathcal{O} \tag{10}$$

$$x_{ijk} \in \{0, 1\}, \qquad \forall i, j \in \mathcal{N}, i \neq j, k \in \mathcal{K} \tag{11}$$

$$y_k \in \{0, 1\}, \qquad \forall k \in \mathcal{K} \tag{12}$$

$$z_{ik} \in \{0, 1\}, \qquad \forall i \in \mathcal{O}, k \in \mathcal{K} \tag{13}$$

The objective (2) is to maximize the total profit, which is equal to the total revenue earned from served orders minus the battery cost of the deployed drones. The first term and the second term of the objective function are the objectives for a team orienteering problem and for a bin packing problem, respectively. Therefore, the problem is a novel composition of those two types of problems. The constraints are annotated below.

(3): A drone's travel distance in a tour cannot exceed the battery capacity.
(4): A drone is deployed if and only if its tour contains one depot-to-order link and the drone must be available at the starting depot.
(5): A drone is deployed if and only if its tour contains one order-to-depot link.
(6): A order is served if and only if there is a link into it.
(7): A order is served if and only if there is a link out of it.
(8): No order can have more than one in-link (or out-link, which is implied by (6)–(8)).
(9)–(10): Subtour elimination constraints.
(11)–(13): Variable type constraints.

## 5. Path-based formulation and algorithms

In the solution of DR, each deployed drone will depart the depot where it is originally parked, serve one or multiple orders in sequence, and then land at some depot for a battery recharge. We call the sequence of nodes visited (including the starting and ending depot nodes) by a deployed drone a *path*. Most of the constraints in the DR model, including (3) to (5) and (9) to (10), are used for describing what constitutes a feasible path. Among them, the bin packing constraints (3) and the subtour elimination constraints (9) are the ones that make the above MIP model extremely difficult to solve. For instance, for a case with $n = 20$, $p = 5$, $K = 10$ and randomly generated parameter values, it took CPLEX 46 minutes to solve the model. In contrast, when the constraints (9) are excluded, the solution time is reduced to 6 minutes, and when both (3) and (9) are excluded, the solution time is less than one second. Of course, the latter two models (with constraints dropped) are not valid.

Another way to formulate the problem is to first enumerate all feasible paths, which can be done separately from optimization, and then choose to fly a subset of the paths to maximize the total profit. In this way, all the complicating constraints that ensure path feasibility are offloaded to the preprocessing step; therefore, the optimization problem is greatly simplified.

To pursue this formulation, let $\mathcal{H}$ be a set of feasible paths, $\mathcal{H}^S(j)$ be the set of paths in $\mathcal{H}$ which start from depot $j$, $j \in \mathcal{D}$, and the parameter $a_{hi}$ be equal to 1 if order $i$ is visited in path $h$ and equal to 0 otherwise, for $h \in \mathcal{H}$ and $i \in \mathcal{O}$. Thus, the profit made by having a drone run path $h$ is equal to the sum of revenues earned from all orders visited along the path minus the battery charging cost of the drone running this path, that is,

$$v_h := \sum_{i \in \mathcal{O}} a_{hi} v_i - C. \tag{14}$$

If the fleet manager is confined to selecting drone paths from $\mathcal{H}$, then his profit maximization problem becomes the following Path Selection (PS) problem.

**PS**($\mathcal{H}$):

$$\text{Maximize} \quad \sum_{h \in \mathcal{H}} v_h s_h \tag{15}$$

$$\text{s.t.} \quad \sum_{h \in \mathcal{H}} a_{hi} s_h \le 1, \qquad \forall i \in \mathcal{O} \tag{16}$$

$$\sum_{h \in \mathcal{H}^S(j)} s_h \le w_j, \qquad \forall j \in \mathcal{D} \tag{17}$$

$$s_h \in \{0, 1\}, \qquad \forall h \in \mathcal{H} \tag{18}$$

where the binary variable $s_h$ indicates whether path $h$ is selected in the solution, and the parameter $w_j$ is the number of drones available at depot $j$, that is, $w_j := \sum_{k \in \mathcal{K}} w_{jk}$.

Let $\mathcal{H}^A$ denote the set of all feasible paths, then DR and PS($\mathcal{H}^A$) are equivalent problems. An ostensible challenge in solving PS($\mathcal{H}^A$) is that there are exponentially many elements in $\mathcal{H}^A$, which, on one hand, are non-trivial to enumerate, and on the other hand, make the path selection problem a huge, thus difficult-to-solve, MIP. A typical workaround is to solve PS($\mathcal{H}$) for a smaller path set $\mathcal{H}$, thus to obtain a lower bound for the original problem, and solve the linear programming relaxation of PS($\mathcal{H}^A$) to obtain an upper bound. Let us denote the latter linear program by PSR($\mathcal{H}^A$), which has the same formulation as PS($\mathcal{H}^A$) except that the constraint (18) is replaced by $s_h \ge 0, \forall h \in \mathcal{H}^A$. The column generation approach can be used for solving PSR($\mathcal{H}^A$) without enumerating all elements in $\mathcal{H}^A$. It can be proven that the optimal objective value of PSR($\mathcal{H}^A$) is no greater than that of the LP relaxation of (or the root relaxation in the branch-and-bound algorithm for solving) the DR model (denoted by DRR), forming a better upper bound for the original problem. We omit the proof here since the idea of the proof is straightforward and well known in the literature — in essence, any feasible solution of PSR($\mathcal{H}^A$) can be expressed as a feasible solution to DRR but not vice versa. The set $\mathcal{H}^*$ of paths generated upon the termination of the column generation algorithm is usually much smaller than $\mathcal{H}^A$, thus PS($\mathcal{H}^*$) can be solved quickly to obtain an integer feasible solution.

We explore three approaches for solving PS($\mathcal{H}^A$) and compare their performances via numerical experiments in Section 7. The first approach enumerates all feasible yet undominated paths in $\mathcal{H}^A$ and selects the best paths via solving the PS problem. The second approach solves PSR($\mathcal{H}^A$) using column generation to obtain an upper bound, and solves PS($\mathcal{H}^*$) to obtain a lower bound. The third approach attempts to expedite the column generation algorithm by substituting a heuristic (hence faster) procedure for the MIP subproblem in the column generation algorithm. We believe that these approaches are representative of the typical lines of attack one would employ when facing a set-covering, set-packing or set-partitioning type of problem, therefore the algorithm designs and computational insights gained herein will contribute new knowledge in a broader context.

### 5.1. Path enumeration algorithm

The basic approach for enumerating all feasible paths is through breadth-first search (BFS), see, e.g., Chapter 22 of Cormen et al. (2009). Starting from a depot, a partial path is formed by adding one of the order nodes reachable from the depot. From any partial path, a new partial path can be formed by appending one of the unvisited yet reachable (from the last node in the current partial path) order nodes. Each partial path can also be completed into a full path by appending the nearest depot to the last order node. Here, "reachable" means that the total energy needed upon completing service for an order does not exceed the battery capacity. An order node is appended to a partial path only if, after its visitation, the remaining battery is sufficient for the drone to fly to the nearest depot. Yet-to-explore partial paths are held in a queue, and the enumeration process is terminated when the queue becomes empty.

Factors contributing to the size of the path pool include $B$, $n$ and $p$. Generally speaking, $p$ determines the number of search trees to explore, and $B$ and $n$ affect the depth and width, respectively, of each search tree. Among these factors, $B$ has the most substantial effect. A greater value of $B$ will give rise to longer paths, and overall, more feasible paths. When a partial path covers three or more order nodes, it may dominate, or be dominated by, other partial paths that (1) start from the same depot, (2) cover the same set of order nodes, and (3) visit the same order node in the last step. In this case, the dominated partial paths can be pruned from further exploration. To see this, consider the following two partial paths, for example.

Partial Path 1: Depot 1 → Order 1 → Order 2 → Order 3

Partial Path 2: Depot 1 → Order 2 → Order 1 → Order 3

They have the same revenue but different flight lengths (due to different node visitation sequence). Suppose that Partial Path 1 is shorter than Partial Path 2, then any feasible path derived from Partial Path 2 must have a shorter counterpart of the same revenue derived from Partial Path 1. For the PS problem, paths having the same starting depot and covering the same set of orders are indistinguishable, and only one is needed in the set $\mathcal{H}$ for selection. Therefore, Partial Path 2 can be pruned from the search tree because all possible paths derived from it will be dominated by some corresponding paths derived from Partial Path 1. Such pruning opportunities must be exploited methodically in the breadth-first search process.

**Algorithm 1** Find all feasible paths starting from a given depot $j^*$

1: **function** FindAllPaths($v_i, d_i, J_i, i \in \mathcal{O}$; $c_{ij}, i, j \in \mathcal{N}$; $B$; $C$; $j^*$)
2: Initialize P, PP, PPL, PPV to be empty lists, $l \leftarrow 0$
3: **for** $i \in \mathcal{O}$ **do**
4: **if** $c_{j^*i} + d_i > B$ **then**
5: Continue
6: PP.ap($[j^*, i]$), PPL.ap($c_{j^*i}$), PPV.ap($v_i - C$), $l \leftarrow l + 1$
7: **while** $l > 0$ **do**
8: $\text{PP}', \text{PPL}', \text{PPV}' \leftarrow$ empty lists, $l' \leftarrow 0$
9: SS, SPP $\leftarrow$ empty sets; SPPL, SPPP $\leftarrow$ empty dictionaries
10: **for** $k \in \{1, \ldots, l\}$ **do**
11: $\mathcal{O}' \leftarrow \mathcal{O} \setminus \text{PP}_k$, $i \leftarrow \text{PP}_k[-1]$
12: **for** $i' \in \mathcal{O}'$ **do**
13: **if** $\text{PPL}_k + c_{ii'} + q_{i'} > B$ **then**
14: Continue
15: cPP $\leftarrow \text{PP}_k + [i']$, cPPL $\leftarrow \text{PPL}_k + c_{ii'}$, cPPV $\leftarrow \text{PPV}_k + v_{i'}$, sPP $\leftarrow S$(cPP)
16: **if** sPP $\notin$ SPP **then**
17: $l' \leftarrow l' + 1$, SPP.add(sPP), SPPL.add(sPP:cPPL), SPPP.add(sPP:$l'$)
18: $\text{PP}'$.ap(cPP), $\text{PPL}'$.ap(cPPL), $\text{PPV}'$.ap(cPPV)
19: **else**
20: **if** cPPL < SPPL(sPP) **then**
21: SPPL(sPP) $\leftarrow$ cPPL, $l^* \leftarrow$ SPPP(sPP)
22: $\text{PP}'_{l^*} \leftarrow$ cPP, $\text{PPL}'_{l^*} \leftarrow$ cPPL, $\text{PPV}'_{l^*} \leftarrow$ cPPV
23: **if** cPPV > 0 **then**
24: $\text{sPP}' \leftarrow S'$(cPP)
25: **if** $\text{sPP}' \notin$ SS **then**
26: SS.add($\text{sPP}'$), $i \leftarrow$ cPP[−1], P.ap(cPP + $[J_i]$)
27: PP $\leftarrow \text{PP}'$, PPL $\leftarrow \text{PPL}'$, PPV $\leftarrow \text{PPV}'$
28: **return** P

Algorithm 1 lists the pseudo code for enumerating all feasible and uniquely distinguishable paths that start from a given depot. The inputs include the revenue $v_i$, the distance to the nearest depot $d_i$ and the index of the nearest depot $J_i$ of order $i$, for all $i \in \mathcal{O}$, the energy consumption $c_{ij}$, for $i, j \in \mathcal{N}$, the battery capacity $B$, the battery charging cost $C$ and the starting depot index $j^*$. The output is a list P containing all identified paths.

The pseudo code uses Python-like data structures such as lists, sets and dictionaries, with the following shorthand notations: A.ap(a) appends a new element a to the end of list A; $\text{A}_k$ returns the $k$th element of list A; A[-1] returns the last element of list A; [a] denotes a list of a single element a; [a,b] denotes a list of two elements a and b; A + B concatenates lists A and B; A.add(a) adds a new element a to set A (if a already exists in A, then nothing happens); A.add(key:value) adds the key:value entry to dictionary A; A(key) returns the value corresponding to the key in dictionary A.

Two special functions $S$ and $S'$ are invoked in line 15 and 24, respectively. The function $S$(A) adds $n + p$ to the last element of list A, and returns a set consisting of elements of the resulting list. The special treatment for the last element is to distinguish the last visited order node from other orders. The function $S'$(A) returns a set consisting of elements of list A. The flow of the algorithm is annotated as follows. Lines 3 to 6 enumerate all feasible partial paths that start from the given depot $j^*$ and visit an order node, and add the node index sequence, length and value of the paths into the corresponding lists. Lines 7 to 27 process the partial paths in the list PP until the list is empty. In the process, each partial path in PP is (1) further branched into a number of new partial paths by appending the next order node to visit, which are temporarily stored in the list $\text{PP}'$, and (2) converted into a full path (and added to P) by appending the depot closest to the last visited order node. After each round, the list $\text{PP}'$ containing the next-level partial paths will become the new PP (in line 27), ready for the next round of processing.

In computation, many instances of Algorithm 1 can be run in parallel with each instance handling a different starting depot. The path lists obtained from different runs can then be combined into one set of paths to serve as input for the PS problem.

### 5.2. The column generation approach

Define $\lambda_i$ and $\mu_j$ to be the dual variables[3] associated with constraints (16) and (17), respectively, for $i \in \mathcal{O}$ and $j \in \mathcal{D}$, then the dual problem of PSR($\mathcal{H}$), denoted by PSR-D($\mathcal{H}$), is written as follows.

**PSR-D**($\mathcal{H}$):

$$\text{Minimize} \quad \sum_{i\in\mathcal{O}} \lambda_i + \sum_{j\in\mathcal{D}} w_j \mu_j \tag{19}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{O}} a_{hi}\lambda_i + \mu_{j(h)} \geq v_h, \qquad \forall h \in \mathcal{H} \tag{20}$$

$$\lambda_i \geq 0, \qquad \forall i \in \mathcal{O} \tag{21}$$

$$\mu_j \geq 0, \qquad \forall j \in \mathcal{D} \tag{22}$$

where $j(h)$ denotes the index of the depot where path $h$ starts.

Suppose that for a given set $\mathcal{H}$, we have solved the linear program PSR($\mathcal{H}$) (or equivalently PSR-D($\mathcal{H}$)), and let $\lambda_i^*$ and $\mu_j^*$ denote the optimal dual variable values, for $i \in \mathcal{O}$ and $j \in \mathcal{D}$, respectively, and let $v^*$ denote the optimal objective value. If we could find a new path $h'$ which violates the inequality (20) at the current dual solution, that is, if $h'$ satisfies

$$\sum_{i\in\mathcal{O}} a_{h'i}\lambda_i^* + \mu_{j(h')}^* < v_{h'}, \tag{23}$$

then adding $h'$ to $\mathcal{H}$ and re-solving PSR-D (or PSR) would result in an optimal value greater than $v^*$, yielding an improved solution. In order to find such a new path, we can solve the problem

$$\text{Maximize} \quad v_h - \sum_{i\in\mathcal{O}} a_{hi}\lambda_i^* - \mu_{j(h)}^* \tag{24}$$

$$\text{s.t.} \quad h \text{ is a feasible path.} \tag{25}$$

If the optimal value is greater than zero, we have found a new path; otherwise, we can conclude that no path in $\mathcal{H}^A \setminus \mathcal{H}$ can further improve the solution to PSR($\mathcal{H}$), and hence PSR($\mathcal{H}^A$) is solved.

To instantiate the path generation (PG) problem, let us define the binary variables

$$\begin{aligned} r_i &= 1 \text{ if order } i \text{ is in the new path}, \ i \in \mathcal{O} \\ u_{ij} &= 1 \text{ if the node sequence } i \to j \text{ is in the new path}, \ i, j \in \mathcal{N} \\ q_j &= 1 \text{ if the new path starts from depot } j, \ j \in \mathcal{D} \end{aligned}$$

Then the problem is formulated into the following MIP with respect to these variables.

**PG**($\lambda^*, \mu^*$):

$$\text{Maximize} \quad \sum_{i\in\mathcal{O}} (v_i - \lambda_i^*) r_i - \sum_{j\in\mathcal{D}} \mu_j^* q_j - C \tag{26}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{D}} q_j = 1, \tag{27}$$

$$\sum_{i\in\mathcal{O}} \sum_{i'\in\mathcal{O}, i'\neq i} c_{ii'} u_{ii'} + \sum_{i\in\mathcal{O}} \sum_{j\in\mathcal{D}} (c_{ij}u_{ij} + c_{ji}u_{ji}) \leq B, \tag{28}$$

$$\sum_{j\in\mathcal{N}, j\neq i} u_{ij} = r_i, \qquad \forall i \in \mathcal{O} \tag{29}$$

$$\sum_{j\in\mathcal{N}, j\neq i} u_{ji} = r_i, \qquad \forall i \in \mathcal{O} \tag{30}$$

$$\sum_{i\in\mathcal{O}} u_{ji} = q_j, \qquad \forall j \in \mathcal{D} \tag{31}$$

$$t_j - t_i \geq 1 - (n+1)(1 - u_{ij}), \qquad \forall i, j \in \mathcal{O}, i \neq j \tag{32}$$

$$r_i \in \{0, 1\}, \qquad \forall i \in \mathcal{O} \tag{33}$$

$$u_{ij} \in \{0, 1\}, \qquad \forall i, j \in \mathcal{N} \tag{34}$$

$$q_j \in \{0, 1\}, \qquad \forall j \in \mathcal{D} \tag{35}$$

$$t_i \geq 0, \qquad \forall i \in \mathcal{O} \tag{36}$$

The objective function (26) implements (24) by substituting the definition of $v_h$ in (14). Note that the variable $r_i$ is the counterpart of $a_{hi}$ in (24). Constraints (27) to (36) describe the feasibility criteria for the new path. Specifically,

[3] For the concept of linear programming duality, readers may consult Chapter 7 of Rardin (1998).

(27): The path must have a starting depot.
(28): The battery consumption along the path must not exceed the battery capacity.
(29): The path must contain an out-link from any served order.
(30): The path must contain an in-link to any served order.
(31): The link out of the starting depot must point to an order.
(32): The node visitation order in each link must be observed.
(33)–(36): Variable type constraints.

The column generation algorithm for solving PSR($\mathcal{H}^A$) is given in Algorithm 2.

**Algorithm 2** Column Generation Algorithm

1: Initialize: $\lambda \leftarrow \boldsymbol{v}$, $\boldsymbol{\mu} \leftarrow \mathbf{0}$, $\mathcal{H} \leftarrow \emptyset$
2: Solve PG($\lambda, \boldsymbol{\mu}$) to obtain optimal solution $\boldsymbol{r}^*$ and $\boldsymbol{q}^*$, and optimal objective value $\omega^*$.
3: **while** $\omega^* > 0$ **do**
4: Create a new path index $h' \leftarrow |\mathcal{H}| + 1$
5: Create a new parameter vector $\{a_{h'i}\}_{i\in\mathcal{O}}$ with $a_{h'i} \leftarrow r_i^*$ for each $i \in \mathcal{O}$
6: Create a new parameter $v_{h'} \leftarrow \sum_{i\in\mathcal{O}} v_i r_i^* - C$
7: Set $j(h') \leftarrow \max\{j : q_j^* = 1, j \in D\}$, $\mathcal{H} \leftarrow \mathcal{H} \cup \{h'\}$
8: Solve PSR-D($\mathcal{H}$) to obtain optimal solution $\lambda^*$, $\boldsymbol{\mu}^*$, and optimal objective value $\nu^*$.
9: Update $\lambda \leftarrow \lambda^*$, $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu}^*$
10: Solve PG($\lambda, \boldsymbol{\mu}$) to obtain optimal solution $\boldsymbol{r}^*$ and $\boldsymbol{q}^*$, and optimal objective value $\omega^*$.
11: Solve PS($\mathcal{H}$) to obtain optimal solution $\boldsymbol{s}^*$ and optimal objective value $\phi^*$.
12: **return** Selected paths $\{h \in \mathcal{H} : s_h^* = 1\}$, lower bound $\phi^*$ and upper bound $\nu^*$.

Upon the termination of the algorithm, the best-found dispatch plan is to run each path in the set $\{h \in \mathcal{H} : s_h^* = 1\}$ by a drone. The quality of this dispatch plan is quantified by its total profit $\phi^*$ and the upper bound $\nu^*$ on the profit. The selected paths in the PS($\mathcal{H}$) solution do not contain information about the visitation sequence of orders and the index of the ending depot. To retain this information, we can simply record, for each candidate path when it is generated (in lines 2 and 10), the $(i, j)$ pairs, $i, j \in \mathcal{N}$, for which the optimal solution value $u_{ij}^*$ in PG($\lambda, \boldsymbol{\mu}$) is equal to 1. Alternatively, the path details, i.e., the order visitation sequence and ending depot, can be re-optimized by a secondary optimization routine which, for example, attempts to minimize the battery consumption of the path. Heuristic approaches can be used as well. For instance, when the number of orders visited in a path is small, complete enumeration can be used to find the optimal sequence; otherwise, a greedy heuristic may be employed. We do not pursue such adjustments in this work, assuming that the amount of residual charge in the battery after a tour is unimportant.

### 5.3. A heuristic enhancement to the column generation algorithm

For large instances, the looping process (lines 3 to 10) in the CG algorithm can be time-consuming. The process can be expedited in two ways: generate the new paths faster, and reduce the number of iterations needed to meet the convergence criterion. In fact, there is no need to solve the path generation problem exactly, except to prove the optimality of the current solution in the last CG iteration, and thus fast heuristics can be used in its place. Costa et al. (2019) reviewed a number of pricing heuristics in the VRP context, ranging from those that tapped into the branch-cut-and-price process for solving the master problem (Pecin et al., 2017) to meta-heuristics such as the Tabu search algorithms (Desaulniers et al., 2008). Most recently, Yu et al. (2021) proposed an enhancement to the pulse algorithm (Lozano et al., 2016) and a random coloring algorithm to improve the pricing efficiency. We note a critical difference between the ESPPRC subproblem in the VRP context and the PG subproblem in our context. Specifically, the capacity constraint (which establishes an upper limit to the sum of the demands collected along the path) in ESPPRC is agnostic of the node visitation sequence, which enables various bounding and pruning strategies in the design of exact algorithms. In contrast, the battery capacity constraint (28) in the PG model does not allow for such a convenience, upholding the strong combinatorial nature of the problem. The time window constraints that are widely considered in ESPPRC also provide additional opportunities for search space reduction (Irnich and Desaulniers, 2006; Costa et al., 2019), whereas such constraints are not present in the PG problem.

We propose a randomized greedy search heuristic intended to quickly find a set of paths, each of which to satisfy (23), so that these paths can be added to the master set $\mathcal{H}$ in lieu of a single "optimal" new path that would have to be found by solving the PG problem. The algorithm is outlined in Algorithm 3. The idea is to make $\kappa$ attempts, each aiming at constructing a maximal-length feasible path that is also improving (i.e., having a positive value in terms of the objective function (24)). The parameter $\kappa$ controls the amount of computational effort allocated to this process. The number of improving paths found by this process may vary, but so long as one such path is found, the more time-consuming process of solving the PG model can be skipped. On the other hand, if Algorithm 3 returned an empty set P, then the PG model would have to be solved, to either find a new path or verify that no improving path could be found (thus terminate the CG process). A faster variant of Algorithm 3 would be a search process that stops as soon as the first satisfactory path is identified. The benefit for finding and keeping multiple (up to $\kappa$) improving paths comes from the fact that more paths (i.e., a larger $\mathcal{H}$) would enable PSR($\mathcal{H}$) to approach PSR($\mathcal{H}^A$) faster in each iteration, thereby reducing the iterations needed for convergence. Moreover, a larger $\mathcal{H}$ upon the termination of the CG process would enable the final PS($\mathcal{H}$) to find a better feasible solution.

**Algorithm 3** Randomized greedy algorithm for finding multiple new paths

**function** FindImprovingPaths($\lambda_i, d_i, J_i, i \in \mathcal{O}$; $\mu_j, j \in \mathcal{D}$; $c_{ij}, i, j \in \mathcal{N}$; $B$; $C$; $\kappa$)
  Initialize: P ← empty list, $k \leftarrow 0$
  **while** $k < \kappa$ **do**
    cP ← empty list, $D \leftarrow 0$, $V \leftarrow -C$, Done ← 0, $k \leftarrow k + 1$
    **while** Done == 0 **do**
      **if** cP is empty **then**
        Randomly pick a $j$ from $\mathcal{D}$, cP.ap($j$), $V \leftarrow -\mu_j$
        $\mathcal{O}' \leftarrow \{i \in \mathcal{O} : D + c_{ji} + d_i \le B\}$, Done ← −1
        **if** $|\mathcal{O}'| > 0$ **then**
          Randomly pick an $i$ from $\mathcal{O}'$
          cP.ap($i$), $D \leftarrow c_{ji}, V \leftarrow V + \lambda_i$, Done ← 0
      **else**
        $j \leftarrow$ cP[−1], $\mathcal{O}' \leftarrow \{i \in \mathcal{O} \setminus \text{cP} : D + c_{ji} + d_i \le B\}$, Done ← −1
        **if** $|\mathcal{O}'| > 0$ **then**
          Randomly pick an $i$ from $\mathcal{O}'$
          cP.ap($i$), $D \leftarrow D + c_{ji}, V \leftarrow V + \lambda_i$, Done ← 0
        **else**
          **if** $V > 0$ **then**
            $i \leftarrow$ cP[−1], $j \leftarrow J_i$, cP.ap($j$), Done ← 1
    **if** Done == 1 **then**
      P.ap(cP)
  Return P

## 6. Optimization–simulation approach for studying multi-period operations

In this section, we discuss a number of potential challenges in extending the model to multiple periods, and then propose a simulation approach to analyzing the continuous business operations that span a long period of time. A case study for the business model described in Section 3 will be presented in Section 7.3.

Mathematically, a simple extension of the DR model to multiple periods is to add a time index to all decision variables, along with necessary constraints that link the system states between successive periods. An example of such linking constraints is that the ending depot of a drone's tour in period $t$ must be the depot at which the drone is available in period $t+1$. However, a multi-period model constructed in this way would not be practical, because it takes varying amount of time for drones to run the delivery routes and in real operations one cannot wait until all drones have landed to start the next dispatch cycle. Another relevant literature model is VRP with time windows (VRPTW), which tracks the vehicles' moving time endogenously. VRPTW is not applicable in our context either, since the customer orders are not known in advance. In fact, it is common practice in the ride-hailing service (another form of the on-demand shipment service discussed here) to adopt a short dispatch interval to quickly match the demand and supply available at the moment, see, e.g., Zhang et al. (2017), Riley et al. (2019), Liu (2021a). We propose to adopt the same method in real operations.

In order to evaluate the long-term system performance (e.g., service level, profitability, etc.) under the hypothesized setting, we can use simulation methods. Fortunately, the ability to quickly solve realistic instances of the fleet routing problem by the proposed approaches enables us to incorporate the optimization model in a high-frequency periodical decision-making process to support both real-world operations and the simulation study. Moreover, in a simulation model, we have abundant flexibility to model the time it takes to perform different activities (mainly, the flight time). For instance, the flight time required for a trip segment may depend on the trip distance, the drone's airspeed and the wind vector (speed and direction). In addition, random variations can be introduced as well.

Once time is properly modeled, we will see that the dispatch decision made in a period not only partly determines the problem faced by the succeeding period, but also affects the resource availability several periods into the future. This is a realistic dynamism to be expected in real-world operations. It would be tremendously more complex to model such a dynamism together with demand uncertainties in a global optimization framework. We believe that the proposed division of work between optimization modeling and simulation modeling offers a practical perspective for analyzing complex business models. The optimization–simulation framework can help entrepreneurs who are interested in running such a business answer a series of system design questions and perform all kinds of what-if analyses. When the real-world operations are up and running, the optimization model and its solution methods, which have gone through thorough validations via simulation, can be directly implemented in practice.

## 7. Computational and simulation assessments

The proposed algorithms and simulation framework have been implemented into computer programs. In this section, we perform computational experiments to validate the models, compare the performances and present a simulation case study using realistic

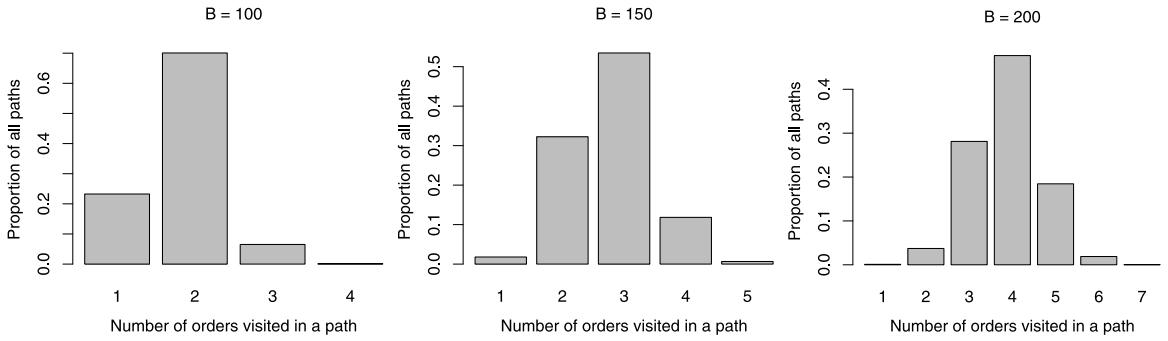



**Fig. 3.** The distribution of path length under different battery capacities.

parameter settings. The hardware platform is a Dell Precision Tower 8520 with an Intel(R) Core(TM) i9-9900X CPU @ 3.50 GHz, 64 GB RAM running Windows 10 Enterprise Operating System. All MIP and LP problems are solved using CPLEX 20.1.0.1 (with default solver options) via GAMS 37.1.0. All algorithms as well as the simulation framework are written in Python 3.9 with the use of the GAMS Python API. The LP and MIP models in the column generation algorithm have been implemented by way of the GAMS Model Instance, which allows the model rims[4] to be constructed once and the instance data to be updated in-between repeated solves. In this way, the model construction time is minimized and the successive LP solves can benefit from a warm-start basis. The source code will be available on the author's Github page.

### 7.1. Comparison between different solution approaches

We create various test instances on which to compare the computational efficiency and solution quality of different approaches for solving the drone routing problem. The service area for these test instances is a 100 x 100 square area.[5] In this area for a fleet of $K = 20$ drones, we generated test instances with the number of depots $p \in [5, 10]$, battery capacity $B \in [50, 100, 150, 200]$ and the number of orders $n \in [10, 20, 30, 40, 50]$, one instance for each combination of $(p, B, n)$ values. For all instances, the orders are randomly generated to have a shipment distance between 10 and 30, a revenue value between \$0 and \$5 and zero payload weight (basically, all trip segments have the same power consumption rate per distance traveled). The battery charging cost is set to \$2. For simplicity, wind effect was not considered while calculating the $c_{ij}$ values for these instances.

The value range for the battery capacity (i.e., 50 to 200) is selected so that the test instances represent the reality that each battery charge can at most sustain a handful of shipment assignments. The distributions of the path length in terms of the number of orders visited under different battery capacity settings are illustrated in Fig. 3. The paths summarized in this figure are generated by Algorithm 1 at $(p, K, n) = (10, 20, 50)$. In reality, it would be extremely rare for a drone to fly more than five shipment assignments in a row before a battery recharge, so we believe that the $B$ value of 100 or 150 is practically most relevant.

Each instance is solved using four methods: (1) Solve the DR model directly using CPLEX; (2) Enumerate all feasible paths (i.e., $\mathcal{H}^A$) using Algorithm 1 and then solve the PS($\mathcal{H}^A$) using CPLEX; (3) Apply the column generation (CG) Algorithm 2; and (4) Apply CG along with the heuristic enhancement of Algorithm 3, denoted as CGH. The parameter $\kappa$ was set to 2000[6] in all invocations of Algorithm 3. A 600-second time limit was enforced for each run. For each of those runs that did not terminate naturally by the time limit, the best solution upon the forced termination point was reported. For each instance when Algorithm 1 was executed, $p$ parallel processes, each using a CPU core, were actually invoked. Each process handled the enumeration of paths starting from a given depot. The path lists from different processes were combined into one (i.e., the set $\mathcal{H}^A$) upon the termination of all processes, before the PS($\mathcal{H}^A$) solution process started. All invocations of the CPLEX solver were by default single-threaded.

The experiment results are listed in Table 2. For each method, the total computing time (CPU) in seconds and the optimality gap are reported. For CG and CGH methods, the number of subproblem MIP invocations (nMIP) and the number of candidate paths in the path pool upon the final PS solve are reported. For the PS method, the total number of unique undominated paths found by Algorithm 1 is reported under column $|\mathcal{H}^A|$. The last three columns under the header Clairvoyant, i.e., UB, Opt and Gap, report the optimal value of PSR($\mathcal{H}^A$) found by CG (which is also equal to that found by CGH, and equal to that found by the root relaxation in the solution of PS($\mathcal{H}^A$)), the optimal value of PS($\mathcal{H}^A$) found by PS and the relative gap between the two, i.e., (UB - Obj)/Obj, respectively. These values help reveal how far from the best possible the CG and CGH solutions are.

[4] The term *model rim* is coined by GAMS to mean the data structure instance that holds variables and equations for a problem, see GAMS (2022).

[5] The unit of length measurement is omitted intentionally here, which can be flexibly instantiated to represent an area of any size. Other distance-related parameters, such as the orders' pickup and dropoff locations and the drones' flight range (battery capacity), are set in coherence with the area size.

[6] The selection of this value was based on trials of different values between 500 to 5000 on a large instance.

**Table 2**
Comparison of different solution approaches.

| $p$ | $B$ | $n$ | DR | | PS | | | CG | | | | CGH | | | | Clairvoyant | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | CPU | Gap | CPU | Gap | $\lvert\mathcal{H}^A\rvert$ | CPU | Gap | nMIP | $\lvert\mathcal{H}\rvert$ | CPU | Gap | nMIP | $\lvert\mathcal{H}^A\rvert$ | UB | Obj | Gap |
| 5 | 50 | 10 | 0.4 | 0.000 | 0.8 | 0 | 1 | 2.7 | 0.000 | 2 | 1 | 2.6 | 0.000 | 1 | 1 | 2.75 | 2.75 | 0 |
| | | 20 | 0.3 | 0.000 | 0.8 | 0 | 5 | 3.4 | 0.000 | 5 | 4 | 3.0 | 0.000 | 1 | 4 | 5.88 | 5.88 | 0 |
| | | 30 | 0.5 | 0.000 | 0.8 | 0 | 8 | 3.8 | 0.000 | 7 | 6 | 3.1 | 0.000 | 1 | 6 | 5.45 | 5.45 | 0 |
| | | 40 | 0.7 | 0.000 | 0.9 | 0 | 5 | 3.8 | 0.000 | 4 | 3 | 3.0 | 0.000 | 1 | 3 | 2.48 | 2.48 | 0 |
| | | 50 | 1.0 | 0.000 | 0.8 | 0 | 6 | 4.2 | 0.000 | 7 | 6 | 3.7 | 0.000 | 1 | 6 | 9.84 | 9.84 | 0 |
| | 100 | 10 | 0.3 | 0.000 | 0.8 | 0 | 16 | 3.7 | 0.000 | 10 | 9 | 3.1 | 0.000 | 1 | 13 | 13.85 | 13.85 | 0 |
| | | 20 | 519.4 | 0.000 | 0.8 | 0 | 48 | 4.7 | 0.000 | 17 | 16 | 3.5 | 0.000 | 1 | 31 | 23.39 | 23.39 | 0 |
| | | 30 | 600.0 | 0.148 | 0.9 | 0 | 114 | 8.1 | 0.031 | 25 | 24 | 3.4 | 0.000 | 1 | 70 | 36.95 | 36.95 | 0 |
| | | 40 | 600.0 | 0.262 | 0.9 | 0 | 161 | 18.9 | 0.031 | 44 | 43 | 4.5 | 0.016 | 1 | 88 | 45.05 | 44.60 | 0.010 |
| | | 50 | 600.0 | 0.187 | 0.9 | 0 | 219 | 29.5 | 0.005 | 51 | 50 | 4.9 | 0.005 | 1 | 129 | 62.77 | 62.44 | 0.005 |
| | 150 | 10 | 27.1 | 0.000 | 0.8 | 0 | 120 | 4.0 | 0.000 | 11 | 10 | 3.9 | 0.000 | 1 | 46 | 16.95 | 16.95 | 0 |
| | | 20 | 600.0 | 0.061 | 1.0 | 0 | 649 | 8.9 | 0.021 | 33 | 32 | 8.6 | 0.015 | 9 | 200 | 33.38 | 32.90 | 0.015 |
| | | 30 | 600.0 | 0.078 | 1.1 | 0 | 2084 | 21.8 | 0.056 | 52 | 51 | 23.3 | 0.013 | 14 | 356 | 50.41 | 49.88 | 0.011 |
| | | 40 | 600.0 | 0.073 | 2.2 | 0 | 5533 | 86.6 | 0.016 | 77 | 76 | 63.2 | 0.011 | 20 | 515 | 65.43 | 64.91 | 0.008 |
| | | 50 | 600.0 | 0.069 | 2.0 | 0 | 5927 | 123.6 | 0.013 | 97 | 96 | 95.7 | 0.002 | 36 | 556 | 85.82 | 85.63 | 0.002 |
| | 200 | 10 | 3.3 | 0.000 | 0.9 | 0 | 446 | 4.9 | 0.000 | 19 | 18 | 5.8 | 0.000 | 5 | 125 | 18.95 | 18.95 | 0 |
| | | 20 | 600.0 | 0.038 | 1.6 | 0 | 5132 | 13.2 | 0.025 | 46 | 45 | 19.4 | 0.005 | 15 | 456 | 36.78 | 36.60 | 0.005 |
| | | 30 | 600.0 | 0.045 | 5.3 | 0 | 27725 | 64.9 | 0.043 | 77 | 76 | 105.5 | 0.024 | 48 | 690 | 55.57 | 55.29 | 0.005 |
| | | 40 | 600.0 | 0.051 | 24.8 | 0 | 119042 | 511.6 | 0.052 | 109 | 108 | 540.6 | 0.007 | 73 | 858 | 71.26 | 71.00 | 0.004 |
| | | 50 | 600.0 | 0.052 | 25.8 | 0 | 108598 | 600.0 | 0.046 | 132 | 132 | 600.0 | 0.011 | 84 | 940 | 92.60 | 92.40 | 0.002 |
| 10 | 50 | 10 | 0.3 | 0.000 | 0.9 | 0 | 1 | 3.1 | 0.000 | 2 | 1 | 2.9 | 0.000 | 2 | 1 | 0.00 | 0.00 | 0 |
| | | 20 | 0.4 | 0.000 | 0.9 | 0 | 8 | 3.8 | 0.000 | 8 | 7 | 2.9 | 0.000 | 1 | 7 | 5.80 | 5.80 | 0 |
| | | 30 | 0.9 | 0.000 | 0.9 | 0 | 11 | 3.9 | 0.000 | 7 | 6 | 3.9 | 0.000 | 1 | 6 | 8.02 | 8.02 | 0 |
| | | 40 | 0.9 | 0.000 | 0.9 | 0 | 27 | 5.3 | 0.000 | 14 | 13 | 4.3 | 0.000 | 2 | 16 | 11.47 | 11.47 | 0 |
| | | 50 | 1.2 | 0.000 | 0.9 | 0 | 13 | 4.7 | 0.000 | 9 | 8 | 4.2 | 0.000 | 2 | 8 | 7.36 | 7.36 | 0 |
| | 100 | 10 | 0.4 | 0.000 | 0.9 | 0 | 40 | 4.0 | 0.004 | 12 | 11 | 3.5 | 0.004 | 2 | 14 | 13.56 | 13.51 | 0.004 |
| | | 20 | 600.0 | 0.039 | 1.0 | 0 | 201 | 6.7 | 0.000 | 27 | 26 | 3.7 | 0.000 | 2 | 56 | 35.07 | 35.07 | 0 |
| | | 30 | 600.0 | 0.068 | 1.0 | 0 | 540 | 12.6 | 0.000 | 44 | 43 | 5.2 | 0.000 | 2 | 134 | 50.27 | 50.27 | 0 |
| | | 40 | 600.0 | 0.104 | 1.0 | 0 | 1165 | 22.1 | 0.025 | 52 | 51 | 6.1 | 0.001 | 2 | 203 | 56.18 | 56.14 | 0.001 |
| | | 50 | 600.0 | 0.222 | 0.9 | 0 | 675 | 44.6 | 0.004 | 85 | 84 | 9.1 | 0.000 | 2 | 212 | 71.29 | 71.29 | 0 |
| | 150 | 10 | 600.0 | 0.043 | 0.9 | 0 | 205 | 4.5 | 0.013 | 14 | 13 | 4.1 | 0.013 | 2 | 36 | 15.70 | 15.50 | 0.013 |
| | | 20 | 600.0 | 0.032 | 1.2 | 0 | 1977 | 8.8 | 0.049 | 34 | 33 | 8.4 | 0.012 | 6 | 237 | 43.48 | 42.94 | 0.013 |
| | | 30 | 600.0 | 0.049 | 1.8 | 0 | 7953 | 31.9 | 0.032 | 71 | 70 | 32.8 | 0.014 | 22 | 495 | 62.01 | 61.72 | 0.005 |
| | | 40 | 600.0 | 0.096 | 6.7 | 0 | 30923 | 185.2 | 0.033 | 122 | 121 | 171.5 | 0.011 | 51 | 691 | 76.53 | 76.10 | 0.006 |
| | | 50 | 600.0 | 0.161 | 4.3 | 0 | 15269 | 346.5 | 0.014 | 125 | 124 | 302.4 | 0.004 | 45 | 721 | 97.96 | 97.60 | 0.004 |
| | 200 | 10 | 117.8 | 0.000 | 1.0 | 0 | 775 | 5.8 | 0.004 | 25 | 24 | 5.4 | 0.004 | 3 | 99 | 18.64 | 18.56 | 0.004 |
| | | 20 | 600.0 | 0.023 | 2.4 | 0 | 13023 | 13.9 | 0.009 | 48 | 47 | 30.5 | 0.009 | 25 | 543 | 46.41 | 46.29 | 0.003 |
| | | 30 | 600.0 | 0.024 | 41.7 | 0 | 80851 | 57.0 | 0.029 | 89 | 88 | 109.1 | 0.011 | 56 | 867 | 67.07 | 66.38 | 0.010 |
| | | 40 | 600.0 | 0.059 | 600.0 | 0.006 | 536364 | 228.9 | 0.064 | 146 | 145 | 349.3 | 0.021 | 98 | 1030 | 82.82 | 82.10 | 0.009 |
| | | 50 | 600.0 | 0.064 | 36.5 | 0 | 274248 | 600.0 | 0.037 | 143 | 143 | 600.0 | 0.011 | 63 | 1051 | 106.11 | 106.10 | 0.000 |

The following observations can be made from these computational experiments.

1. The compact formulation (DR) was the most difficult to solve, having only 16 smaller instances solved within the time limit. From the solution process we observed that the root relaxation was of low quality and the branch-and-bound process converged slowly. Therefore, we conclude that the original DR formulation is not computationally viable. The observation that the column generation algorithm is generally more efficient at closing the optimality gap than the compact MIP formulation is not surprising, as it has been made in other similar problems, e.g., Faiz et al. (2019) on a variant of the VRPTW.
2. Both variants of the column generation approach, CG and CGH, were able to solve PSR($\mathcal{H}^A$) and the final PS($\mathcal{H}$) for 38 instances. Comparing CG and CGH, CGH was able to achieve a smaller optimality gap than CG for all cases while consuming less computing time in most cases. The time saving was attributed to the reduced number of MIP solves (nMIP) needed for convergence, and the better solutions were attributed to the larger set of paths available for the final optimized selection. Overall, CGH is an effective enhancement over CG.
3. The path selection (PS) formulation coupled with an exhaustive enumeration of feasible paths turns out to be the most computationally efficient approach for all test instances, outperforming other approaches by a large margin in both computing time and solution exactness. The observation that this simple, seemingly brute-force approach so substantially outperformed the column generation approach is somewhat surprising, as it contradicts the general impression that column generation is design for, thus effective at, mitigating scalability challenges due to "innumerable" columns in a linear program. The outstanding performance of the PS approach can be attributed to both the effective implementation of the path enumeration algorithm (Algorithm 1), its parallel execution and CPLEX solver's ability to solve large-scale set packing type of MIP models sufficiently fast.

**Table 3**
Comparison of solution approaches on larger instances, at $p = 10$, $K = 20$, $B = 150$.

| $n$ | PS | | | | | | GC | | | CGH | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Enum | Solve | Total | Paths | UB | LB | Time | UB | LB | Time | UB | LB |
| 10 | 0.7 | 0.2 | 0.9 | 205 | 15.50 | 15.50 | 6.2 | 15.70 | 15.50 | 7.0 | 15.70 | 15.50 |
| 20 | 0.7 | 0.3 | 1.2 | 1977 | 42.94 | 42.94 | 9.3 | 43.48 | 41.43 | 13.9 | 43.48 | 42.94 |
| 30 | 0.8 | 0.5 | 1.6 | 7953 | 61.72 | 61.72 | 32.0 | 62.01 | 60.11 | 73.3 | 62.01 | 61.44 |
| 40 | 1.2 | 3.4 | 6.0 | 30923 | 76.10 | 76.10 | 181.5 | 76.53 | 74.09 | 240.3 | 76.53 | 75.87 |
| 50 | 0.9 | 2.3 | 4.0 | 15269 | 97.60 | 97.60 | 344.2 | 97.96 | 96.59 | 278.3 | 97.96 | 97.60 |
| 60 | 4.3 | 19.3 | 30.9 | 149902 | 118.49 | 118.49 | 2530.3 | 118.83 | 116.20 | 3515.5 | 118.83 | 118.32 |
| 70 | 5.0 | 23.2 | 35.1 | 138792 | 137.08 | 137.08 | 3457.1 | 137.33 | 134.94 | 3227.8 | 137.33 | 136.59 |
| 80 | 3.2 | 18.2 | 26.2 | 105082 | 158.77 | 158.77 | 3600.0 | 159.38 | 153.86 | 3600.0 | 159.39 | 157.82 |
| 90 | 8.5 | 73.2 | 94.9 | 274912 | 182.50 | 182.50 | 3600.0 | 183.07 | 171.35 | 3600.0 | 183.18 | 180.54 |
| 100 | 20.3 | 136.5 | 180.7 | 474178 | 200.35 | 200.35 | 3600.0 | 198.89 | 182.66 | 3600.0 | 200.38 | 195.86 |
| 110 | 21.8 | 117.8 | 174.5 | 665899 | 232.56 | 232.56 | 3600.0 | 230.87 | 218.67 | 3600.0 | 229.76 | 222.22 |
| 120 | 22.1 | 599.3 | 651.3 | 620006 | 238.67 | 238.67 | 3600.0 | 236.34 | 213.81 | 3600.0 | 237.96 | 232.07 |
| 130 | 32.5 | 182.8 | 247.7 | 673005 | 249.51 | 249.51 | 3600.0 | 245.38 | 211.28 | 3600.0 | 244.63 | 237.25 |
| 140 | 50.0 | 168.7 | 284.6 | 1255179 | 276.35 | 276.35 | 3600.0 | 273.64 | 245.96 | 3600.0 | 272.66 | 264.19 |
| 150 | 79.3 | 543.4 | 727.0 | 2039987 | 291.42 | 291.42 | 3600.0 | 281.32 | 233.24 | 3600.0 | 281.25 | 270.09 |
| 160 | 98.6 | 365.9 | 557.8 | 1808240 | 282.50 | 282.50 | 3600.0 | 270.18 | 229.21 | 3600.0 | 275.30 | 267.02 |
| 170 | 103.6 | 384.3 | 603.6 | 2285992 | 293.37 | 293.37 | 3600.0 | 268.27 | 230.33 | 3600.0 | 282.51 | 273.87 |
| 180 | 130.7 | 369.6 | 618.9 | 2303002 | 291.54 | 291.54 | 3600.0 | 203.76 | 186.65 | 3600.0 | 273.31 | 265.40 |
| 190 | 245.5 | 1479.4 | 1962.4 | 4315542 | 329.41 | 329.41 | 3600.0 | 255.29 | 214.36 | 3600.0 | 306.57 | 294.54 |
| 200 | 348.4 | 759.3 | 1387.3 | 5324890 | 320.43 | 320.43 | 3600.0 | 159.44 | 138.03 | 3600.0 | 299.09 | 286.64 |

To better compare how different approaches scale, we also conducted experiments on a series of instances with increasing numbers of orders $n$, at fixed values of $(p, K, B)$ of $(10, 20, 150)$. The time limit for each of these runs was set to 3600 seconds. Table 3 lists the results. The total time consumption of the PS approach can be decomposed into three parts: (1) the path enumeration time, (2) the time taken by Python and GAMS to prepare data for the (potentially large) MIP instance of the PS($\mathcal{H}^A$) model, and (3) the time taken by CPLEX to solve the model instance. Among these components, (1) and (3) are shown under columns enum and solve, and the total time is under column total (so (2) can be inferred and thus not shown to save space). The upper and lower bounds (which are both equal to the optimal objective value for each case) are listed for the purpose of comparison with the other approaches. We can see that, despite the exponentially growing number of paths involved, the PS approach is able to exactly solve all instances within one hour. In contrast, the CG and CGH approaches were only able to terminate naturally (i.e., complete solving the PSR($\mathcal{H}^A$)) for the smallest seven instances. Even for those seven cases, CG and CGH both left some optimality gap, a phenomenon inherent to the nature of the column generation heuristics. For the larger cases in which CG and GCH algorithms were forced to terminate upon reaching the time limit, the values listed under the UB columns are not valid upper bounds; they are listed here to show relatively how far the forced termination points are to the natural termination points, at which the real UB values must be at least as big as the UB values listed under the PS approach. By comparing, between CG and CGH, the UB and LB values upon the forced termination points, we can also glean the advantage of CGH over CG, particularly for large instances. Compared to CG, the CGH approach is generally able to make more progress toward convergence in a set amount of time, and produce a better feasible solution upon termination. Nonetheless, both approaches have been dwarfed by the more effective PS approach.

All these comparison results suggest decisively that the PS approach, i.e., parallel execution of Algorithm 1 to generate $\mathcal{H}^A$ and then solving PS($\mathcal{H}^A$) using CPLEX, is the most practical approach for data instances of all sizes.

### *7.2. Effects of battery capacity on the fleet routing solution*

Battery capacity is the most important parameter in the problem, which to a large extent determines the feasible space and, hence, the computational complexity of the problem. In this section, we experiment different values for battery capacity on the same test instance, to reveal its effects on the dispatch solution and computing load. We use the input instance of $(p, K, n) = (5, 20, 30)$ generated in the previous section, and set the drone's battery capacity $B$ in the set $[50, 100, 150, 200, 250]$ to create five scenarios. The solutions obtained by the PS approach for these scenarios are listed in Table 4. As seen earlier, a larger battery capacity makes the problem much harder to solve, as exhibited by the monotone increase in computing time (CPU) and the number of total paths to select from as the battery capacity increases from 50 to 250.

It is worth comparing the Drones Dispatched and Orders Served across these scenarios. Among the 30 orders, 17 have a revenue (i.e., $v_i$ in the model) greater than \$2 (i.e., the battery charging cost), indicating that it is profitable to serve each of these orders by a separate drone trip, when permitted by the drone availability and flight range. This is what happens when the battery capacity is small. At $B = 50$, 6 drones are dispatched to serve 6 orders, leaving no room for consolidation due to the limited battery capacity. On the other hand, when the battery capacity is large (i.e., $B = 250$), there is much more room for assignment and routing optimization; in the end, only four drone trips are needed to serve 27 of the 30 orders, and most of the low-value orders which would be uneconomical for any dedicated drone trip are now served, thanks to the optimized consolidation of shipment segments into just a few drone trips. The solutions under $B = 50$ and 250 are visualized in Fig. 4 for contrast. We can see that the solution on the left ($B = 50$ case) looks more like the solution to the problem of matching profitable orders with available drones, while the solution

**Table 4**
Effects of the battery capacity on drone routing solution.

| $B$ | Profit | CPU | Total paths | Drones dispatched | Orders served |
|---|---|---|---|---|---|
| 50 | 5.45 | 0.6 | 8 | 6 | 6 |
| 100 | 36.95 | 0.6 | 114 | 11 | 21 |
| 150 | 49.88 | 0.9 | 2084 | 8 | 27 |
| 200 | 55.29 | 3.9 | 27725 | 5 | 26 |
| 250 | 57.68 | 30.4 | 255107 | 4 | 27 |

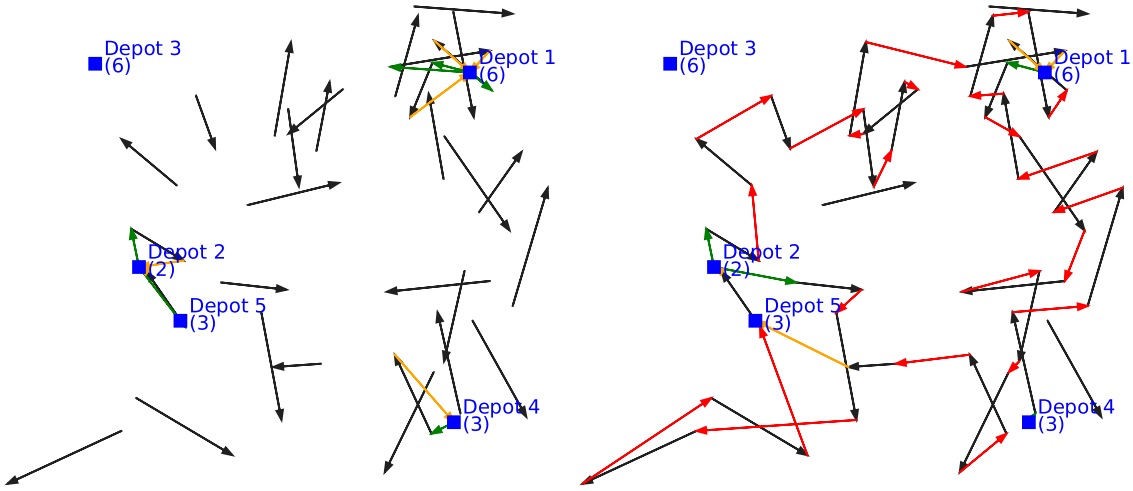



**Fig. 4.** Fleet routing solutions for an instance with 5 depots, 20 drones and 30 orders. Left: Battery capacity is 50, and 6 drones (three departing from Depot 1, two departing from Depot 2 and one departing from Depot 4) are dispatched to serve 6 orders. Right: Battery capacity is 250, 4 drones (two departing from Depot 2, one departing from Depot 1 and one departing from Depot 4) are dispatched to serve 27 orders.

on the right ($B = 250$ case) resembles the solution to a vehicle routing problem which is more challenging from the computing perspective. It can be anticipated that in practice the drone's battery capacity is neither too small nor too big. A capacity to sustain two to four shipments per charge seems to be a reachable goal given the current battery technology. Incidentally, this reality would leave sufficient room for fleet dispatch optimization without causing excessive computational difficulty.

### *7.3. Simulation study of the proposed business model*

The drone routing (DR) model addresses a temporal cross section (or snapshot) of the business operation. To be useful in practice, the model must be integrated in a periodical decision-making process to support continuous operations. In this section, we simulate such scenarios and analyze the performance of the intended business operation under different configurations.

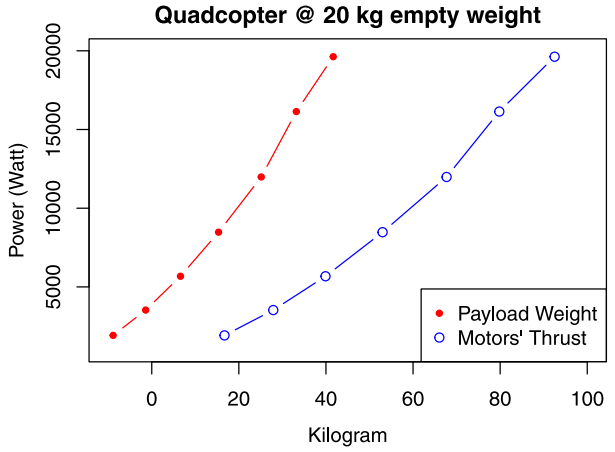
#### *7.3.1. Data and system configuration*

The drone being considered for operation is a quadcopter equipped with KDE13218XF-105 electric motors (KDE, 2021) (8.5 kg), a 12-cell 32 Ah LiPo battery pack at 44.4V (Adorama, 2021) (8.5 kg), and carbon fiber airframe and propellers (3 kg). In all, the drone's total empty weight is 20 kg and the effective battery capacity is 68160 Watt minutes (assuming 80% of the nameplate capacity is usable). We assume a constant airspeed of 15 m/s in all flights. For this drone, the relationship between payload weight and energy consumption is estimated to be the red curve in Fig. 5(a), which is produced based on the motor specification data (KDE, 2021). For ease of calculation, we use the following fitted regression line ($R^2 = 0.98$, thus quite accurate) to characterize the relationship,

$$\text{Power (Watt)} = 354.4 \times \text{Payload Weight (kg)} + 3967.5$$

In windless weather, for example, this drone is able to fly 15 km empty or 8 km while carrying 10 kg payload. A linear relation between the payload weight and battery consumption rate was also shown to be highly accurate in Torabbeigi et al. (2020). We have constructed a smaller version of the quadcopter used in this simulation and it is demonstrated in Fig. 5(b).

The service area is the city of Troy, Michigan, which spans an approximately 10 km by 10 km square area. Five depots are located in the service area, and the fleet consists of 20 drones. In other words, the computational instances encountered in operation will have $p = 5$ and $K = 20$, which, from what has been observed in Table 2, are quite amenable to the computational algorithms. Indeed, in all simulation runs, the total solution time in each dispatch optimization (using the PS approach) was consistently less than one second.




(a) Quadcopter power curve, assuming a thrust-to-weight ratio of 1.5 at 15 m/s cruise speed. Motor thrust data (the blue curve) is from KDE [2021].


(b) A scaled-down prototype of the hypothesized quadcopter used in the simulation. This prototype is equipped with KDE3510XF-715 motors and has an empty weight of around 3 kg.

**Fig. 5.** Quadcopter specification in the simulation experiment.

The shipment orders arrive randomly following a Poisson process with a given arrival rate. For each order, the origin and destination are randomly sampled within the service region, the payload weight is randomly sampled in the interval $[1, 10]$ kg and the bid value (revenue to the fleet operator) is randomly sampled in the interval $[1, 5]$ dollars. The drone routing optimization is run every five minutes. At the time of each optimization run, only the drones that are parked at a depot and only the orders that have arrived and not served are considered. In order to know which drones are available (i.e., landed) at each depot when the optimization model is run, we need to keep track of the time taken for a drone to complete an assigned trip. Specifically, the flight time in each segment of a trip is determined by the trip length, the payload weight and the wind speed and direction, in the same way as described in the formula (1).

*7.3.2. Simulation and result analysis*

We consider a business scenario where all arrived orders are eventually served. Specifically, if an order is not served (i.e., assigned to a drone) in the dispatch cycle immediately following its arrival, it will double its bid value and compete (with other new and existing orders) for service in the next dispatch cycle, and keep doing so until it is dispatched. Thus, an order's bid value implicitly prices the order's urgency of being served immediately.

We simulate 3 hours of continuous operations in each simulation run. To do so, we generate a sequence of customer orders according to the Poisson arrival process for a 3-hour period. Then, the dispatch optimization is run periodically at each five-minute mark along the simulation time line until all orders are served.

The system state evolution across successive dispatch events is explained via Fig. 6, which depicts the first two periods in a simulation run. First, let us look at the subfigure on the left which shows the system state and the dispatch solution at the end of the first period (Period 0). By this time, six orders, represented by black arrows, have arrived, and four drones are available at each depot. The optimal solution is to dispatch three drones, departing from depots 1, 3 and 4, respectively, to serve four of those ready orders. Trip segments that leave a depot, enter a depot and connect two orders are represented by green, orange and red arrows, respectively. Now, let us look at Period 1 on the right subfigure. There are nine ready orders, two of which are leftovers from the previous period, and the other seven are new arrivals during Period 1. Note that the drones available at depots 1, 3 and 4 are one fewer than there were in the previous period. This is because those drones that were dispatched in Period 0 are still in service by the end of Period 1. When a drone trip is over (at some future period as determined by the trip duration), the landing depot of the trip will see an increment in the number of drones available there.

Fig. 7 plots the instantaneous drone and order status over time, at two different order arrival rates, five arrivals per period (or equivalently one order per minute) and ten arrivals per period. Comparing the two scenarios, we can see that (1) At the arrival rate of five, the system is stable and the resource is under-utilized, for the dispatched drones are consistently fewer than the numbers available; (2) At the arrival rate of ten, the system is capacity-constrained and unstable, for the number of ready orders starts to build up monotonically at around period 27, which indicates that the service rate is lower than the order arrival rate. The dispatch period 36, the ending period for order arrival, is marked by a vertical dashed line in the plots. Observations like these can help the fleet manager understand the service capacity and fleet utilization under different hypothesized fleet configurations and uncertain operating conditions.

For the scenario with order arrival rate of five (i.e., the system is in steady-state), Fig. 8 groups the orders by their dispatch delay and plots the distribution of the orders' bid value, shipment distance and payload weight by group. The dispatch delay of an order is the time difference between the order's dispatch and arrival events. For example, if an order is assigned to a drone in the

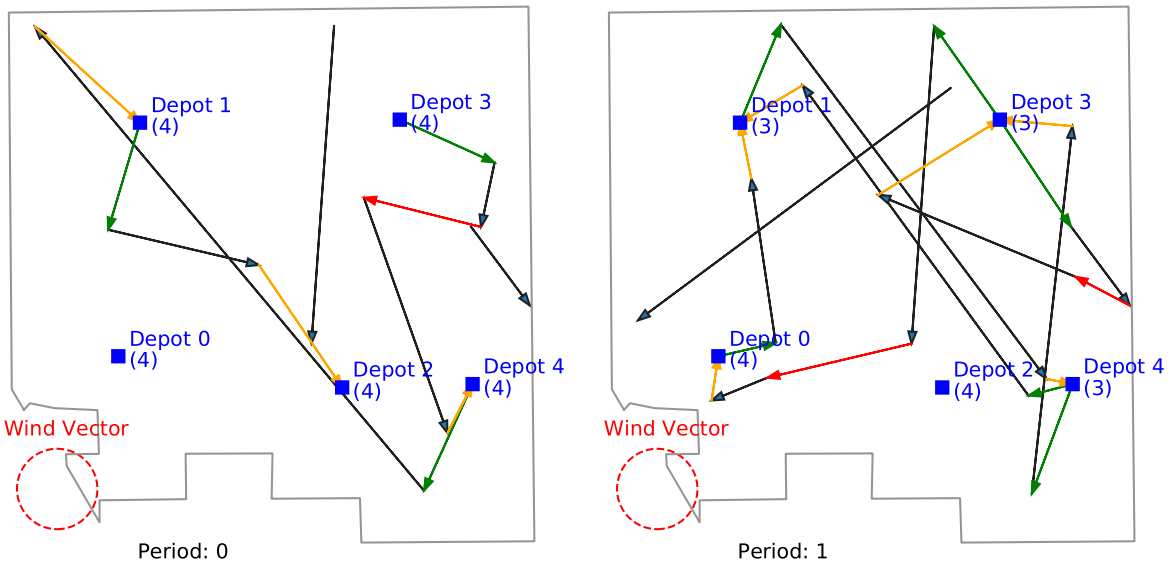
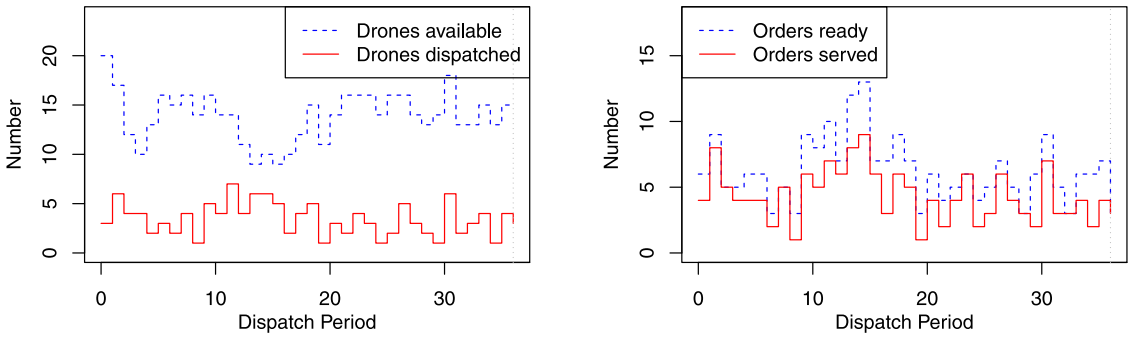
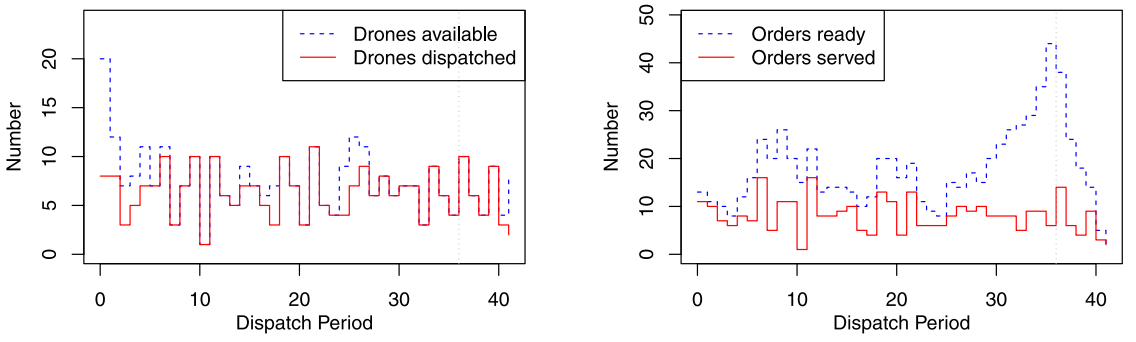



**Fig. 6.** System state and dispatch solution for the first two periods in a simulation run. The depots are strategically located in the service region to achieve the best coverage (Liu, 2022).




(a) Order arrival rate: five arrivals per five-minute period.




(b) Order arrival rate: ten arrivals per five-minute period.

**Fig. 7.** System state evolution under two order arrival rate settings. In scenario (a), the system is under-utilized, thus in steady state; in scenario (b), the system is capacity-constrained and unstable.

same period as its arrival, then its dispatch delay is 0 period. These plots reveal that the main cause of an order's dispatch delay is its low bid value. Since each drone trip costs $3, any order that bids lower than $3 cannot enjoy a dedicated drone trip even when drones are widely available at depots. Low-value orders may be "tucked" in a multi-legged trip when such an opportunity arises.

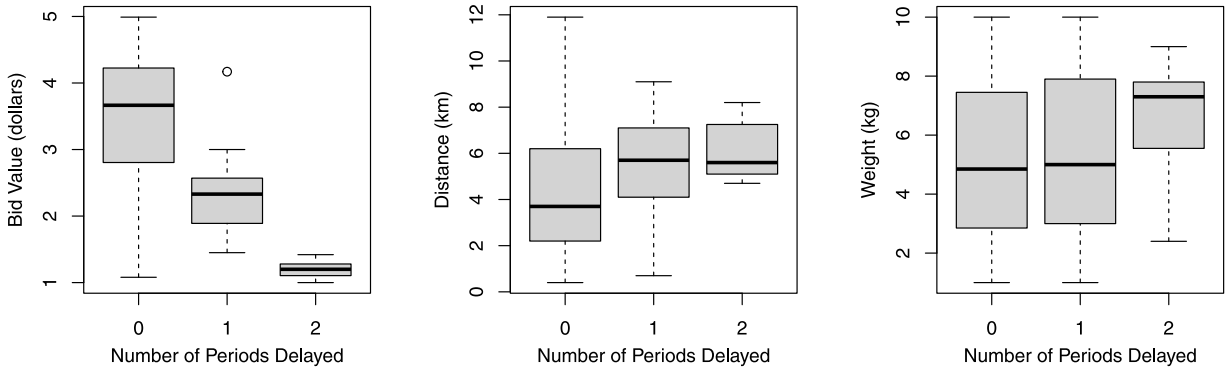



**Fig. 8.** The relationship between the dispatch delay experienced by an order and the order's bid value, shipment distance and payload wight.

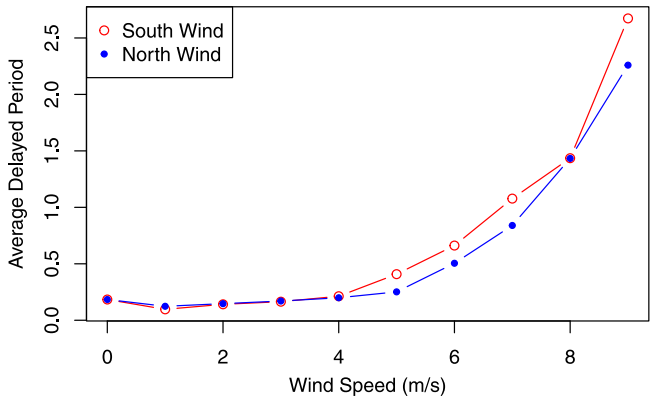



**Fig. 9.** The effects of wind on the system's service level.

Thus, a customer needs to select the right trade-off point between the cost of shipment and the risk of not being served on time. Apart from the bid value, a long shipment distance and/or a heavy payload also contributes to a longer dispatch delay, though the relationship is much weaker. All these observations are consistent with what is expected for a drone-based shipment service where battery capacity and charging cost constitute the main operating constraint and cost.

*7.3.3. The effect of wind on service quality*

In windy weather, an individual drone's battery consumption rate will differ by the flight direction. How does the effect of wind pass on to influencing the fleet level operational efficiency in the hypothesized business scenario? Simulation experiments can help answer this question. We perform simulation runs by varying the wind speed and direction, and observe shipment delays experienced by the customer orders. To control for nuisance factors, we generate random shipment orders (at Poisson rate of 3 arrivals per minute) whose OD distances range between 2 km and 3 km (excessively short or long orders are excluded) and set the payload weight to 5 kg for all orders. Other parameter settings remain the same as given in Section 7.3.2. A 3-hour simulation is run for each wind scenario, configured with wind speed in the set $[0, 1, \ldots, 9]$ m/s and wind direction in [South, North]. Note that in the 20 simulation runs the same set of orders (517 in total) were used and the only difference is in wind. The average periods of delay (over all orders) against the wind situation is plotted in Fig. 9. In mild wind (below 4 m/s), the system's average performance is barely affected; on the other hand, when the wind speed exceeds some threshold, the system becomes capacity-constrained, causing increasing shipment delays. We can conclude that the system operation remains insensitive to the wind effect when the wind speed is below a certain threshold, and will be increasingly disrupted as the wind speed exceeds this threshold.

Wind direction is not a significant factor in determining the system performance, given that customer orders are random in shipment directions. However, for an individual order, whether its shipment direction is aligned with the wind direction will affect its ease of shipment by a drone. Fig. 10 exhibits this correlation under different wind speed. In each radial plot, each line originating from the origin represents a delayed customer order, with its length indicating the periods of delay and the pointing direction indicating the order's shipment direction (pointing from the order's origin to the destination, up is north). We can see that in mild

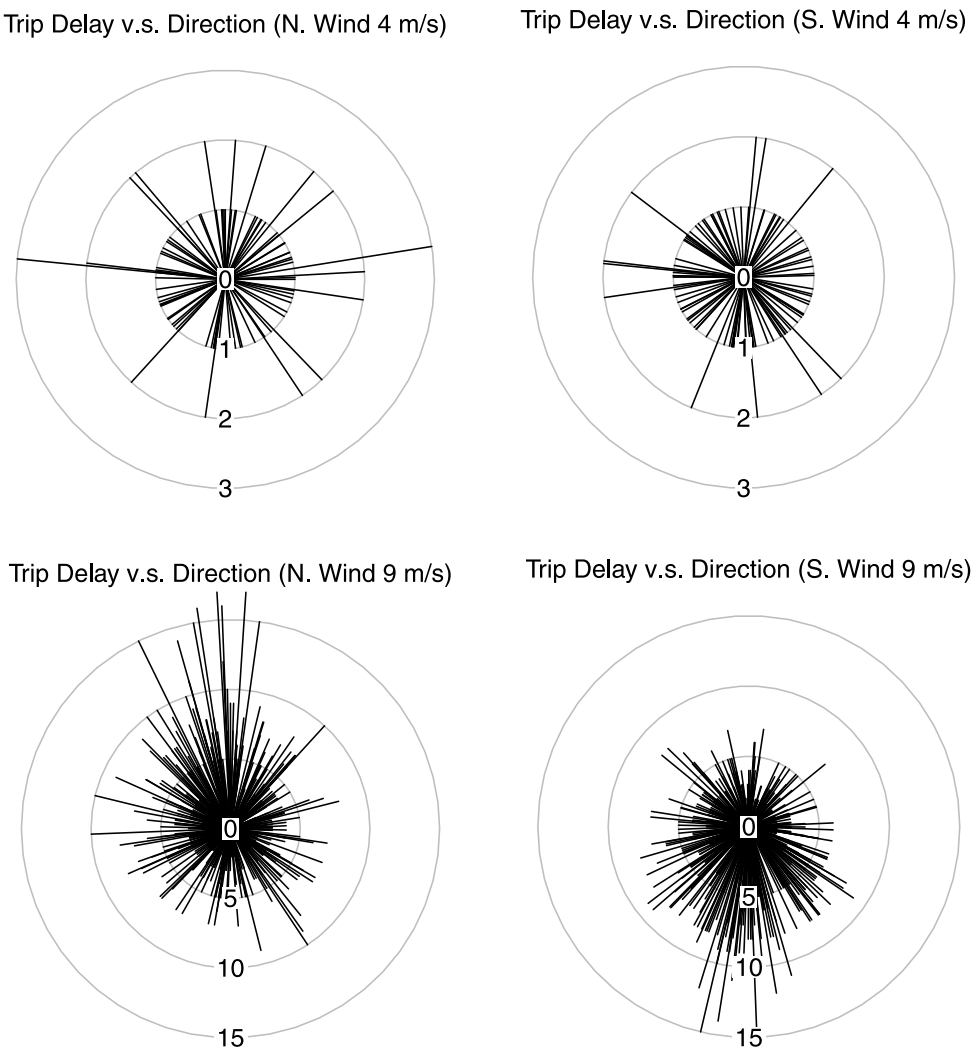



**Fig. 10.** The effect of an order's shipment direction on its shipment delay in windy conditions.

wind (i.e., 4 m/s) when the system is not capacity-constrained, the alignment between an order's shipment direction and the wind blowing direction (e.g., the south wind blows northward) does not help in reducing the order's likelihood of delay. In contrast, if the wind is so strong (i.e., 9 m/s) as to severely impact the system's capacity and cause lots of orders to queue up, then the orders that require upwind flight for shipment will generally experience longer delays. In such situations, customers who need upwind shipment would have to place a higher bid to secure a timely shipment.

*7.3.4. Management insights and practical considerations*

A dispatch interval of five minutes was considered in the proposed business operation. In practice, there is great flexibility in choosing the dispatch interval. A practical choice should depend on the actual fleet size, demand arrival rate and customer experience requirements. In fact, the purported benefit of performing optimized routing is predicated on the abundance of shipment requests. If shipment requests arise only sporadically such that most drones are available most of the time, then a greedy dispatch approach might be appropriate, that is, schedule a drone shipment tour whenever the revenue exceeds the cost. On the contrary, if shipment requests frequently exceeds the capacity of the fleet, the situation should first be mitigated by expanding the fleet size; in the meantime, a smaller dispatch interval (e.g., 2 minutes) could be used to accommodate the increased system throughput.

To further improve customer experience, a dynamic rerouting mechanism can be implemented to complement the periodic optimization process. Specifically, each new shipment request is evaluated immediately upon its arrival to determine if it can be feasibly inserted into an on-going drone tour, instead of being held until the next optimization epoch. Specialized search algorithms can be developed to handle such evaluation tasks.

## 8. Conclusion

Batteries are costly consumables for a commercial drone fleet and it is important to consider batteries' life cycle costs in fleet routing decisions. While battery capacity limitation has been extensively addressed in vehicle routing models involving drones, the charging costs from the life cycle management perspective have lacked adequate treatment.

In this paper, we have presented a new business model for drone-based on-demand shipment services in which battery charging costs are explicitly considered. We have developed two MIP formulations of the drone routing problem and proposed several solution approaches. Through performance comparison on an extensive set of experiments, we have found that the path-based formulation together with an effective path enumeration algorithm is the best solution approach, outperforming the column generation approach as well as a heuristic enhancement of it by a significant margin. The compact MIP formulation turned out to be the least amenable, thus should not be used in practical computing. We have furthermore proposed a simulation approach for validating the optimization model and analyzing the overall performance of the service system under hypothesized settings. Simulation results show that the selected model and solution approach are more than capable to support at least a startup scale business operation, e.g., one that operates a fleet of 20 drones to serve a small city. Effects of the battery capacity and wind condition on the fleet's performance have been probed, as usage examples of the simulation approach.

Uncertainty about customer demand, resource usage and operational costs often hinders entrepreneurs from turning business ideas into trial-run operations. The optimization–simulation framework can help to clear some of the uncertainty via supporting what-if analyses, parameter search and scenario evaluations. Furthermore, once the business model is validated and operational parameters determined, the optimization model vetted in the simulation phase can be seamlessly ported to the production environment. We therefore recommend the same approach be applied to assessing other similar business innovations that lack data from reality in their infancy stage.

In this and most other work that addresses drone routing problems, factors such as the airspace access, air traffic management, and takeoff and landing capacity at depots have been overlooked. Future work could explore relevant yet manageable models to capture the effects of these factors on logistic operations. Other lines of future work could include integrating strategic decisions such as depot location, fleet sizing and battery sourcing with operational decisions of fleet routing, order fulfillment and battery charging scheduling, and designing better subproblem heuristics to enhance the column generation algorithm.

## CRediT authorship contribution statement

**Yanchao Liu:** Conceptualization, Algorithm designs, Computer programming, Experimentation, Writing.

## Data availability

Data will be made available on request.

## Acknowledgments

The research is supported by the National Science Foundation (NSF), USA under grant CMMI-1944068. The market research that inspired this research was supported by National Science Foundation, USA under grant I-Corps 2018127. Members of the I-Corps team (as mentioned in Section 1) included Adarash Mishra (Entrepreneurial lead), Yanchao Liu (PI and technical lead), Zhenyu Zhou (Co-technical lead) and Eric Petersen (Industrial mentor). The prototype drone shown in Fig. 5(b) was constructed by Zhenyu Zhou. Sina Kazemdehbashi discovered a formulation error in the earlier version of the battery consumption model, which led to its correction. Their contributions are gratefully acknowledged. The author would like to thank three anonymous reviewers whose comments helped improve the quality of this paper.

## References

Adorama, 2021. Freefly Alta X 12S flight battery packs Lithium Polymer, 16Ah, Pair. URL https://www.adorama.com/fr91000421.html.

Agatz, Niels, Bouman, Paul, Schmidt, Marie, 2018. Optimization approaches for the traveling salesman problem with drone. Transp. Sci. 52 (4), 965–981. http://dx.doi.org/10.1287/trsc.2017.0791.

Archetti, Claudia, Speranza, M. Grazia, Corberán, Ángel, Sanchis, José M., Plana, Isaac, 2014. The team orienteering arc routing problem. Transp. Sci. 48 (3), 442–457.

Aydın, Nurşen, Muter, İbrahim, Birbil, Ş. İlker, 2020. Multi-objective temporal bin packing problem: An application in cloud computing. Comput. Oper. Res. (ISSN: 0305-0548) 121, 104959. http://dx.doi.org/10.1016/j.cor.2020.104959, URL https://www.sciencedirect.com/science/article/pii/S0305054820300769.

Barnhart, Cynthia, Johnson, Ellis L., Nemhauser, George L., Savelsbergh, Martin W.P., Vance, Pamela H., 1998. Branch-and-price: Column generation for solving huge integer programs. Oper. Res. 46 (3), 316–329. http://dx.doi.org/10.1287/opre.46.3.316.

Campbell, James F., Corberán, Ángel, Plana, Isaac, Sanchis, José M., 2018. Drone arc routing problems. Networks (ISSN: 0028-3045) 72 (4), 543–559. http://dx.doi.org/10.1002/net.21858.

Chao, I-Ming, Golden, Bruce L., Wasil, Edward A., 1996. The team orienteering problem. European J. Oper. Res. (ISSN: 0377-2217) 88 (3), 464–474. http://dx.doi.org/10.1016/0377-2217(94)00289-4, URL https://www.sciencedirect.com/science/article/pii/0377221794002894.

Cheng, Chun, Adulyasak, Yossiri, Rousseau, Louis-Martin, 2020. Drone routing with energy function: Formulation and exact algorithm. Transp. Res. B 139 (C), 364–387. http://dx.doi.org/10.1016/j.trb.2020.06.011.

Chung, Sung Hoon, Sah, Bhawesh, Lee, Jinkun, 2020. Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. Comput. Oper. Res. (ISSN: 0305-0548) 123, 105004. http://dx.doi.org/10.1016/j.cor.2020.105004, URL http://www.sciencedirect.com/science/article/pii/S0305054820301210.

Corberán, Angel, Eglese, Richard, Hasle, Geir, Plana, Isaac, Sanchis, Jose Maria, 2020. Arc routing problems: A review of the past, present, and future. Networks 77, http://dx.doi.org/10.1002/net.21965.
Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford, 2009. Introduction to algorithms, third ed. The MIT Press.
Costa, Luciano, Contardo, Claudio, Desaulniers, Guy, 2019. Exact branch-price-and-cut algorithms for vehicle routing. Transp. Sci. 53 (4), 946–985. http://dx.doi.org/10.1287/trsc.2018.0878.
Dell'Amico, Mauro, Montemanni, Roberto, Novellani, Stefano, 2019. Models and algorithms for the Flying Sidekick Traveling Salesman Problem. arXiv:1910.02559.
Desaulniers, Guy, Lessard, François, Hadjar, Ahmed, 2008. Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. Transp. Sci. 42 (3), 387–404. http://dx.doi.org/10.1287/trsc.1070.0223.
Desrochers, Martin, Desrosiers, Jacques, Solomon, Marius, 1992. A new optimization algorithm for the vehicle routing problem with time windows. Oper. Res. 40 (2), 342–354. http://dx.doi.org/10.1287/opre.40.2.342.
Desrosiers J., Lübbecke M.E., 2005. A primer in column generation. In: Desaulniers, G., Desrosiers, J., Solomon, M.M. (Eds.), Column Generation. Springer, Boston, MA, http://dx.doi.org/10.1007/0-387-25486-2_1.
Dorling, K., Heinrichs, J., Messier, G.G., Magierowski, S., 2017. Vehicle routing problems for drone delivery. IEEE Trans. Syst. Man Cybern.: Systems 47 (1), 70–85.
Dror, Moshe, 1994. Note on the complexity of the shortest path models for column generation in VRPTW. Oper. Res. 42 (5), 977–978. http://dx.doi.org/10.1287/opre.42.5.977.
Faiz, Tasnim Ibn, Vogiatzis, Chrysafis, Noor-E-Alam, Md., 2019. A column generation algorithm for vehicle scheduling and routing problems. Comput. Ind. Eng. (ISSN: 0360-8352) 130, 222–236. http://dx.doi.org/10.1016/j.cie.2019.02.032, URL https://www.sciencedirect.com/science/article/pii/S0360835219301172.
Feillet, Dominique, Dejax, Pierre, Gendreau, Michel, Gueguen, Cyrille, 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks 44 (3), 216–229. http://dx.doi.org/10.1002/net.20033.
Flynt, Joseph, 2019. What is the lifespan of a LiPo battery? How long they last. URL https://3dinsider.com/lipo-lifespan/.
French, Sally, 2021. how long do LiPo batteries last, and how do i know if an old LiPo battery is safe to use? URL https://www.thedronegirl.com/2020/05/24/lipo-batteries-last/.
GAMS, 2022. Gather-update-solve-scatter (GUSS). URL https://www.gams.com/39/docs/S_GUSS.html.
Garey, Michael R., Johnson, David S., 1978. Computers and Intractability: A Guide To the Theory of NP-Completeness, first ed. W.H. Freeman.
Golden, Bruce L., Levy, Larry, Vohra, Rakesh, 1987. The orienteering problem. Nav. Res. Logist. 34 (3), 307–318.
Gunawan, Aldy, Lau, Hoong, Vansteenwegen, Pieter, 2016. Orienteering problem: A survey of recent variants, solution approaches and applications. European J. Oper. Res. 225, http://dx.doi.org/10.1016/j.ejor.2016.04.059.
Irnich, Stefan, Desaulniers, Guy, 2006. Shortest path problems with resource constraints. ISBN: 0-387-25485-4, pp. 33–65. http://dx.doi.org/10.1007/0-387-25486-2_2,
Jeong, Ho Young, Song, B., Lee, Seokcheon, 2019. Truck-drone hybrid delivery routing: Payload-energy dependency and No-Fly zones. Int. J. Prod. Econ. 214, 220–233.
KDE, 2021. KDE13218XF-105 brushless motor for heavy-lift electric multi-rotor (UAS) series. URL https://www.kdedirect.com/collections/uas-multi-rotor-brushless-motors/products/kde13218xf-105 Date accessed: July 26, 2021.
Khoufi, Ines, Laouiti, Anis, Adjih, Cedric, 2019. A survey of recent extended variants of the traveling salesman and vehicle routing problems for unmanned aerial vehicles. Drones (ISSN: 2504-446X) 3 (3), URL https://www.mdpi.com/2504-446X/3/3/66.
Liu, Yanchao, 2019a. An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones. Comput. Oper. Res. 111, 1–20.
Liu, Yanchao, 2019b. A progressive motion-planning algorithm and traffic flow analysis for high-density 2D traffic. Transp. Sci. 53 (6), 1501–1525. http://dx.doi.org/10.1287/trsc.2019.0903.
Liu, Yanchao, 2020. Prompt Air Fleet Routing. URL https://youtu.be/JL0TjNJ1uWo.
Liu, Yanchao, 2021a. A note on solving DiDi's driver-order matching problem. Optim. Lett. 15, 109–125.
Liu, Yanchao, 2021b. A faster algorithm for the constrained minimum covering circle problem to expedite solving p-center problems in an irregularly shaped area with holes. Nav. Res. Logist. http://dx.doi.org/10.1002/nav.22023.
Liu, Yanchao, 2022. An elliptical cover problem for delivery network design and its solution algorithms. European J. Oper. Res. (ISSN: 0377-2217) http://dx.doi.org/10.1016/j.ejor.2022.04.034.
Lozano, Leonardo, Duque, Daniel, Medaglia, Andrés L., 2016. An exact algorithm for the elementary shortest path problem with resource constraints. Transp. Sci. 50 (1), 348–357. http://dx.doi.org/10.1287/trsc.2014.0582.
Lübbecke, Marco E., Desrosiers, Jacques, 2005. Selected topics in column generation. Oper. Res. 53 (6), 1007–1023. http://dx.doi.org/10.1287/opre.1050.0234.
Mufalli, Frank, Batta, Rajan, Nagi, Rakesh, 2012. Simultaneous sensor selection and routing of unmanned aerial vehicles for complex mission plans. Comput. Oper. Res. (ISSN: 0305-0548) 39 (11), 2787–2799. http://dx.doi.org/10.1016/j.cor.2012.02.010, URL https://www.sciencedirect.com/science/article/pii/S0305054812000366.
Murray, Chase C., Chu, Amanda G., 2015. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transp. Res. C (ISSN: 0968-090X) 54, 86–109. http://dx.doi.org/10.1016/j.trc.2015.03.005.
Murray, Chase C., Raj, Ritwik, 2020. The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. Transp. Res. C (ISSN: 0968-090X) 110, 368–398. http://dx.doi.org/10.1016/j.trc.2019.11.003, URL https://www.sciencedirect.com/science/article/pii/S0968090X19302505.
Nemhauser, George L., 2012. Column generation for linear and integer programming. Doc. Math. - Extra Volume ISMP 65–73.
Otto, Alena, Agatz, Niels, Campbell, James, Golden, Bruce, Pesch, Erwin, 2018. Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. Networks 72 (4), 411–458.
Pecin, Diego, Pessoa, Artur, Poggi, Marcus, Uchoa, Eduardo, 2017. Improved branch-cut-and-price for capacitated vehicle routing. Math. Program. Comput. 9, 61–100.
Poikonen, Stefan, Campbell, James, 2020. Future directions in drone routing research. Networks 77, http://dx.doi.org/10.1002/net.21982.
Poikonen, Stefan, Wang, Xingyin, Golden, Bruce, 2017. The vehicle routing problem with drones: Extended models and connections. Networks 70 (1), 34–43. http://dx.doi.org/10.1002/net.21746, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/net.21746.
Rardin, Ronald L., 1998. Optimization in Operations Research, first ed. Prentice Hall.
Reyes, Damián, Erera, A., Savelsbergh, M., Sahasrabudhe, S., O'Neil, Ryan J., 2018. The meal delivery routing problem. URL http://www.optimization-online.org/DB_FILE/2018/04/6571.pdf.
Riera-Ledesma, Jorge, Salazar-González, Juan José, 2017. Solving the team orienteering arc routing problem with a column generation approach. European J. Oper. Res. (ISSN: 0377-2217) 262 (1), 14–27. http://dx.doi.org/10.1016/j.ejor.2017.03.027, URL https://www.sciencedirect.com/science/article/pii/S037722171730231X.
Riley, Connor, Legrain, Antoine, Van Hentenryck, Pascal, 2019. Column generation for real-time ride-sharing operations. In: Rousseau, Louis-Martin, Stergiou, Kostas (Eds.), Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Springer International Publishing, Cham, pp. 472–487.
Roberti, R., Ruthmair, Mario, 2021. Exact methods for the traveling salesman problem with drone. Transp. Sci. 55, 315–335.
Torabbeigi, Maryam, Lim, Gino, Kim, Seon Jin, 2020. Drone delivery scheduling optimization considering payload-induced battery consumption rates. J. Intell. Robot. Syst. 97, http://dx.doi.org/10.1007/s10846-019-01034-w.

Ulmer, Marlin W., Thomas, Barrett W., 2018. Same-day delivery with heterogeneous fleets of drones and vehicles. Networks (ISSN: 0028-3045) 72 (4), 475–505. http://dx.doi.org/10.1002/net.21855.

Vanderbeck, F., 1999. Computational study of a column generation algorithm for bin packing and cutting stock problems. Math. Program. 86, 565–594.

Wang, Xingyin, Poikonen, Stefan, Golden, Bruce, 2017. The vehicle routing problem with drones: several worst-case results. Optim. Lett. (ISSN: 1862-4480) 11 (4), 679–697. http://dx.doi.org/10.1007/s11590-016-1035-3.

Wang, Zheng, Sheu, Jiuh-Biing, 2019. Vehicle routing problem with drones. Transp. Res. B 122, 350–364. http://dx.doi.org/10.1016/j.trb.2019.03.005.

Wang, Kangzhou, Yuan, Biao, Zhao, Mengting, Lu, Yuwei, 2020. Cooperative route planning for the drone and truck in delivery services: A bi-objective optimisation approach. J. Oper. Res. Soc. 71 (10), 1657–1674. http://dx.doi.org/10.1080/01605682.2019.1621671.

Yu, Miao, Nagarajan, Viswanath, Shen, Siqian, 2021. Improving column generation for vehicle routing problems via random coloring and parallelization. INFORMS J. Comput. http://dx.doi.org/10.1287/ijoc.2021.1105.

Yurek, Emine Es, Ozmutlu, H. Cenk, 2018. A decomposition-based iterative optimization algorithm for traveling salesman problem with drone. Transp. Res. C (ISSN: 0968-090X) 91, 249–262. http://dx.doi.org/10.1016/j.trc.2018.04.009.

Zhang, Juan, Campbell, James F., Sweeney II, Donald C., Hupman, Andrea C., 2021. Energy consumption models for delivery drones: A comparison and assessment. Transp. Res. D (ISSN: 1361-9209) 90, 102668. http://dx.doi.org/10.1016/j.trd.2020.102668, URL https://www.sciencedirect.com/science/article/pii/S1361920920308531.

Zhang, Lingyu, Hu, Tao, Min, Yue, Wu, Guobin, Zhang, Junying, Feng, Pengcheng, Gong, Pinghua, Ye, Jieping, 2017. A taxi order dispatch model based on combinatorial optimization. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '17, ACM, New York, NY, USA, ISBN: 978-1-4503-4887-4, pp. 2151–2159. http://dx.doi.org/10.1145/3097983.3098138.