Optimizing Drone-Assisted Last-Mile Deliveries: The Vehicle Routing Problem with Flexible Drones

Ilke Bakir[1] and Gizem Özbaygın Tiniç[2]

[1]Department of Operations, Faculty of Economics and Business, University of Groningen, 9700 AB Groningen, Netherlands

[2]Faculty of Engineering and Natural Sciences, Sabancı University, İstanbul 34956, Turkey

April 9, 2020

Abstract

The ever-widening acceptance and deployment of drones in last-mile distribution continue to increase the need for planning and managing the delivery operations involving drones effectively. Motivated by the growing interest towards drone-assisted last-mile transportation, we study a hybrid delivery system in which (multiple) trucks and (multiple) drones operate in tandem. In particular, we introduce the vehicle routing problem with flexible drones (VRPFD), which seeks to find a set of delivery routes for a fleet of trucks and drones operating in synchronization, with the goal of minimizing the makespan, *i.e.*, the return time of the very last vehicle (drone or truck) to the depot after completing its service. The VRPFD is fundamentally different from the majority of the vehicle routing problems with drones considered in the literature, because (1) drones are not dedicated to certain trucks, and (2) the trucks are allowed to visit the same location multiple times in our problem setting. We formulate the VRPFD as a mixed integer linear program on a time-space network, and present an efficient optimization algorithm based on a dynamic discretization discovery approach. We demonstrate the benefits of drone flexibility and the efficiency of the proposed solution approach through a detailed computational study performed on a newly generated set of benchmark instances. Our findings suggest that the flexible use of drones facilitates higher drone utilization and therefore results in makespan improvements. In clustered geographies, drone flexibility reduces the makespan by up to 12.12%, with an average of 5.39%. Our proposed solution approach is able to efficiently solve the VRPFD instances by making use of an intelligent lower bounding mechanism while keeping its subproblems small. Computational experiments reveal that it is able to reduce the solution time by up to a factor of 6.5 when compared to solving the VRPFD using a commercial solver. With additional computational experiments where we attempt to solve larger VRPFD instances within a time limit, we observe that our solution approach is able to find good bounds, even when directly solving the VRPFD with a commercial solver does not return a feasible solution.

Keywords: vehicle routing; drone-assisted deliveries; time-expanded networks; dynamic discretization discovery

1 Introduction

Unmanned aerial vehicles (UAVs), also known as drones, were mostly being used for military or surveillance purposes until Jeff Bezos announced in late 2013 that Amazon will start testing drone delivery to its customers. The company successfully trialled its so-called "Prime Air" drone delivery service in Cambridge, UK (December

2016) and in the U.S. (March 2017). Following Amazon, many other companies have also joined the race for delivery-by-drone with the goal of providing faster-than-ever service; examples include Walmart, DHL, UPS, FedEx, Zomato, Uber Eats, Dominos and many others around the world as outlined by the drone delivery market map 2019 (Radovic, 2019a).

The global drone market, valued at $14 billion in 2018, is expected to grow at a compound annual rate of %20.5 and hit $43 billion in 2024 with last-mile delivery services expanding the fastest within the market (Radovic, 2019b). Having received an investment of more than $300 million since 2012, drone delivery companies keep attracting attention in the market: The largest drone investment of 2019 was made into Zipline, a medical product delivery company, according to Radovic (2019b). The ever-widening acceptance and deployment of drones in last-mile distribution continue to increase the need for planning and managing the delivery operations involving drones effectively.

There are certain restrictions regarding this innovative mode of last-mile transportation despite its obvious appeal (*e.g.* drones require less manpower, are more environmentally friendly, can travel at high speeds without getting stuck in traffic etc.). Most drones currently available for delivery services are not designed to carry large or heavy parcels, so they can only be used for delivering reasonably light and small packages, and typically, one package at a time. Moreover, these drones have limited range, *i.e.*, a maximum duration that they can fly and hover in the air once dispatched for a sortie. However, a hybrid delivery system making combined use of (conventional) trucks and drones can partially alleviate the flight range limitations, *i.e.*, allowing drones to hitch rides on trucks extends the range for drone deliveries artificially, thereby, enabling drones to play a more efficient role in last-mile operations. To this end, the e-commerce giant Amazon has already patented several technologies for launching drones from trucks, vans, trains, ships as well as for the synchronization of drones with various types of autonomous vehicles (Mogg, 2018; Coombs, 2019; Michel, 2017). Several other companies offering/planning-to-offer drone deliveries have also been testing hybrid systems in which trucks and drones cooperate (see *e.g.* Burns, 2017; WORKHORSE, 2019).

Motivated by the increasing interest towards the adoption of drones as a means of last-mile transportation, we study a hybrid delivery system in which a fleet of vehicles consisting of (multiple) trucks and (multiple) drones operate in tandem. The collaboration between the two vehicle types facilitates rapid service through parallelizing the delivery operations while benefiting from the advantages of both vehicle types. In particular, trucks compensate for the limited range and capacity of drones, and drones move faster and reduce the workload of trucks, which translates into delivering more in a shorter amount of time with less fuel consumption and less manpower.

The use of drones and trucks in parallel towards achieving faster deliveries has attracted attention within the scientific community in recent years. The first study considering drone-truck collaboration in its framework is due to Murray and Chu (2015). The authors introduced the so-called flying sidekick traveling salesman problem (FSTSP) and the parallel drone scheduling traveling salesman problem (PDSTSP). Given a set of customers, the FSTSP aims to find a delivery plan for a truck-drone tandem with the objective of minimizing the time at which both vehicles complete service and return to the depot (*i.e.*, makespan) such that every customer is either served by the truck, or by the drone that operates in synchronization with the truck. In the PDSTSP, there are multiple drones and a single truck, but the drones are independent of the truck, *i.e.*, drones are dispatched from and retrieved at the depot, and thus, there is no coordination between the truck and any of the drones. Murray and Chu (2015) presented mixed integer linear programming formulations and greedy construction heuristics for both problems.

Following the work of Murray and Chu (2015), many others have investigated several variants of the problem of optimally routing a set of trucks and a set of drones operating jointly. Some examples of mixed integer linear programming models and heuristic algorithms for different variants of the single-truck, single-drone problem,

commonly referred to as traveling salesman problem with drone (TSPD), can be found in Ponza (2016), Agatz et al. (2018), Ha et al. (2018), Wang et al. (2019b), Poikonen et al. (2019), Dayarian et al. (2020), and de Freitas and Penna (2020). Although fewer, exact solution methods have been developed for the TSPD as well, namely, a dynamic programming approach due to Bouman et al. (2018), a constraint programming approach due to Tang et al. (2019), and a branch-and-price algorithm due to Roberti and Ruthmair (2019). The largest TSPD instance solved to optimality contains 39 customers (Roberti and Ruthmair, 2019), which is a clear indication of the problem difficulty even in the presence of a single truck and a single drone.

Single-truck multi-drone setting have also been investigated in several studies. Mixed integer linear programming (MILP) formulations are provided in Yoon (2018), and Murray and Raj (2020). Heuristic approaches have been proposed to address a number of variants of the problem (see *e.g.* Ferrandez et al., 2016; Chang and Lee, 2018; Mbiadou Saleu et al., 2018; Tu et al., 2018; Poikonen and Golden, 2020 and Murray and Raj, 2020). For a detailed review of literature concerning the single-truck single-drone and the single-truck multi-drone problems, we refer the interested reader to the survey due to Otto et al. (2018). A very self-contained literature review is also available in Murray and Raj (2020).

An immediate extension of the TSPD and the TSP with multiple drones is the problem class involving multi-trucks and multi-drones, namely, the vehicle routing problem with drones (VRPD), which seeks to identify the fastest delivery plan for multiple trucks and multiple drones operating in synchronization. The VRPD was introduced by Wang et al. (2017) with the motivation to study the largest possible savings that can be obtained by having drones coordinate with conventional vehicles. They make some simplifying assumptions, such as, unlimited drone battery life, same distance metric for trucks and drones etc., and derive several worst-case bounds, depending on the number of drones per truck and the ratio of the drone speed to the truck speed. Accordingly, when there are k drones per truck, and they are α times as fast as trucks, the ratio of the optimal value of the VRP with no drones having the same objective function (*i.e.*, minimizing makespan) to that of the VRPD is at most $\alpha k+1$, *e.g.* it is possible to save up to 75% with two drones per truck that are 50% faster than the trucks. Poikonen et al. (2017) extended the worst-case bounds proposed by Wang et al. (2017) considering the case where drones have limited endurance and they do not have to use the same metric as trucks.

Later, different variants of the VRPD have been addressed in a few studies, with the goal of developing solution techniques, which are mostly heuristics. Ham (2018) presented a constraint programming approach for a multi-truck version of the PDSTSP where drones can be used for both pickup and delivery operations. However, being an extension of PDSTSP, the problem setting does not involve any drone-truck synchronization. Sacramento et al. (2019) considered the FSTSP with multiple trucks, each carrying a single drone, and devised an adaptive large neighborhood search (ALNS) procedure to tackle the problem. Wang et al. (2019a) developed a three-step heuristic method for a generalization of the multi-truck FSTSP, where, in addition to the truck-carried drones, multiple independent drones are available, and can be launched/retrieved at the depot as in PSTSP.

Although in a relatively fewer studies, problem settings in which there are multiple trucks and multiple drones per truck have been explored as well. Schermer et al. (2019) proposed a matheuristic for the VRPD, and a variable neighborhood search (VNS) algorithm for an extension of the problem in which drones can rendezvous the trucks not only at customer locations, but also at some discrete points along the arcs. In Kitjacharoenchai et al. (2019a), a mixed integer programming formulation and two heuristic methods were provided for a VRPD variant, where trucks and drones have limited capacities, and a drone can visit multiple customers in a row, before meeting with the truck it is dispatched from.

In this paper, we introduce the VRP with flexible drones (VRPFD). The VRPFD seeks to find a delivery plan for a fleet of vehicles consisting of multiple trucks and multiple drones that operate in tandem, with the goal of minimizing the makespan, *i.e.*, the return time of the very last vehicle (drone or truck) to the depot after completing its service. One of the main differences of the VRPFD from the majority of the VRPD variants in

the literature is that drones are not dedicated to certain trucks in our problem setting, that is, a drone launched from a truck (or from the depot) can later rendezvous any of the trucks. As a consequence of this flexibility, it may be possible to achieve even larger savings with regard to the worst-case results reported in Wang et al. (2017) and Poikonen et al. (2017).

We are aware of only three studies considering flexible drones within a VRPD framework. Daknama and Kraus (2017) developed a nested local search procedure. Kitjacharoenchai et al. (2019b) presented a MILP formulation and an insertion based heuristic for an extension of the FSTSP in which drones have unlimited range, and only one drone can be launched/retrieved at any customer location. Wang and Sheu (2019) proposed arc and path based formulations, and an exact branch-and-price algorithm for a variant of the problem, where drones can be retrieved by the trucks only at designated docking hubs, and each drone can serve multiple customers on a sortie. The largest instance their algorithm is capable of solving optimally within a five-hour time limit contains 13 customers and two docking hubs.

All three of the above mentioned studies assuming drone flexibility (as well as many others considering dedicated drones in the literature) disregard the fact that in an optimal solution, a customer location may have to be visited more than once to facilitate truck-drone synchronizations, as demonstrated in Agatz et al. (2018) for the TSPD. More precisely, allowing customer nodes to be visited multiple times may yield solutions with better makespan values compared to the restricted case, where every customer should be visited exactly once. Therefore, we choose not to impose any restrictions on the number of visits to the customer nodes, which is perhaps one of the most challenging aspects of the VRPFD, and distinguishes our study from majority of the existing work. Some other differences of our problem setting are noted below:

- We do not constrain the number of drones that a truck can launch/retrieve at any customer node, and we assume that drones have limited endurance (unlike Kitjacharoenchai et al., 2019b).
- Trucks can retrieve drones at any customer location, but drones cannot serve more than one customer per sortie (unlike Wang and Sheu, 2019).

Our main contributions can be summarized as follows: (1) we introduce a vehicle routing problem with flexible drones that is fundamentally different from the existing VRPD variants, (2) we formulate the problem as a mixed integer linear program on a time-space network, (3) we propose an efficient optimization algorithm by adapting the dynamic discretization discovery approach in Boland et al. (2017) originally developed for solving the continuous-time service network design problem, (4) we generate a set of benchmark instances, and (5) perform a detailed computational analysis to investigate the performance of the proposed algorithm as well as to assess the benefits of flexibility in a hybrid delivery system with truck-drone coordination. To the best of our knowledge, none of the aforementioned studies involving flexible drones evaluate the value of having flexible drones over dedicated ones.

The rest of this paper is organized as follows. Section 2 formally describes the VRPFD and introduces a mathematical formulation of the problem based on a time-expanded network. Details of the dynamic discretization discovery algorithm developed for the VRPFD are presented in Section 3. Section 4 is dedicated to an extensive computational study conducted in order to analyse the efficacy of the proposed solution approach as well as to assess the benefits of using flexible drones. Finally, Section 5 provides some concluding remarks.

2 Problem Definition and Formulations

Now we formally define the vehicle routing problem with flexible drones (VRPFD) in which the goal is to find a set of routes for a fleet of vehicles consisting of trucks and drones so that the makespan, *i.e.*, the time at which all vehicles are back at the depot after completing their service, is minimized. Consider a network defined by the locations $N = N_c \cup \{s, f\}$, where

- N_c: set of customer locations,
- s: origin (starting depot) for all trucks and drones,
- f: destination (ending depot) for all trucks and drones,

and the links connecting these locations to each other. Suppose that K is the set of trucks and D is the set of drones each having a range of r units. Also, let τ_{ij}^k and τ_{ij}^d be the travel times of traversing arc (i, j) with a truck and a drone, respectively.

We assume that a drone can carry a single parcel at a time, so after each delivery, it has to meet with one of the trucks before serving another customer. Upon completing its service, the drone can either meet with and land on top of a truck, and stay there until the truck returns to the depot, or it can directly fly to the depot from the last customer location it has visited. Note that the latter is possible only if the remaining battery level of the drone is enough to reach the depot. For the sake of simplicity, we assume that all trucks are equipped with a sufficiently large number of batteries, and the batteries of the drones can be swapped instantaneously. We also assume that the time required to load the drones with a package is negligible.

In an optimal solution of the VRPFD, some customer locations may have to be visited multiple times as shown in Agatz et al. (2018). Therefore, one needs to keep track of the time of each visit to all locations in the network to be able to model truck and drone movements accurately. To this end, we define a time-expanded network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \{(i, t) : i \in N_c, 1 \leq t \leq T\} \cup \{(s, 0), (f, T)\}$ is the set of nodes, $\mathcal{A} = \{((i, t), (j, t')) : (i, t), (j, t') \in \mathcal{N}, t < t'\}$ is the set of arcs, and T is the number of time points resulting from the discretization of the planning horizon. Notice that the set $\mathcal{A}$ contains two types of arcs. Arcs of the form $((i, t), (j, t'))$ with $j \neq i$ model the possibility of trucks and drones moving from location i at time t to arrive at location j at time t', whereas arcs of the form $((i, t), (i, t'))$ model the possibility of trucks and drones to synchronize, *i.e.*, wait for one another to meet. We should note here that we do not define multiple timed copies of the origin s and the destination f, but we keep the time indices of the nodes $(s, 0)$ and (f, T) for notational consistency.

To differentiate between the two cases where the drones are moving independently of any truck and where they are attached to a truck, we extend the graph $\mathcal{G}$ by splitting each node in $\mathcal{N} \setminus \{(s, 0), (f, T)\}$ into a drone-only node and a truck node. Drone-only nodes are allowed to be visited only by drones, whereas truck nodes can be visited both by trucks and by drones. Nodes $(s, 0)$ and (f, T) are regarded as truck nodes. Since drones cannot carry multiple packages at a time, we do not add any arcs connecting drone-only nodes associated with distinct customers. Moreover, we do not add any waiting arcs incident to drone-only nodes, because drones and trucks can meet only at truck nodes, and we can assume, without loss of generality, that drones are allowed to wait only at truck nodes (if necessary). Let $\mathcal{N}_d^i$ and $\mathcal{N}_k^i$ denote the sets of drone-only nodes and truck nodes corresponding to customer $i \in N_c$, respectively. Also, let:

- $\mathcal{N}_d = \bigcup_{i \in N_c} \mathcal{N}_d^i$,
- $\mathcal{N}_k = \bigcup_{i \in N_c} \mathcal{N}_k^i \cup \{(s, 0), (f, T)\}$,

- $\mathcal{A}_d = \{((i,t),(j,t')) : (i,t) \in \mathcal{N}_k, (j,t') \in \mathcal{N}_d, i \neq j, t' = t + \tau^d_{ij}\} \cup \{((i,t),(j,t')) : (i,t) \in \mathcal{N}_d, (j,t') \in \mathcal{N}_k, i \neq j, t' = t + \tau^d_{ij}\}$,
- $\mathcal{A}_k = \{((i,t),(j,t')) : (i,t),(j,t') \in \mathcal{N}_k, i \neq j, t' = t + \tau^k_{ij}\}$,
- $\mathcal{A}_h = \{((i,t),(i,t')) : (i,t),(i,t') \in \mathcal{N}_k \setminus \{(s,0),(f,T)\}, t' = t+1\}$.

Here, $\mathcal{A}_d$ is the set of drone movement arcs (*i.e.*, arcs with one endpoint in $\mathcal{N}_d$), $\mathcal{A}_k$ is the set of truck movement arcs (*i.e.*, movement arcs with both endpoints in $\mathcal{N}_k$), and $\mathcal{A}_h$ is the set of holdover (waiting) arcs.

The VRP with flexible drones can now be formulated on the resulting time-expanded network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with $\mathcal{N} = \mathcal{N}_d \cup \mathcal{N}_k$, and $\mathcal{A} = \mathcal{A}_d \cup \mathcal{A}_k \cup \mathcal{A}_h$. For a given arc $a = ((i,t),(j,t'))$, we let

$$\bar{t}_a = \begin{cases} t + \tau^d_{ij} \text{ if } a \in \mathcal{A}_d, \\ t + \tau^k_{ij} \text{ if } a \in \mathcal{A}_k. \end{cases}$$

We define the following decision variables:

- x_a = number of trucks using arc $a \in \mathcal{A} \setminus \mathcal{A}_d$
- y_a = number of drones using arc $a \in \mathcal{A}$
- $u_i = 1$ if customer $i \in N_c$ is visited by a truck; 0 otherwise
- $v_i = 1$ if customer $i \in N_c$ is visited by a drone; 0 otherwise
- z = time period at which the last truck or drone returns to the depot

For a given node $(i,t) \in \mathcal{N}$, we use $\delta^-(i,t)$ to denote the set of all incoming arcs of (i,t) and $\delta^+(i,t)$ to denote the set of all outgoing arcs of (i,t). Finally, for a vector $\alpha \in \mathbb{R}^{|U|}$ and $U' \subseteq U$, we let $\alpha(U') = \sum_{u \in U'} \alpha_u$.

Below is a mixed integer linear programming (MILP) formulation of the VRPFD:

$$\min z \tag{1}$$

$$\text{s.t. } x(\delta^+(i,t)) - x(\delta^-(i,t)) = \begin{cases} |K| \text{ if } (i,t) = (s,0) \\ -|K| \text{ if } (i,t) = (f,T) \\ 0 \text{ otherwise} \end{cases} \qquad (i,t) \in \mathcal{N}_k \tag{2}$$

$$y(\delta^+(i,t)) - y(\delta^-(i,t)) = \begin{cases} |D| \text{ if } (i,t) = (s,0) \\ -|D| \text{ if } (i,t) = (f,T) \\ 0 \text{ otherwise} \end{cases} \qquad (i,t) \in \mathcal{N}, \tag{3}$$

$$y_a \leq |D| x_a \qquad a \in A_k, \tag{4}$$

$$y_a \leq x(\delta^-(i,t) \setminus \mathcal{A}_d) \qquad (i,t) \in \mathcal{N}_k, a \in \delta^+(i,t) \cap \mathcal{A}_d, \tag{5}$$

$$\tau^d_a y_a + \tau^d_{a'} y_{a'} \leq r \qquad (i,t) \in \mathcal{N}_d, a \in \delta^-(i,t), a' \in \delta^+(i,t), \tag{6}$$

$$u_i \leq x(\delta^-(\mathcal{N}^i_k) \cap \mathcal{A}_k) \leq |\mathcal{N}^i_k| u_i \qquad i \in N, \tag{7}$$

$$v_i \leq y(\delta^-(\mathcal{N}^i_d)) \leq |\mathcal{N}^i_d| v_i \qquad i \in N, \tag{8}$$

$$u_i + v_i \geq 1 \qquad i \in N, \tag{9}$$

$$z \geq \bar{t}_a x_a \qquad a \in \delta^-(f,T) \cap \mathcal{A}_k, \tag{10}$$

$$z \geq \bar{t}_a y_a \qquad a \in \delta^-(f,T) \cap \mathcal{A}_d, \tag{11}$$

$$y(\delta^-(i,t)) \leq 1 \qquad (i,t) \in \mathcal{N}_d, \tag{12}$$

$$x_a \in \{0,1\} \qquad a \in \mathcal{A}_k, \tag{13}$$
$$x_a \in \mathbb{Z}_+ \qquad a \in \mathcal{A}_h, \tag{14}$$
$$y_a \in \{0,1\} \qquad a \in \mathcal{A}_d, \tag{15}$$
$$y_a \in \mathbb{Z}_+ \qquad a \in \mathcal{A}_k \cup \mathcal{A}_h, \tag{16}$$
$$u_i \in \{0,1\} \qquad i \in N, \tag{17}$$
$$v_i \in \{0,1\} \qquad i \in N, \tag{18}$$
$$z \geq 0. \tag{19}$$

The objective function (1) minimizes the makespan, *i.e.*, the latest return time (over all trucks and drones) to the depot. Flow balance constraints for trucks and drones are given by (2) and (3), respectively. Constraints (4) guarantee that drones cannot use the truck movement arcs without being accompanied by a truck. Constraints (5) prevent a drone to make two deliveries in a row without meeting with a truck in between, thereby ensuring proper synchronization of drones and trucks. The range restrictions of the drones are imposed by (6). Constraints (7) and (8) establish the relationship between the variables indicating whether a customer is visited by a truck/drone and the truck/drone flow variables associated with the arcs that are incident to the customer. All customers must be visited at least once, either by a truck, or by a drone (or both) due to (9). Makespan is determined by the maximum of the arrival times of all trucks and drones at the depot as expressed by the inequalities (10) and (11). Finally, constraints (12) bound the number of visits to each drone-only node from above by one. Domain requirements for the variables are provided in (13)–(19). Due to (12), we can restrict the y variables defined for drone movement arcs to take binary values, whereas for the other arcs, drone flow variables can take any integer value ranging from 0 to $|D|$. In addition, because we assume that the travel times satisfy the triangle inequality, the x variables associated with truck movement arcs are defined as binary variables. Note that since truck nodes can be visited by multiple trucks, and there are no holdover arcs incident to the nodes $(s,0)$ and (f,T), the number of trucks waiting simultaneously at a customer location can be greater than one. Hence, we allow truck flow variables to take any non-negative integer value up to $|K|$ for holdover arcs.

Our formulation can easily be adapted to model the case where each drone is dedicated to a particular truck, and always has to work in collaboration with the same truck. To model the VRP with dedicated drones, we simply append a truck index to all x and y variables, and express the constraints in terms of these truck-indexed variables. The complete model can be found in Appendix A.

3 Solution Approach

Modelling the VRPFD on a time-expanded network has a natural consequence: the MILP formulation (1)–(19) tends to become computationally intractable as the time granularity of the network gets high. As a result, finding optimal or near-optimal solutions to the problem within a reasonable amount of time may not be possible using an off-the-shelf solver. Depending on the discretization scheme adopted, it may take hours to identify even a feasible solution, *i.e.*, the finer the time discretization is, the larger the number of variables in the MILP, and thus, the harder it becomes to solve the problem. On the other hand, reducing the time granularity of the network may yield solutions of poor quality due to the inability to detect some truck-drone synchronization opportunities.

Despite the trade-off between solution quality and discretization granularity in formulations based on time-expanded networks, full discretization, *i.e.*, modeling each point in time explicitly, is not needed to produce a continuous-time optimal solution; a partial discretization over a set of carefully chosen time points suffices (see

Boland et al., 2017). To solve the VRPFD efficiently without creating the fully time-expanded network, we propose a solution approach based on the so-called *dynamic discretization discovery* algorithm of Boland et al. (2017), which iteratively refines a partially time-expanded network until converging to optimality.

The dynamic discretization discovery algorithm (DDD) has been introduced by Boland et al. (2017), and tested, for the first time, on the continuous-time service network design problem. Due to its promising performance, the DDD then has been adapted and used to address other computationally challenging problems as well (see *e.g.* Vu et al., 2019; He et al., 2019; Lagos et al., 2020). With this motivation, we devise a dynamic discretization discovery algorithm for the VRPFD, namely, DDD-VRPFD, which iterates between solving and refining a relaxation of the problem until the solution of the relaxed problem is feasible, and therefore optimal, for the VRPFD. In the sequel, we provide the details of our solution approach.

Let $\bar{\mathcal{G}} = (\bar{\mathcal{N}}, \bar{\mathcal{A}})$ be a partially time-expanded network with $\bar{\mathcal{N}} \subset \mathcal{N}$ and $\bar{\mathcal{A}} \subset \bar{\mathcal{N}} \times \bar{\mathcal{N}}$. We denote by $VRPFD(\bar{\mathcal{G}})$ the vehicle routing problem with flexible drones defined over the network $\bar{\mathcal{G}}$. The DDD-VRPFD aims to obtain a provably optimal solution to the VRPFD by repeatedly solving and expanding a relaxation. Hence, we construct and refine the network $\bar{\mathcal{G}} = (\bar{\mathcal{N}}, \bar{\mathcal{A}})$ so that the $VRPFD(\bar{\mathcal{G}})$ is and remains to be a relaxation of the $VRPFD(\mathcal{G})$ throughout the algorithm. In particular, we maintain a partially time-expanded network that satisfies the following properties.

Property 1. *For all $i \in N_c$, the drone-only nodes $(i,0)$, (i,τ^d_{ij}), the truck nodes $(i,0)$, (i,τ^k_{ij}) as well as the nodes $(s,0)$ and (f,T) are in $\bar{\mathcal{N}}$.*

Property 2. *Every drone (truck) movement arc $((i,t),(j,t')) \in \bar{\mathcal{A}}$ has $t' \leq t + \tau^d_{ij}$ ($t' \leq t + \tau^k_{ij}$).*

Property 3. *For every $i \in N \setminus \{f\}$, $j \in N \setminus \{s,i\}$ with $(i,j) \neq (s,f)$, and $(i,t) \in \bar{\mathcal{N}}$, there is a timed copy of the arc (i,j) in $\bar{\mathcal{A}}$ originating from the timed node (i,t).*

Property 4. *If there is a drone movement arc $((i,t),(j,t')) \in \bar{\mathcal{A}}$, then there does not exist a node $(j,t'') \in \bar{\mathcal{N}}$ with $t' < t'' \leq t + \tau^d_{ij}$. Similarly, if there is a truck movement arc $((i,t),(j,t')) \in \bar{\mathcal{A}}$, then there does not exist a node $(j,t'') \in \bar{\mathcal{N}}$ with $t' < t'' \leq t + \tau^k_{ij}$.*

Consistent with the terminology used in Boland et al. (2017), we will refer to Property 4 as the *longest-feasible-arc property.*

Theorem 1. *If the partially time-expanded network $\bar{\mathcal{G}}$ satisfies Properties 1–3, then $VRPFD(\bar{\mathcal{G}})$ is a relaxation of $VRPFD(\mathcal{G})$.*

Proof. *Our proof is based on the following observations: (1) the nodes $(s,0)$ and (f,T) can be regarded as synchronization points where all trucks and drones meet, and (2) any drone or truck path in $\mathcal{G}$ can be represented as a sequence of path segments, first one starting at $(s,0)$, last one ending at (f,T), and all others starting at the end point of the preceding path segment in the sequence and ending at the next synchronization point (i.e., the next node where a truck-drone synchronization takes place in the path). Therefore, given a solution to the $VRPFD(\mathcal{G})$, it suffices to show that any path segment between two consecutive synchronization nodes in the solution can be mapped to a path segment in $\bar{\mathcal{G}}$ with equal or smaller duration.*

First, we will consider the initial segment of a given path from $(s,0)$ to (f,T) in $\mathcal{G}$. In particular, let $p = ((i_0,t_0),(i_1,t_1),\ldots,(i_m,t_m))$ be a path segment in $\mathcal{G}$ with $(i_0,t_0) = (s,0)$ and (i_m,t_m) being a synchronization node. We note here that p may involve holdover arcs if the vehicle (truck/drone), whose path contains p, has to wait for synchronization. If that is the case, the holdover arc(s) will be the last arc(s) in p since the vehicles wait at the meeting location whenever they have to. As our goal is to find a path segment $\bar{p}$ in $\bar{\mathcal{G}}$ that visits the same sequence of locations as p, and using which the time of arrival at the meeting location would be no later than that with p, we can eliminate the holdover arc(s) from p, and focus on the remaining segment. Hence, without loss of generality, we assume that $p = ((i_0,t_0),(i_1,t_1),\ldots,(i_m,t_m))$ is free of holdover arcs. We will show, by

induction, that there exists a path segment $\bar{p} = ((i_0, t'_0), (i_1, t'_1), \ldots, (i_m, t'_m))$ *in* $\bar{\mathcal{G}}$ *satisfying (1)* $(i_j, t'_j) \in \bar{\mathcal{N}}$ *with* $t'_j \leq t_j$ *for* $j = 1, \ldots, m$*, and (2)* $((i_{j-1}, t'_{j-1}), (i_j, t'_j)) \in \bar{\mathcal{A}}$ *for* $j = 1, \ldots, m$*. Letting* $(i_0, t'_0) = (s, 0)$*, we have* $(i_0, t'_0) \in \bar{\mathcal{N}}$ *by Property 1. Consider the base case where* $j = 1$*. If* $(i_1, t_1) \in \mathcal{N}^d$*, setting* $t'_1 = \tau^d_{s,i_1}$ *ensures* $t'_1 \leq t_1$ *since* $(i_1, t_1) \in \mathcal{N}^{i_1}_d$ *implies* $t_1 \geq \tau^d_{s,i_1}$*. If* $(i_1, t_1) \in \mathcal{N}^k$*, we set* $t'_1 = \tau^k_{s,i_1}$ *instead. Then, we know by Properties 1–3 that* $(i_1, t'_1) \in \bar{\mathcal{N}}$ *and* $((s, 0), (i_1, t'_1)) \in \bar{\mathcal{A}}$*. Assume that the given conditions hold for* $j = l$ *where* $2 \leq l \leq m - 1$*, that is, there exists* $((i_0, t'_0), (i_1, t'_1), \ldots, (i_l, t'_l))$ *in* $\bar{\mathcal{G}}$ *satisfying (1) and (2). Therefore,* $(i_l, t'_l) \in \bar{\mathcal{N}}$ *with* $t'_l \leq t_l$*. Suppose that* (i_l, t'_l) *is a drone-only node. By Properties 2 and 3, there exists a drone movement arc* $((i_l, t'_l), (i_{l+1}, t'_{l+1})) \in \bar{\mathcal{A}}$ *with* $t'_{l+1} \leq t'_l + \tau^d_{i_l,i_{l+1}} \leq t_l + \tau^d_{i_l,i_{l+1}} = t_{l+1}$*. Similarly, if* (i_l, t'_l) *is a truck node, Properties 2 and 3 guarantee that there is a truck movement arc* $((i_l, t'_l), (i_{l+1}, t'_{l+1})) \in \bar{\mathcal{A}}$ *with* $t'_{l+1} \leq t'_l + \tau^k_{i_l,i_{l+1}} \leq t_l + \tau^k_{i_l,i_{l+1}} = t_{l+1}$*. This proves the existence of* $\bar{p}$ *with a total travel time that is less than or equal to the travel time of* p*. Hence, for each path segment with* $(s, 0)$ *and* (i_m, t_m) *as the first and last nodes, there is an associated path segment in* $\bar{\mathcal{G}}$ *using which a truck or a drone arrives at* i_m *no later than* t_m*, implying that truck-drone synchronization can happen at a time* $t \leq t_m$*.*

Using similar arguments, one can establish a correspondence between any path segment in $\mathcal{G}$ *starting at* (i_m, t_m) *and a path segment in* $\bar{\mathcal{G}}$ *starting at* (i_m, t) *with* $t \leq t_m$*, as well as between any path segment starting and ending at an arbitrary pair of synchronization nodes in* $\mathcal{G}$ *and a path segment defined by the same sequence of locations in* $\bar{\mathcal{G}}$ *having an equal or smaller travel time. Consequently, in the solution of* $VRPFD(\bar{\mathcal{G}})$ *mapped from a given feasible solution of* $VRPFD(\mathcal{G})$*, each truck/drone arrives at the synchronization point* (f, T) *earlier than or at the same time as that in the solution of* $VRPFD(\mathcal{G})$*, proving that the* $VRPFD(\bar{\mathcal{G}})$ *is a relaxation of the* $VRPFD(\mathcal{G})$*.*

Theorem 2. *If* $\bar{\mathcal{G}}$ *satisfies the* longest-feasible-arc property*, the optimal objective function value of the* $VRPFD(\bar{\mathcal{G}})$ *yields the strongest lower bound for the* $VRPFD(\mathcal{G})$ *among all partially time-expanded networks that are induced by the same set of timed nodes* $\bar{\mathcal{N}}$ *and that satisfy Properties 1–3.*

Proof. *Consider two partially time expanded networks* $\bar{\mathcal{G}}_1$*,* $\bar{\mathcal{G}}_2$*, both defined over the same set of timed nodes* $\bar{\mathcal{N}}$*, and satisfying Properties 1–3. Suppose that* $\bar{\mathcal{G}}_2$ *also possesses the* longest-feasible-arc property*, while* $\bar{\mathcal{G}}_1$ *does not. Using the ideas in the proof of Theorem 1, it is easy to show that any path in* $\bar{\mathcal{G}}_1$ *can be mapped to a path in* $\bar{\mathcal{G}}_2$ *with greater or equal total travel time, implying that the optimal objective function value of* $VRPFD(\bar{\mathcal{G}}_2)$ *is at least as large as that of* $VRPFD(\bar{\mathcal{G}}_1)$*. This is indeed a consequence of the fact that the set of feasible solutions to the latter problem is a subset of that to the former.*

We solve the VRPFD by creating and iteratively refining a partially time-expanded network, $\bar{\mathcal{G}}$, which fulfills Properties 1–4. Every time an optimal solution to the relaxation $VRPFD(\bar{\mathcal{G}})$ is obtained, the algorithm makes an attempt to convert it into a feasible solution to $VRPFD(\mathcal{G})$ with the same makespan value. In case of success, the algorithm terminates with an optimal solution to $VRPFD(\mathcal{G})$; otherwise, it identifies at least one arc whose length is to be corrected, refines $\bar{\mathcal{G}}$ accordingly, and iterates. An algorithmic summary is given in Algorithm 1, and important steps of the algorithm are detailed in the remainder of this section.

Algorithm 1 Dynamic Discretization Discovery (DDD) for VRPFD

Generate initial partially time-expanded network, $\bar{\mathcal{G}}$
$\underline{z} := -\infty$, $\bar{z} := \infty$
while $\bar{z} - \underline{z} > 0$ **do**
    Solve $VRPFD(\bar{\mathcal{G}})$; obtain lower bound, $\underline{z}$, and route, $\mathcal{P}(v)$, of each vehicle v
    Solve upper bound model; update best upper bound, $\bar{z}$
    if $\bar{z} - \underline{z} < \epsilon$ **then**
        STOP. The optimal solution is found.
    end if
    while there are subtours in optimal solution of $VRPFD(\bar{\mathcal{G}})$ **do**
        Refine $\bar{\mathcal{G}}$ by lengthening the subtour arcs that are too short
        Solve $VRPFD(\bar{\mathcal{G}})$; obtain lower bound, $\underline{z}$, and route, $\mathcal{P}(v)$, of each vehicle v
    end while
    Solve refinement model and identify arcs to lengthen
    Refine $\bar{\mathcal{G}}$ by lengthening these arcs
end while

The DDD-VRPFD converges to an optimal VRPFD solution in finite number of iterations, because (1) it stops when the solution of a relaxed problem can be transformed into a feasible VRPFD solution with the same objective function value, (2) at least one arc is lengthened at each iteration, and (3) there are finitely many arcs in the fully time-expanded network, implying that there are finitely many arcs that can be lengthened throughout the solution process.

3.1 Generating Initial Partially Time-Expanded Network

The initial partially time-expanded network $\bar{\mathcal{G}} = (\bar{\mathcal{N}}, \bar{\mathcal{A}})$ consists of the origin and destination nodes $(s, 0)$ and (f, T) as well as the truck nodes $(i, 0)$, (i, τ^k_{0i}) and the drone-only nodes $(i, 0)$, (i, τ^d_{0i}) for every customer $i \in N_c$. Notice that this is in line with Property 1.

For every truck node $(i, t) \in \bar{\mathcal{N}} \setminus \{(f, T)\}$ and every $j \in N_c \setminus \{i\}$: (1) we find the truck node $(j, t') \in \bar{\mathcal{N}}$ with the largest t' such that $t' \leq t + \tau^k_{ij}$, and add the truck movement arc $((i, t), (j, t'))$ to $\bar{\mathcal{A}}$, (2) we find the drone node $(j, t') \in \bar{\mathcal{N}}$ with the largest t' such that $t' \leq t + \tau^d_{ij}$, and add the drone movement arc $((i, t), (j, t'))$ to $\bar{\mathcal{A}}$. Similarly, for every drone-only node $(i, t) \in \bar{\mathcal{N}}$ and every $j \in N_c \setminus \{i\}$, we find the truck node $(j, t') \in \bar{\mathcal{N}}$ with the largest t' such that $t' \leq t + \tau^d_{ij}$, and add the drone movement arc $((i, t), (j, t'))$ to $\bar{\mathcal{A}}$. Note here that such nodes (j, t') always exist since $(j, 0) \in \bar{\mathcal{N}}$ for all $j \in N_c$, and consequently, we ensure that the network possesses the longest-feasible-arc property by construction. Finally, we also add the holdover arcs $((i, 0), (i, \tau^k_{0i}))$ for every $i \in N_c$, and the depot return arcs $((i, t), (f, T))$ for all $(i, t) \in \bar{\mathcal{N}} \setminus \{(s, 0), (f, T)\}$.

The initial partially time-expanded network $\bar{\mathcal{G}}$ built this way satisfies Properties 1–4. Since we construct the arcs of the network relying on the *longest-feasible-arc* property, some nodes may have either no incoming or no outgoing arcs incident to them. Obviously, such nodes cannot be a part of any feasible solution to $VRPFD(\bar{\mathcal{G}})$. Therefore, after creating the initial network, we apply a preprocessing stage to remove the nodes (and their incident arcs) with either no incoming or no outgoing arcs.

3.2 Solving $VRPFD(\bar{\mathcal{G}})$

As mentioned earlier, the dynamic discretization discovery algorithm introduced in Boland et al. (2017) iterates between solving a relaxation (*i.e.*, the problem defined on a partially time-expanded network with Properties 1–4), and refining the partially time-expanded network until the optimal solution of the relaxation can be translated into a feasible solution of the original problem (*i.e.*, the problem defined on the fully time-expanded network). In other words, the relaxed problems are solved to optimality at each iteration. However, especially due the makespan minimization objective, the VRPFD has a highly symmetric solution space, as a result of which, it may not always be possible to find an optimal solution to the relaxed problems quickly.

To accelerate the solution process, we propose to limit the amount of time spent on solving each relaxation and apply the refinement steps at the end of this time limit regardless of whether an optimal solution has been found or not. This *early refinement scheme*, which we implemented using the MIPSOL callback feature of Gurobi, allows prematurely refining the network if the solution time of a relaxed problem hits a prespecified limit (in our experiments, 120 seconds for instances with 10 customers; 720 seconds for larger instances). If the incumbent solution has subtours, the subtour arcs that are too short are lengthened to their correct length; otherwise the refinement procedure described in Section 3.3 is applied. Although this *early refinement scheme* may cause the partially time-expanded network to grow faster, it speeds up the convergence of the algorithm based on our computational tests.

As another idea to enhance algorithmic performance, we also considered adding subtour elimination constraints using the set of customer locations rather than the set of nodes in a particular partially time-expanded network so that they can be added to the MILP for any partially time-expanded network throughout the solution process. This is similar to the approach presented in Vu et al. (2019) wherein a DDD algorithm is developed for solving the time-dependent traveling salesman problem with time windows. However, since we observed that adding subtour elimination constraints did not improve the algorithmic performance based on the results of our preliminary experiments, we decided not to include them in our final computations. Instead, whenever there are subtours in an intermediate solution, the network is refined so that the subtour arcs that are too short are lengthened to their actual length.

3.3 Refining the Partially Time-Expanded Network

At every iteration of the algorithm, the VRPFD is solved on the current partially time-expanded network, $\bar{\mathcal{G}}$, and a lower bound, $\underline{z}$, on the optimal makespan of $VRPFD(\mathcal{G})$ is obtained. Subsequently, the upper bound model is solved in order to convert the solution of $VRPFD(\bar{\mathcal{G}})$ into a feasible solution of $VRPFD(\mathcal{G})$ with the same makespan $\underline{z}$. If such a solution is found, the algorithm terminates, as this is an optimal solution to $VRPFD(\mathcal{G})$. However, if this conversion is unsuccessful, then there must be at least one arc in the optimal solution of $VRPFD(\bar{\mathcal{G}})$ that is shorter than its actual length. To identify such arc(s), we solve a *refinement model*, which is defined in what follows.

Let V denote the set of all vehicles (drone and truck), and $\mathcal{P}(v)$ denote the set of timed arcs, *i.e.*, the path, to be traced by vehicle v. $\mathcal{A}^{Docked}$ and $\mathcal{N}^{Redock}$ denote the set of arcs on which a drone travels docked on a truck and the set of nodes where a drone re-docks on a truck, respectively. Due to short arcs, it is possible that a node is visited by the same vehicle multiple times. For every re-docking node $n \in \mathcal{N}^{Redock}$, drone and truck arrivals are stored in sets $\Lambda_d^-(n)$ and $\Lambda_k^-(n)$, respectively. Both sets contain (v, a) pairs, where v denotes the vehicle arriving at the re-docking node, and a denotes the last movement arc (so, excluding holdover arcs) traversed by vehicle v before arriving at node n. Note that $\mathcal{P}(v)$, $\mathcal{A}^{Docked}$, $\mathcal{N}^{Redock}$, Λ_d^-, and Λ_k^- can be easily extracted from a given solution to $VRPFD(\bar{\mathcal{G}})$.

We define σ_a^v, θ_a^v, and γ_n^v variables to denote whether arc a is allowed to be too short when used by vehicle v, travel time of vehicle v on arc a, and dispatch time of vehicle v from timed node n, respectively. Additionally, α^n_{d,a_d,k,a_k} variables represent whether drone d, which arrives at the re-docking node n through arc a_d docks on truck k, which arrives at the re-docking node through arc a_k.

The actual travel time for vehicle v to traverse arc a is τ_a^v, and the length of this arc in $\bar{\mathcal{G}}$ is $\bar{\tau}_a^v$. For a given arc a, $o(a)$ and $d(a)$ represent the origin and destination nodes, respectively. $D(a)$ denotes the sets of drones traversing arc a.

The refinement model aims to identify the fewest number of arcs that have to be too short in order to convert the solution of $VRPFD(\bar{\mathcal{G}})$ into a feasible solution to $VRPFD(\mathcal{G})$ with a makespan of at most $\underline{z}$, which preserves the docked travel and re-docking locations dictated by the solution of $VRPFD(\bar{\mathcal{G}})$. The MILP formulation of the refinement model is given in (20) - (31).

$$\min \sum_{v \in V} \sum_{a \in \mathcal{P}(v)} \sigma_a^v \tag{20}$$

$$\text{s.t. } \gamma^v_{(f,T)} \leq \underline{z} \qquad v \in V \tag{21}$$

$$\theta_a^v \geq \tau_a^v (1 - \sigma_a^v) \qquad v \in V, a \in \mathcal{P}(v) \tag{22}$$

$$\gamma^v_{o(a)} + \theta_a^v \leq \gamma^v_{d(a)} \qquad v \in V, a \in \mathcal{P}(v) \tag{23}$$

$$\gamma^d_{o(a)} = \gamma^{k(d,a)}_{o(a)} \qquad a \in \mathcal{A}^{Docked}, d \in D(a) \tag{24}$$

$$\gamma^k_{o(a_k)} + \theta^k_{a_k} \leq \gamma^d_n + M(1 - \alpha^n_{d,a_d,k,a_k}) \qquad n \in \mathcal{N}^{Redock}, (d,a_d) \in \Lambda_d^-(n), (k,a_k) \in \Lambda_k^-(n) \tag{25}$$

$$\gamma^d_{o(a_d)} + \theta^d_{a_d} \leq \gamma^k_n + M(1 - \alpha^n_{d,a_d,k,a_k}) \qquad n \in \mathcal{N}^{Redock}, (d,a_d) \in \Lambda_d^-(n), (k,a_k) \in \Lambda_k^-(n) \tag{26}$$

$$\sum_{(k,a_k) \in \Lambda_k^-(n)} \alpha^n_{d,a_d,k,a_k} \geq 1 \qquad n \in \mathcal{N}^{Redock}, (d,a_d) \in \Lambda_d^-(n) \tag{27}$$

$$\sigma_a^v \in \{0,1\} \qquad v \in V, a \in \mathcal{P}(v) \tag{28}$$

$$\theta_a^v \geq \bar{\tau}_a^v \qquad v \in V, a \in \mathcal{P}(v) \tag{29}$$

$$\gamma_n^v \geq 0 \qquad v \in V, n \in \{o(a) : a \in \mathcal{P}(v)\} \cup \{(f,T)\} \tag{30}$$

$$\alpha_n^{d,k} \in \{0,1\} \qquad n \in \mathcal{N}^{Redock}, (d,a_d) \in \Lambda_d^-(n), (k,a_k) \in \Lambda_k^-(n) \tag{31}$$

Objective function (20) minimizes the number of arcs that are too short. Constraints (21) ensure that the network after lengthening still has a makespan no more than $\underline{z}$. Constraints (22) set the σ values of short arcs to 1, while constraints (23) establish the relationship between dispatch times and travel times. Constraints (24) enforce that when a drone d traverses arc a docked on truck $k(d,a)$, dispatch times of d and $k(d,a)$ from the origin of arc a are the same. Constraints (25) and (26) enforce that if drone d arriving through arc a_d and truck k arriving through arc a_k are meeting at node n for re-docking, they must wait for each other before leaving node n. Constraints (27) ensure that at least one truck is synchronized with a drone d if it needs to re-dock at node n. Constraints (28) - (31) establish the ranges of decision variables.

If the optimal objective value of the refinement model is zero, the solution to the partially time-expanded problem can be converted to a feasible solution to the fully time-expanded problem with the same makespan, which means that the current solution is an optimal solution to $VRPFD(\mathcal{G})$. If the optimal objective value is greater than zero, then the arcs a with $\sigma_a^v = 1$ for some $v \in V$ are identified as arcs to be lengthened to their actual length. To lengthen these arcs to their actual length while preserving Properties 1–4, we follow a refinement procedure similar to that of Boland et al. (2017).

Details of our arc lengthening procedures for truck and drone arcs are given in Algorithms 2 and 3, respectively. The lengthening procedure adds a new time point, t_j^{new}, and a new node, (j, t_j^{new}), to the partially time expanded network. Subsequently, the holdover arcs are accordingly refined (if the new node is a truck node), and arcs

emanating from and entering the new node are added while preserving Properties 1–4. At the end of the procedure, if there are nodes left with zero in- or out-degree, these nodes and their incident arcs are removed, since no flow can be sent through these nodes/arcs.

Algorithm 2 Lengthen truck arc $a = ((i, t_i), (j, t_j)) \in \mathcal{A}_k$

$t_j^{new} := t_i + \tau_{ij}^k$
Let $t_j^- := max\{t : (j,t) \in \mathcal{N}_k, t < t_j^{new}\}$, $t_j^+ := min\{t : (j,t) \in \mathcal{N}_k, t > t_j^{new}\}$
Add node (j, t_j^{new})
Remove holdover arc $((j, t_j^-), (j, t_j^+))$, add holdover arcs $((j, t_j^-), (j, t_j^{new}))$ and $((j, t_j^{new}), (j, t_j^+))$
for all movement arcs $((j, t_j^-), (k, t_k)) \in \mathcal{A}_k \cup \mathcal{A}_d$ emanating from node (j, t_j^-) **do**
    Add $((j, t_j^{new}), (k, t_k^{LF}))$, where t_k^{LF} is the period that preserves the longest-feasible-arc property
end for
for all movement arcs $((i, t_i), (j, t_j^-)) \in \mathcal{A}_k \cup \mathcal{A}_d$ entering node (j, t_j^-) such that t_j^{new} preserves the longest-feasible-arc property **do**
    Remove movement arc $((i, t_i), (j, t_j^-))$
    Add movement arc $((i, t_i), (j, t_j^{new}))$
end for

Algorithm 3 Lengthen drone arc $a = ((i, t_i), (j, t_j)) \in \mathcal{A}_d$

$t_j^{new} := t_i + \tau_{ij}^d$
Let $t_j^- := max\{t : (j,t) \in \mathcal{N}_d, t < t_j^{new}\}$, $t_j^+ := min\{t : (j,t) \in \mathcal{N}_d, t > t_j^{new}\}$
Add node (j, t_j^{new})
for all movement arcs $((j, t_j^-), (k, t_k)) \in \mathcal{A}_k \cup \mathcal{A}_d$ emanating from node (j, t_j^-) **do**
    Add $((j, t_j^{new}), (k, t_k^{LF}))$, where t_k^{LF} is the period that preserves the longest-feasible-arc property
end for
for all movement arcs $((i, t_i), (j, t_j^-)) \in \mathcal{A}_k \cup \mathcal{A}_d$ entering node (j, t_j^-) such that t_j^{new} preserves the longest-feasible-arc property **do**
    Remove movement arc $((i, t_i), (j, t_j^-))$
    Add movement arc $((i, t_i), (j, t_j^{new}))$
end for

3.4 Computing Upper Bounds

An important step in DDD-VRPFD is to convert the solution of the partially time-expanded model to a feasible solution to the fully time-expanded model with the same makespan value. As noted in Section 3.3, if the optimal objective value of the refinement model is zero, then this conversion is successful without needing to lengthen any arcs in the partially time-expanded network. However, when the refinement model returns a non-zero objective value, the partially time-expanded solution cannot be converted to a feasible solution with actual arc lengths, and therefore, one or more of the arcs must be lengthened to their true length based on actual travel times.

At any iteration of DDD-VRPFD, even when the conversion to a feasible solution with the same makespan is unsuccessful, the solution to the partially time-expanded model, $VRPFD(\bar{\mathcal{G}})$, can be converted to a feasible solution to the fully time-expanded model, $VRPFD(\mathcal{G})$. This conversion preserves the order of visited nodes for each vehicle, and the synchronization of drones and trucks when a drone re-docks on a truck and/or travels docked. However, the preservation of the makespan is not guaranteed in this conversion process. It is important to note that a converted solution provides a feasible solution to the fully time-expanded model, and therefore,

an upper bound to the optimal makespan.

In what follows, we present a MILP that obtains an upper bound for $VRPFD(\mathcal{G})$ by converting the solution of $VRPFD(\bar{\mathcal{G}})$ into a feasible solution for the fully time-expanded model with minimum makespan. The objective function (32) minimizes the makespan, denoted as μ. Constraints (33) establish that the overall makespan is defined as the time period in which all vehicle routes are completed. Constraints (23) – (27), (30), (31), which ensure that the converted solution conforms with the routes and the synchronization patterns of the partially time-expanded solution, are the same as in the refinement model. Constraints (34) enforce that the travel time on each arc must be at least as long as the actual travel time.

$$\min \ \mu \tag{32}$$

$$\text{s.t. } \mu \geq \gamma^v_{(f,T)} \qquad v \in V \tag{33}$$

$$(23) - (27), (30), (31)$$

$$\theta^v_a \geq \tau^v_a \qquad v \in V, a \in \mathcal{P}(v) \tag{34}$$

When the optimal objective value of the above MILP is equal to the makespan of $VRPFD(\bar{\mathcal{G}})$, the corresponding solution is optimal, thus the algorithm stops. Otherwise, this objective value serves as an upper bound, with which we assess how far we are from the optimal makespan.

4 Computational Experiments

In this section, we demonstrate the operational benefits of flexible drones and assess the performance of DDD-VRPFD in solving the VRPFD. We first provide the details of our experimental setting and problem instances in Section 4.1. Section 4.2 highlights the potential in using drones flexibly with computational experiments. Subsequently, Section 4.3 provides a detailed performance analysis of DDD-VRPFD, and Section 4.4 demonstrates the performance of DDD-VRPFD in finding good bounds for VRPFD on larger problem instances, solving which to optimality was impractical.

4.1 Experimental Setting

Computational experiments are conducted on an extensive set of problem instances with varying characteristics, namely, *(i)* geography, *i.e.*, number and locations of customers, *(ii)* number of vehicles, *(iii)* drone/truck speed ratio, and *(iv)* time granularity. For investigating solution characteristics and algorithm performance, the following performance measures are used: *(i)* makespan (or lower and upper bounds for makespan, when applicable), *(ii)* number of arcs in the fully or partially time-expanded networks, and *(iii)* solution time. All experiments are implemented with Python 3.6, by using Gurobi 8.0.1 for solving MILPs, and are executed on a computing cluster with Intel Xeon E5 2680v3 2.5 GHz processors. The remainder of this section presents the details of problem instances used in experimentation.

Locations and Travel Times. For all experiments, single-depot problem instances are considered. Depot and customer locations are randomly generated on a 2-dimensional plane. Based on these locations, Euclidean distances are computed and rounded to the nearest integer (in order to properly control the time granularity), and an all-pairs shortest path algorithm is executed to correct these distances to satisfy the triangle inequality. With the assumption that the travel time is directly proportional to the travel distance, these distances are used in representing the travel times (in number of time periods) between locations. The maximum number of

time periods needed in the time-expanded network is computed by solving a makespan minimizing VRP with only trucks (no drones).

Time Granularity. An important strength of DDD is that its performance is minimally affected by time granularity, by construction. To strengthen this claim with computational experiments, we use multiple time granularity values, namely, 1, 2, 3, 4, 5, 6, and 7. These values can be thought of as number of time periods in a unit time, say, an hour. For example, in time granularity 1, denoted as $\Delta = 1$, each hour can correspond to a time period (1 time period = 1 hour), whereas in time granularity 7, denoted as $\Delta = 7$, each hour is divided into 7 equal time periods (1 time period = $1/7$ hour).

Instance Size. Since it is of interest to compare the solution performance of DDD in solving the VRPFD to that of a commercial solver, it is desirable to have a small enough number of customers for both methods to terminate with an optimal solution in reasonable time. To achieve this, the majority of the problem instances are designed with 10 customers. In Section 4.4, where the lower and upper bounds obtained with both methods are presented, instances with 25 customers are used.

Vehicles. In order to observe solution characteristics and algorithm performance under a wide range of parameter settings, we consider various number of vehicles and speed ratios. All problem instances have 2 trucks, and 1, 2, or 3 drones. Drone/truck speed ratios of 1, 1.5, and 2 are considered, where a speed ratio ρ denotes that the drones are ρ times as fast as the trucks, and therefore the travel times of the drones are $1/\rho$ times the travel time of trucks.

Drone Endurance. Drone endurance can significantly differ between different technologies. As drone endurance greatly impacts the solution space of our problem, different endurance values are used in computational experiments. To simulate *short*, *medium*, and *long* drone endurance ranges, 25^{th}, 50^{th}, and 75^{th} percentiles of all possible drone sortie travel times, *i.e.*, $\tau^d_{ij} + \tau^d_{jk}, \forall i, j, k \in N, j \notin \{s, f\}$, respectively, are used.

4.2 Impact of Flexible Drones

The motivation for using a flexible drone scheme, where drones can be retrieved by any truck regardless of which truck they were launched from, instead of a dedicated drone scheme, where each drone must be retrieved by the truck from which it was launched, is the potential for improved makespan. In this section, we demonstrate the improvements in makespan, drone utilization, and wait times granted by using drones flexibly. We use two different types of geographies for these computational experiments.

In order to demonstrate the impact of flexible drones, we solve the VRPFD, as well as a problem variant that restricts each drone to be *dedicated* to a truck, *i.e.*, it must be launched from and retrieved by the same truck. As mentioned earlier, the formulation for this problem variant, which we call the VRP with dedicated drones (VRPDD), is given in Appendix A.

As an illustrative example, consider a network with 10 customers with the geography given in Figure 1. On this network, we solve the VRPFD, as well as the VRPDD, with 2 trucks and 1 long-endurance drone that is 50% faster than the trucks. The optimal makespan values for the VRPFD and the VRPDD are 58 and 66, and the optimal solutions are shown in Figure 2. It can be clearly seen that VRPFD takes advantage of the drone's flexibility in switching between trucks, and enables the drone to be utilized more extensively, without having to wait for the truck from which it was launched. In the VRPFD solution, the drone travels a visibly longer distance in total, and serves 5 customers, as opposed to the VRPDD solution, where the drone can serve only 3 customers due to having to wait for the truck at the meeting node for retrieval. Since the drone is 50% faster than the trucks, more extensive use of the drone enabled by drone flexibility results in a significantly shorter

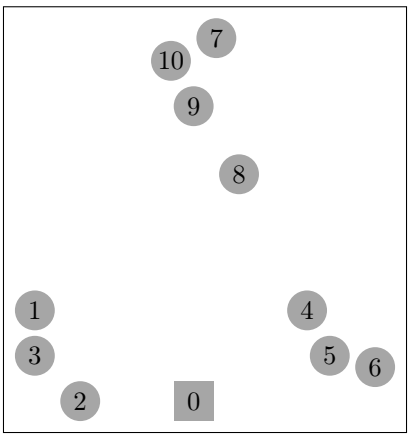
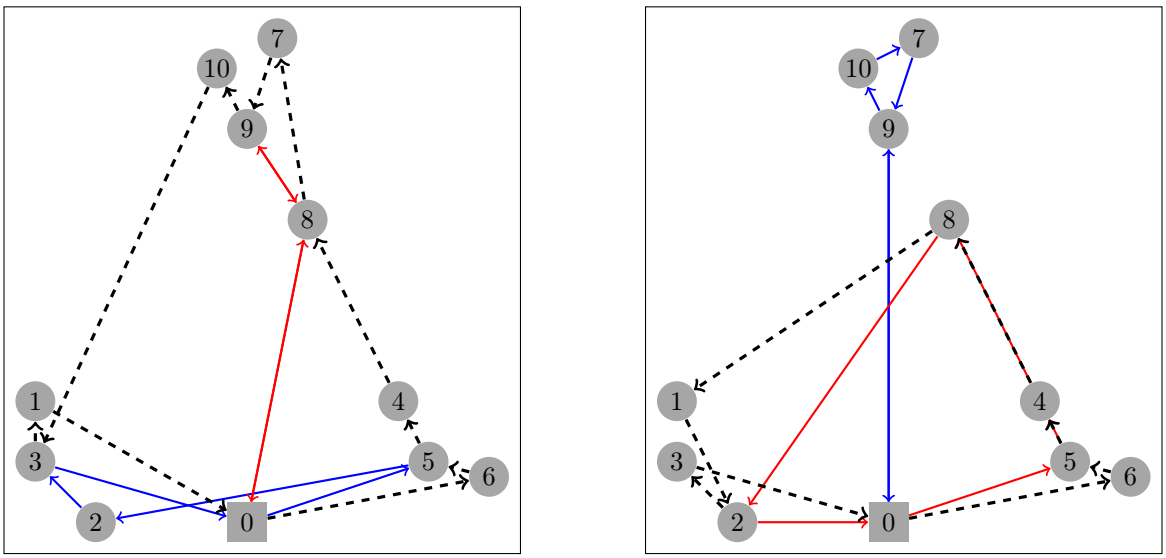


Figure 1: Clustered geography

makespan.



Note. Solid arcs denote truck moves (for each of the two trucks), and dashed arcs denote drone moves.

Figure 2: Solution to VRP with a flexible (left) and a dedicated (right) drone

A large set of experiments is conducted in order to investigate the impact of drone flexibility. For this purpose, two types of geographies are created: *uniform* and *clustered*. In both types of geographies, the depot is located in the lower center (as in Figure 1). In uniform geography, customer locations are uniformly generated on a square grid. Clustered geography, which is generated to simulate suburban neighborhoods, has three clusters in the lower left, lower right, and upper center of the square grid, and customer locations are uniformly dispersed within each cluster. Figure 1 depicts an example clustered geography.

Tables 1 and 2 present the optimal makespan values attained with dedicated and flexible drones for uniform and clustered geographies, respectively, on representative problem instances. It can be observed in Table 1 that in uniform geographies, flexible drones provided a makespan improvement in 9 out of 30 instances, when compared to the dedicated drone case. The average makespan is 61.90 and 61.03 for dedicated and flexible drones, respectively, which translates into a 1.40% makespan improvement on average. Table 2 provides evidence that the impact of drone flexibility is more prominent in clustered geographies. It can be observed that in clustered geographies, flexible drones resulted in a shorter makespan than dedicated drones in 22 out of the 30 instances. Furthermore, the average makespan values are 58.70 and 55.53 for dedicated and flexible drones, respectively, which means that drone flexibility yields an average makespan reduction of 5.39% in the test instances with a clustered geography.

Table 1: Dedicated vs. flexible makespan (Uniform geography)

Instance #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Makespan (Dedicated)	70	52	64	64	60	67	58	60	58	58	55	61	64	62	60
Makespan (Flexible)	70	52	64	60	60	64	56	60	58	58	55	61	64	62	60
Instance #	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Makespan (Dedicated)	64	64	64	60	56	70	54	62	56	67	68	63	68	64	64
Makespan (Flexible)	62	64	64	60	56	66	52	60	56	64	68	63	64	64	64

Note. The figures in this table belong to problem instances with 10 customers, 2 trucks, and 1 drone, where the drone has a long endurance and is 50% faster than the trucks.

Table 2: Dedicated vs. flexible makespan (Clustered geography)

Instance #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Makespan (Dedicated)	60	62	59	64	62	54	53	56	64	52	60	62	56	62	56
Makespan (Flexible)	54	58	55	64	56	54	50	56	58	48	56	58	52	53	52
Instance #	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Makespan (Dedicated)	56	54	58	56	61	60	60	64	66	58	56	58	56	52	64
Makespan (Flexible)	56	52	55	52	57	55	60	60	58	58	56	56	54	52	61

Note. The figures in this table belong to problem instances with 10 customers, 2 trucks, and 1 drone, where the drone has a long endurance and is 50% faster than the trucks.

To further demonstrate the impact of flexible drones, Table 3 presents detailed route characteristics of solutions with dedicated and flexible drones, in uniform and clustered geographies. In both geographies, drone flexibility allows more customers to be served by the drone, which is faster than the trucks. Additionally, since the drone does not necessarily have to wait for the truck it was launched from, the average drone waiting time is lower (and drone utilization is higher) when the drones are flexible. Increased utilization and reduced waiting time of the drone naturally results in improved makespan values.

Table 3: Dedicated vs. flexible route characteristics

Geography	Flexible drones?	# Customers served by drone	Drone utilization	Truck travel time	Drone travel time	Truck wait time	Drone wait time
Uniform	Dedicated	3.07	62.04%	115.87	38.17	6.43	6.50
	Flexible	3.43	70.72%	113.8	43.10	6.60	5.97
Clustered	Dedicated	2.87	57.03%	110.40	33.23	4.70	8.97
	Flexible	3.80	77.43%	106.67	42.93	3.83	5.83

Note. Each figure in this table reflects an average of 30 problem instances.

4.3 Performance Analysis of Dynamic Discretization Discovery on VRPFD

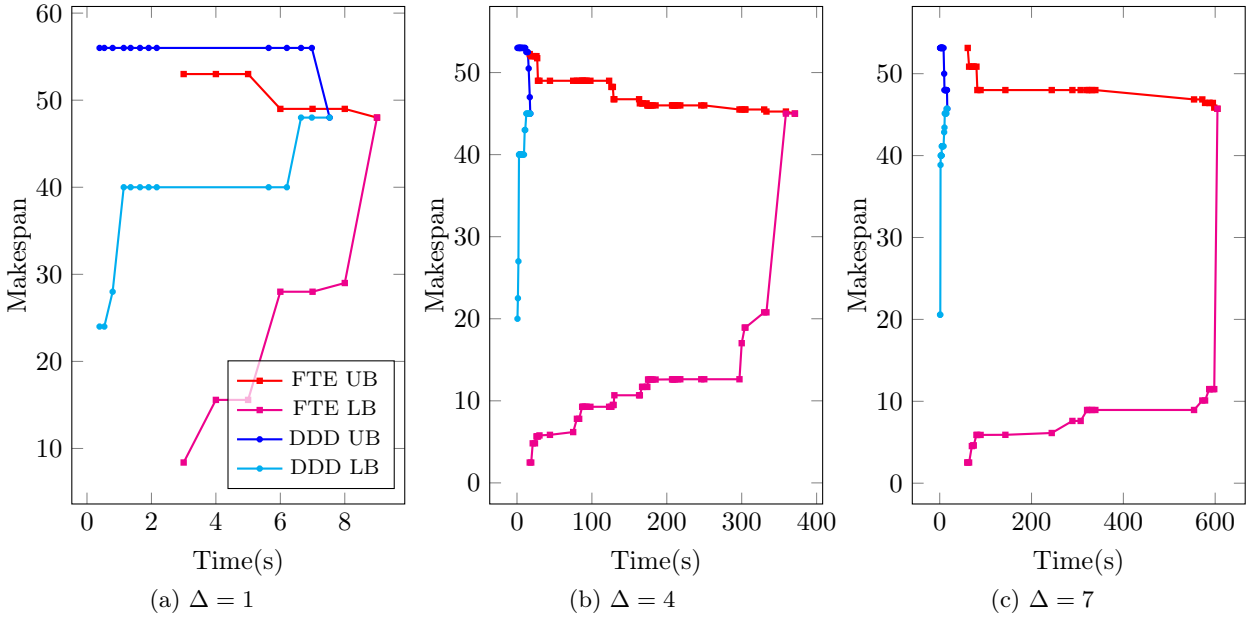
In this section, we provide detailed computational results to analyze the performance of DDD in solving VRPFD instances. Section 4.3.1 focuses on convergence behavior of DDD in comparison with the solution of the fully time-expanded formulation using Gurobi. In Section 4.3.2, we examine the network sizes of partially time-expanded networks observed in DDD, in contrast with the sizes of the fully time-expanded networks. Finally, Section 4.3.3 expands on the scalability of DDD with increasing time granularity, and presents computational evidence. In all performance comparisons, solving the fully time-expanded (FTE) formulation with Gurobi is used as a benchmark.

4.3.1 Convergence

To analyze convergence behavior of DDD in comparison to that of solving the FTE formulation, a total of 1890 problem instances with varying geographies, number of drones, drone/truck speed ratios, drone endurance ranges, and time granularity values are considered. Convergence behavior of both methods is observed over all problem instances. In the interest of space, this section presents the detailed convergence behavior of three representative instances (in Figures 3-5) under three time granularity scenarios, and provides general insights.

By design, DDD strives to reach the optimal solution by progressively improving the lower bounds, while keeping the partially time-expanded networks as small as possible. Thus, it is particularly promising in terms of lower bound quality and scalability. In all three representative problem instances presented in this section (see Figures 3-5), the benefits provided by DDD in lower bound quality and scalability are clearly evidenced. Note also that DDD attains the largest lower bound improvements in early iterations.

Figure 3 demonstrates the progression of lower and upper bounds in DDD and in solving the FTE formulation on a representative problem instance with time granularity values of 1, 4, and 7. It can be observed that DDD is able to find the optimal solution faster, regardless of time granularity. Closer inspection of the scalability of both methods reveals that DDD provides a clear advantage against solving the FTE formulation in finer time granularity values. The solution time of DDD varies between 7 and 11 seconds as time granularity increases, whereas the solution time of the FTE formulation varies between 9 and 605 seconds.

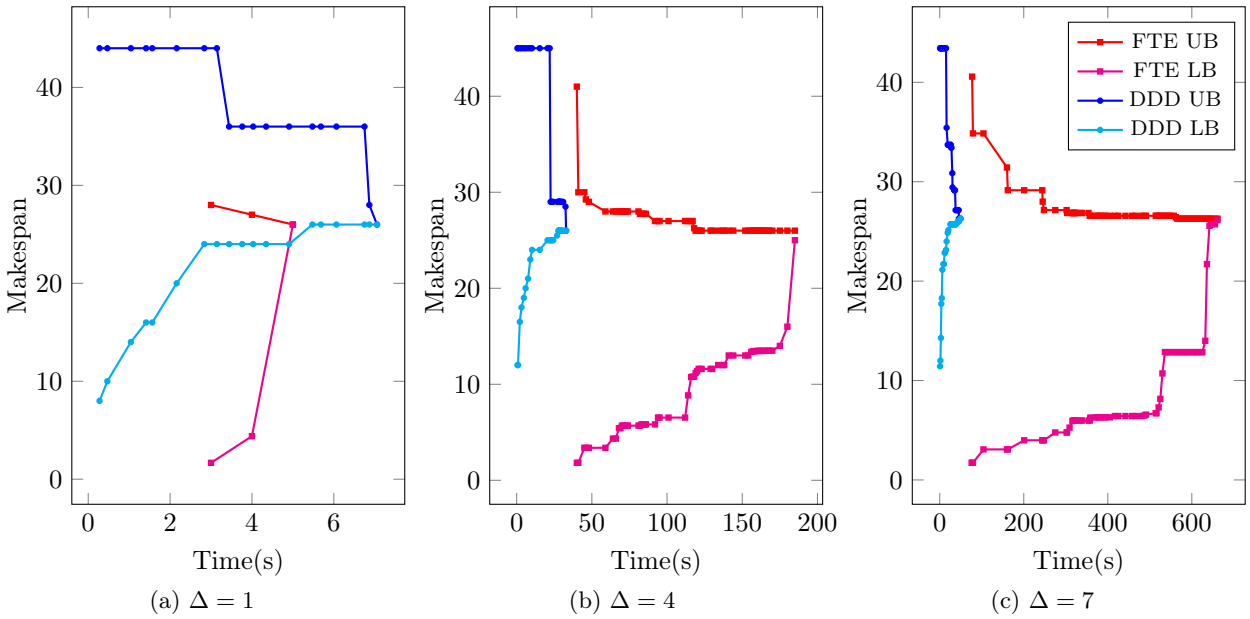


(a) $\Delta = 1$ (b) $\Delta = 4$ (c) $\Delta = 7$

Note. The instance has 10 customers, 2 trucks, 3 drones, where the drones have short endurance and are twice as fast as trucks.

Figure 3: Progression of lower and upper bounds (DDD vs. FTE formulation), representative instance 1

In some instances, solving the FTE formulation is more effective than using DDD when time granularity is very coarse. Figure 4 exemplifies such a case. It can be observed that when time granularity is 1 (see Figure 4a), DDD takes longer to converge to the optimal solution than the FTE formulation. However, with increasing time granularity, it becomes impractical to solve the FTE formulation due to the steep increase in the number of arcs in the underlying network. Figures 4b and 4c demonstrate that solving the FTE model is prohibitively slow with time granularity values of 4 and 7, and that DDD remains relatively unaffected by this increase in time granularity. The solution time of DDD increases from 7 to 49 seconds between the coarsest and finest time granularity values, whereas the solution time of the FTE formulation increases from 5 to 661 seconds.

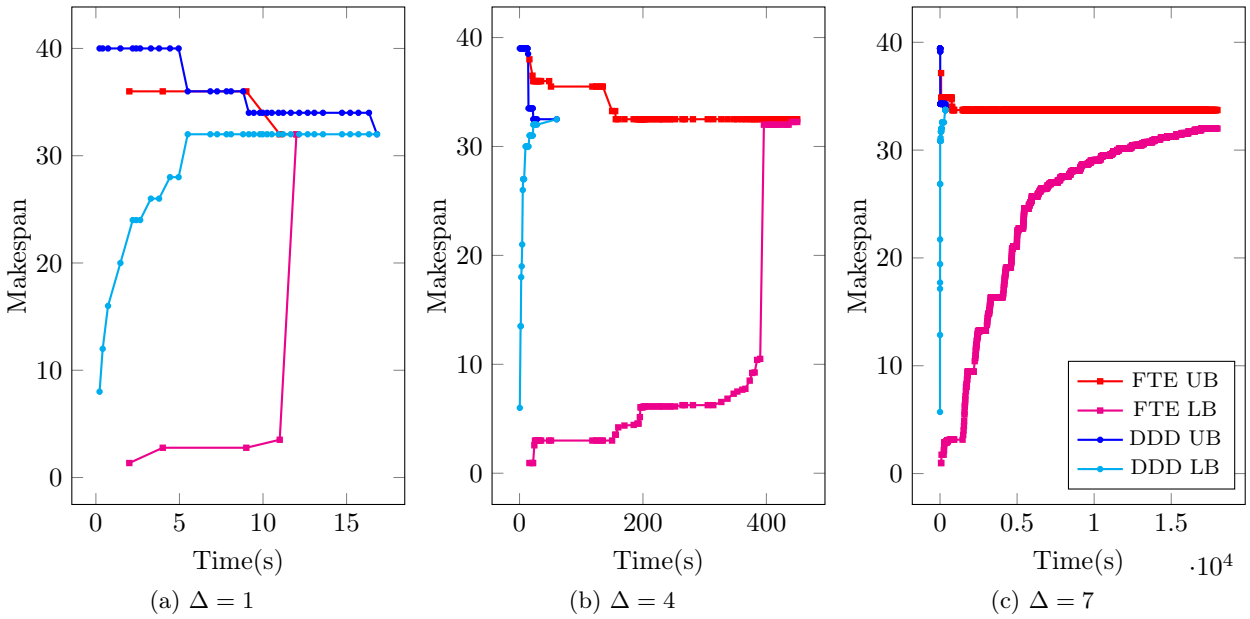


(a) $\Delta = 1$ (b) $\Delta = 4$ (c) $\Delta = 7$

Note. The instance has 10 customers, 2 trucks, 2 drones, where the drones have long endurance and are twice as fast as trucks.

Figure 4: Progression of lower and upper bounds (DDD vs. FTE formulation), representative instance 2

Figure 5 presents the bound progression on a problem instance where the FTE formulation with the finest time granularity could not be solved to optimality within the given time limit of 5 hours (even though FTE formulation was solved faster with the coarsest granularity). FTE formulation was not solved to optimality within the time limit in 6 out of the 1890 problem instances, whereas DDD solved all problem instances to optimality with a maximum solution time of 1083 seconds. Average scalability of DDD is discussed in Section 4.3.3.



(a) $\Delta = 1$ (b) $\Delta = 4$ (c) $\Delta = 7$

Note. The instance has 10 customers, 2 trucks, 1 drone, where the drone has medium endurance and is twice as fast as trucks.

Figure 5: Progression of lower and upper bounds (DDD vs. FTE formulation), representative instance 3

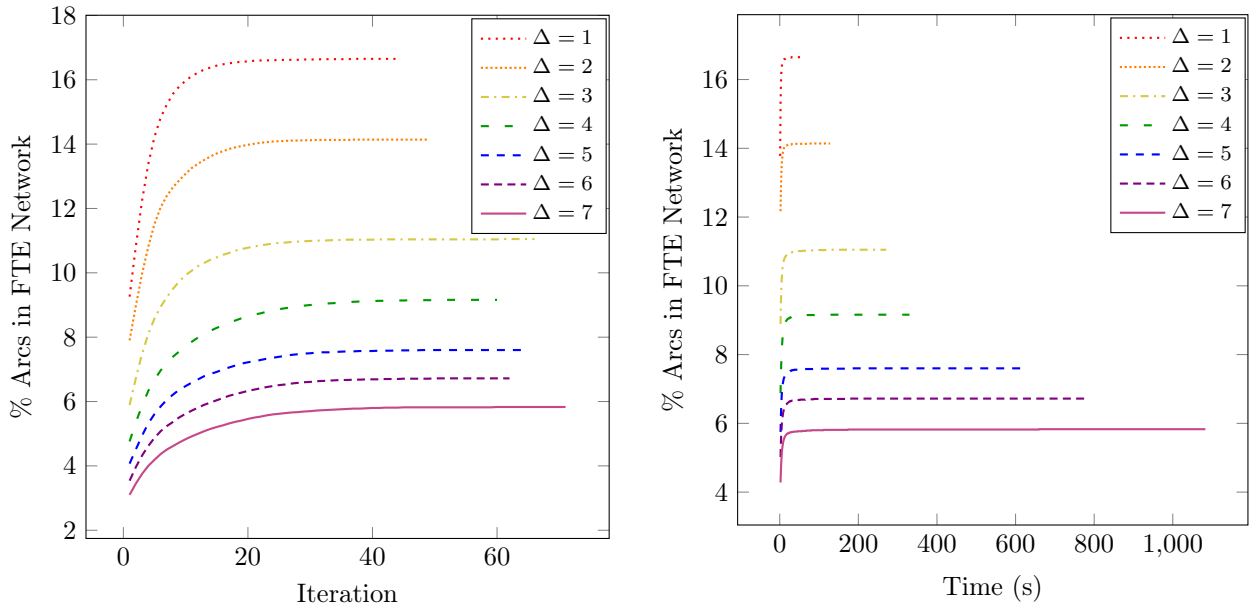
When we inspect the progression of the upper and lower bounds more closely, we observe that in general, DDD provides better lower bounds, and the FTE formulation provides better upper bounds. When solving the FTE formulation, significant computation time goes into proving the optimality of the incumbent solution in the majority of instances. A clear example of this can be seen in Figure 5. However, it is noteworthy that there

are also numerous problem instances where the incumbent solution keeps improving throughout the procedure, when solving the FTE model (see Figure 3).

4.3.2 Network Size

By design, DDD starts with a very small partially time-expanded (PTE) network. In this section, we demonstrate that the PTE networks remain small, especially in comparison with the fully time-expanded networks, throughout the iterations of the algorithm.

As can be seen in Figure 6, on average, the number of arcs in the PTE network does not exceed 16.6% and 5.8% of the number of arcs in the FTE network for the coarsest and finest time granularity values, respectively. Figures 6a and 6b present the proportion of the PTE network size to the FTE network size, averaged over all problem instances used in Section 4.3.1 for the specified granularity values. It can be clearly observed that as the algorithm progresses, the number of arcs in the PTE network stabilizes after increasing in the early iterations. By keeping the network sizes manageable, DDD tackles much smaller models than that of the FTE formulation, and therefore is a demonstrably more efficient solution method.



(a) Average relative network size by iteration

(b) Average relative network size in time

Figure 6: Average partially time-expanded network size relative to fully time-expanded network size ($|\bar{\mathcal{A}}|/|\mathcal{A}|$)

Figure 7 presents the average number of arcs in the last PTE network of DDD and the FTE network for various time granularity scenarios. We note that PTE network size increases only moderately (and at a decreasing rate) with increasing time granularity, whereas the FTE network size increases steadily. The ratio of number of arcs in the finest time-expanded networks to that in the coarsest is 1.88 for DDD, and 6.05 for the FTE formulation. This evidence of PTE networks of DDD remaining relatively unaffected by increased time granularity demonstrates the potential of DDD in solving VRPFD instances with even finer time granularity, and therefore in solving the continuous-time VRPFD.

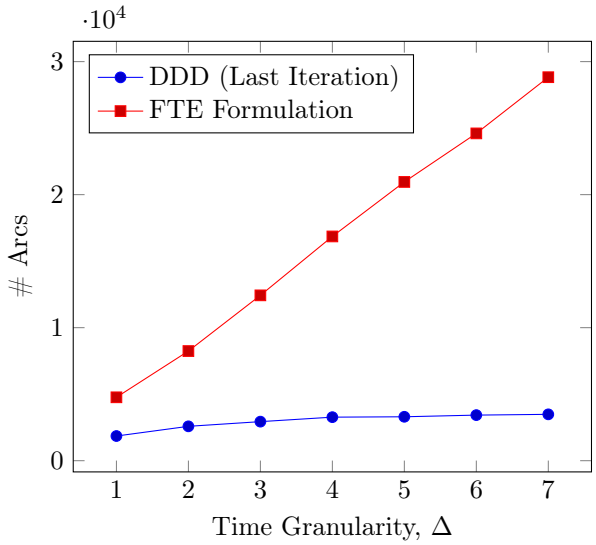
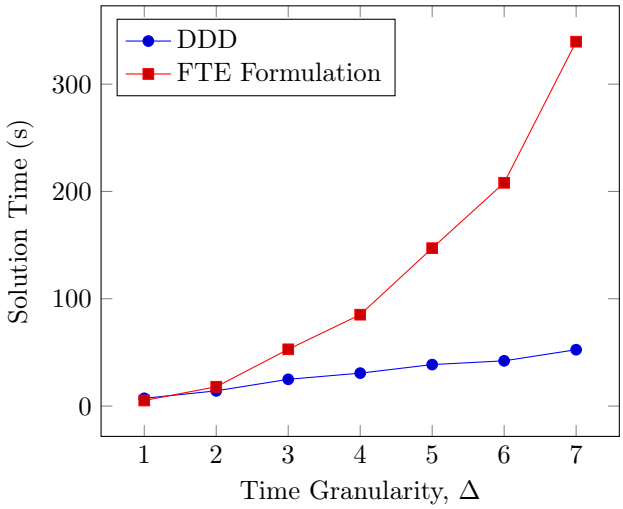


Figure 7: Average number of arcs on FTE network vs. the final PTE network in DDD

4.3.3 Scalability

Solving $VRPFD$ on partially time-expanded networks that are much smaller than the fully time-expanded network allows DDD to terminate with the optimal solution in a shorter time than solving the FTE formulation, when fine time granularity is of interest. In the vast majority of the problem instances, DDD solution time is minimally affected by increasing time granularity, while the time it takes to solve the FTE formulation increases at an increasing rate. The average behavior of solution time can be observed in Figure 8, where the averages are computed using all problem instances mentioned in Section 4.3.1.



Note. 6 instances (1, 2, and 3 instances for granularity values of 5, 6, and 7, respectively) were not solved to optimality within the 5-hour time limit with the FTE formulation.

Figure 8: Average solution time (DDD vs. FTE formulation)

Solution times are further examined by grouping the problem instances by certain problem characteristics. Figures 10, 11, and 12 of Appendix B present average solution times when instances are grouped by number of drones, drone endurance, and drone/truck speed ratio, respectively. It is notable that a moderate (steep) increase in DDD (FTE formulation) solution time with increasing time granularity, similar to that demonstrated for the average case (Figure 8), is observed in all instance classes.

Although a thorough analysis of the effects of problem characteristics on solution time is beyond the scope of this paper, some general insights can be drawn. It can be observed in Figures 10 - 12 that the FTE formulation solution times are longer on average for problem instances with fewer drones, shorter drone endurance, and higher drone/truck speed ratio. We also observe that fewer drones and higher drone/truck speed ratios (especially increasing the ratio from 1 to 1.5) resulted in a notable increase in DDD solution times. However, it is important to note that DDD solution times are much less sensitive to problem characteristics than the FTE formulation. The low variance of DDD solution time despite varying problem characteristics demonstrates the strength of this method in solving VRPFD effectively.

4.4 DDD on Larger VRPFD Instances

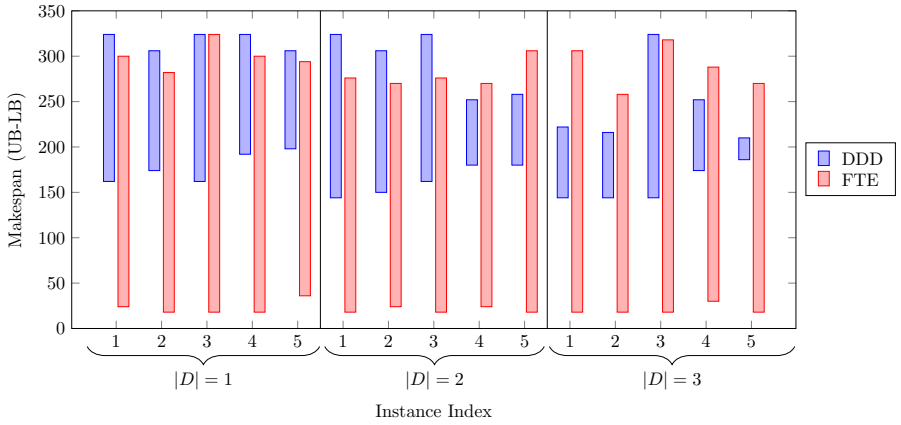
Obtaining feasible solutions and good primal or dual bounds may be of importance when solving the VRPFD to optimality is not of interest or practical. As illustrated on some example problem instances in Section 4.3.1, the progression of primal and dual bounds obtained at each iteration of DDD is promising, especially when compared to solving the FTE formulation. In this section, we attempt to solve larger problem instances with 25 customers with a 5-hour time limit and observe the primal and dual bounds obtained by DDD and the FTE formulation.

For this last set of computational experiments, we randomly generate 5 geographies with 25 customers, and solve VRPFD with 2 trucks and 1, 2, and 3 drones with $r = 1.5$ and medium endurance. To highlight the effect of increasing time granularity, we use time granularity values of 1, 3, and 6. The outcome of these experiments are presented in Figure 9. It is clearly observed in all problem instances with all time granularity scenarios that the lower bounds obtained in DDD are superior to those obtained when solving the FTE formulation. This is an expected result, since DDD obtains lower bounds in an intelligent way, by iteratively refining the PTE networks. As mentioned in Section 4.3.1, the quality of upper bounds obtained in DDD is often not better than those obtained when solving the FTE formulation, given that the solver can handle the size of the FTE network (fewer customers, and/or coarser time granularity scenarios) somewhat comfortably. We observe that effect here as well, when the time granularity is 1 (Figure 9a). However, for finer time granularity scenarios (Figures 9b and 9c), the FTE formulation struggles to find feasible solutions. In many problem instances, a feasible solution to the FTE formulation cannot be found in the given time limit; and even when feasible solutions are found, DDD is able to find better upper bounds thanks to the PTE networks being significantly smaller than the FTE network.

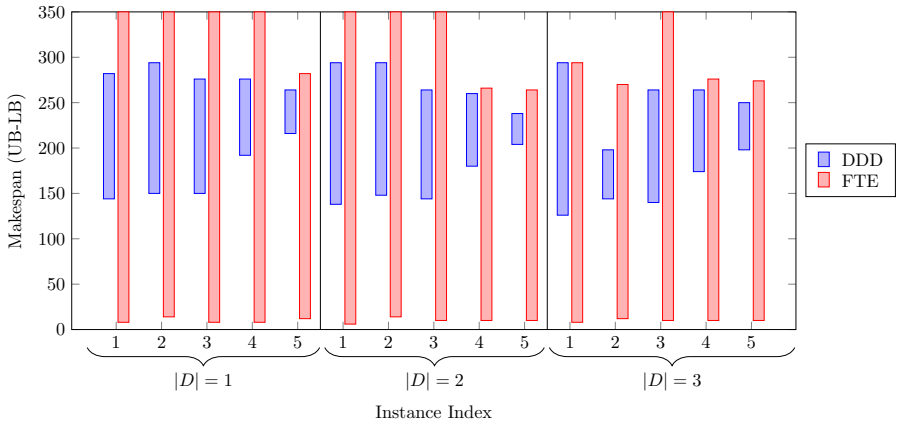
5 Conclusion

In this paper, we introduced the VRP with *flexible* drones (VRPFD), where the well-established assumption of *dedicated* drones, *e.g.* , that a drone must be retrieved by the truck from which it was launched, is relaxed. With computational experiments, we investigated the impact of relaxing the dedicated drone assumption and observed that drone flexibility reduces the average time the drones spend waiting to be retrieved by a truck, and therefore increases drone utilization. Since drones are assumed to be at least as fast as trucks, higher drone utilization results in a reduced makespan in most cases.

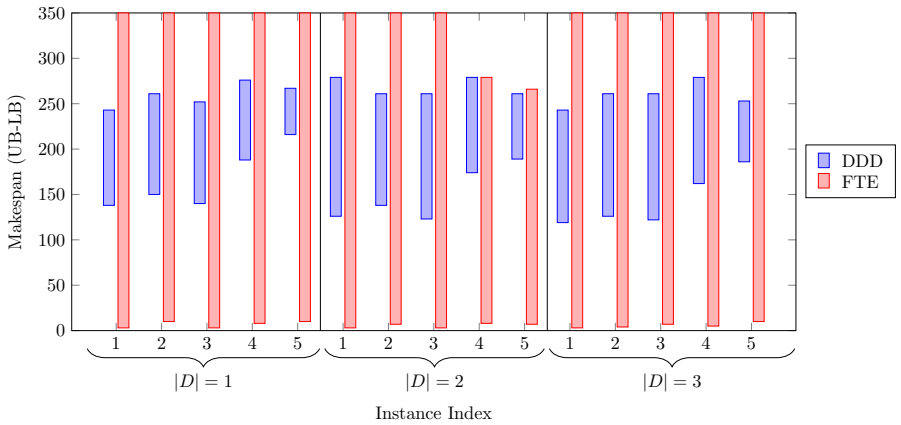
We presented a novel time-expanded network formulation for the VRPFD, and argued that directly solving this fully time-expanded formulation is not practical for larger problem instances with a large number of customers and/or fine time granularity values. To efficiently solve these larger problem instances, we proposed a *dynamic discretization discovery* (DDD) algorithm, which creates a partially time-expanded network and iteratively refines it until an optimal solution is found. We conducted an extensive computational study to demonstrate



(a) $\Delta = 1$



(b) $\Delta = 3$



(c) $\Delta = 6$

Note. The bars with an upper limit equal to the upper limit of the y-axis represent instances where the FTE formulation was not able to find a feasible solution, and therefore an upper bound.

Figure 9: Primal and dual bounds on large VRPFD instances after 5-hour time limit

that these partially time-expanded networks are effective in finding good lower bounds and are much smaller than their fully time-expanded counterparts. By exploiting the small size and the lower bounding properties of the partially time-expanded networks, DDD efficiently solves the VRPFD. To give a comprehensive view on the solution performance of DDD-VRPFD, we presented an in-depth analysis that addresses *(i)* the convergence behavior of DDD in various problem instances, *(ii)* the sizes of underlying partially and fully time-expanded networks, *(iii)* the scalability of DDD-VRPFD with increasing time granularity, and *(iv)* the quality of bounds obtained by DDD-VRPFD on larger problem instances which were not solved to optimality within a prespecified time limit.

References

Agatz, N., Bouman, P., and Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52(4):965–981.

Boland, N., Hewitt, M., Marshall, L., and Savelsbergh, M. (2017). The continuous-time service network design problem. *Operations research*, 65(5):1303–1321.

Bouman, P., Agatz, N., and Schmidt, M. (2018). Dynamic programming approaches for the traveling salesman problem with drone. *Networks*, 72(4):528–542.

Burns, S. (2017). Drone meets delivery truck. https://www.ups.com/us/es/services/knowledge-center/article.page?kid=cd18bdc2. [Accessed: 2019-12-10].

Chang, Y. S. and Lee, H. J. (2018). Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. *Expert Systems with Applications*, 104:307–317.

Coombs, C. (2019). Amazon's Ambitious Drone Delivery Plans Take Shape. https://www.thedailybeast.com/with-prime-air-amazon-wants-to-deliver-packages-in-30-minutes-or-less-via-drone. [Accessed: 2019-12-10].

Daknama, R. and Kraus, E. (2017). Vehicle routing with drones. *arXiv preprint arXiv:1705.06431*.

Dayarian, I., Savelsbergh, M., and Clarke, J.-P. (2020). Same-day delivery with drone resupply. *Transportation Science*.

de Freitas, J. C. and Penna, P. H. V. (2020). A variable neighborhood search for flying sidekick traveling salesman problem. *International Transactions in Operational Research*, 27(1):267–290.

Ferrandez, S. M., Harbison, T., Weber, T., Sturges, R., and Rich, R. (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management (JIEM)*, 9(2):374–388.

Ha, Q. M., Deville, Y., Pham, Q. D., and Hà, M. H. (2018). On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, 86:597–621.

Ham, A. M. (2018). Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming. *Transportation Research Part C: Emerging Technologies*, 91:1–14.

He, E., Boland, N., Nemhauser, G., and Savelsbergh, M. (2019). Dynamic discretization discovery algorithms for time-dependent shortest path problems. *Optimization online*, 7082.

Kitjacharoenchai, P., Min, B.-C., and Lee, S. (2019a). Two echelon vehicle routing problem with drones in last mile delivery. *International Journal of Production Economics*, page 107598.

Kitjacharoenchai, P., Ventresca, M., Moshref-Javadi, M., Lee, S., Tanchoco, J. M., and Brunese, P. A. (2019b). Multiple traveling salesman problem with drones: Mathematical model and heuristic approach. *Computers & Industrial Engineering*, 129:14–30.

Lagos, F., Boland, N., and Savelsbergh, M. (2020). The continuous-time inventory-routing problem. *Transportation Science*.

Mbiadou Saleu, R. G., Deroussi, L., Feillet, D., Grangeon, N., and Quilliot, A. (2018). An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem. *Networks*, 72(4):459–474.

Michel, A. H. (2017). Amazon's Drone Patents. https://dronecenter.bard.edu/files/2017/09/CSD-Amazons-Drone-Patents-1.pdf. [Accessed: 2019-12-10].

Mogg, T. (2018). Amazon's delivery drones could hitch rides on trucks to save power. https://www.digitaltrends.com/cool-tech/amazon-drone-patent-save-battery-power/. [Accessed: 2019-12-10].

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

Murray, C. C. and Raj, R. (2020). The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies*, 110:368–398.

Otto, A., Agatz, N., Campbell, J., Golden, B., and Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. *Networks*, 72(4):411–458.

Poikonen, S. and Golden, B. (2020). Multi-visit drone routing problem. *Computers & Operations Research*, 113:104802.

Poikonen, S., Golden, B., and Wasil, E. A. (2019). A branch-and-bound approach to the traveling salesman problem with a drone. *Informs Journal on Computing*, 31(2):335–346.

Poikonen, S., Wang, X., and Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks*, 70(1):34–43.

Ponza, A. (2016). Optimization of drone-assisted parcel delivery.

Radovic, M. (2019a). DRONEII: The Drone Delivery Market Map. https://dronelife.com/2019/11/07/droneii-the-drone-delivery-market-map/. [Accessed: 2019-12-10].

Radovic, M. (2019b). The Drone Delivery Report 2019. https://www.droneii.com/project/the-drone-delivery-report-2019. [Accessed: 2019-12-10].

Roberti, R. and Ruthmair, M. (2019). Exact methods for the traveling salesman problem with drone.

Sacramento, D., Pisinger, D., and Ropke, S. (2019). An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies*, 102:289–315.

Schermer, D., Moeini, M., and Wendt, O. (2019). A matheuristic for the vehicle routing problem with drones and its variants. *Transportation Research Part C: Emerging Technologies*, 106:166–204.

Tang, Z., van Hoeve, W.-J., and Shaw, P. (2019). A study on the traveling salesman problem with a drone. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 557–564. Springer.

Tu, P. A., Dat, N. T., and Dung, P. Q. (2018). Traveling salesman problem with multiple drones. In *Proceedings of the Ninth International Symposium on Information and Communication Technology*, pages 46–53.

Vu, D. M., Hewitt, M., Boland, N., and Savelsbergh, M. (2019). Dynamic discretization discovery for solving the time-dependent traveling salesman problem with time windows. *Transportation Science*.

Wang, D., Hu, P., Du, J., Zhou, P., Deng, T., and Hu, M. (2019a). Routing and scheduling for hybrid truck-drone collaborative parcel delivery with independent and truck-carried drones. *IEEE Internet of Things Journal*, 6(6):10483–10495.

Wang, K., Yuan, B., Zhao, M., and Lu, Y. (2019b). Cooperative route planning for the drone and truck in delivery services: A bi-objective optimisation approach. *Journal of the Operational Research Society*, pages 1–18.

Wang, X., Poikonen, S., and Golden, B. (2017). The vehicle routing problem with drones: Several worst-case results. *Optimization Letters*, 11(4):679–697.

Wang, Z. and Sheu, J.-B. (2019). Vehicle routing problem with drones. *Transportation research part B: methodological*, 122:350–364.

WORKHORSE (2019). HorseFly Autonomous: fully functional end to end drone delivery system. https://workhorse.com/horsefly.html. [Accessed: 2019-12-10].

Yoon, J. J. (2018). *The traveling salesman problem with multiple drones: an optimization model for last-mile delivery*. PhD thesis, Massachusetts Institute of Technology.

Appendix A MILP formulation for the VRP with dedicated drones

The VRP with dedicated drones can be formulated using the following decision variables:

- x_a^k = binary variable denoting whether vehicle $k \in K$ uses arc $a \in \mathcal{A} \setminus \mathcal{A}_d$
- y_a^k = number of drones dedicated to vehicle $k \in K$ using arc $a \in \mathcal{A}$
- u_i, v_i, z: same as before (as defined in the VRPFD formulation)

$$\min z \tag{35}$$

$$\text{s.t. } x^k(\delta^+(i,t)) - x^k(\delta^-(i,t)) = \begin{cases} 1 \text{ if } (i,t) = (s,0) \\ -1 \text{ if } (i,t) = (f,T) \\ 0 \text{ otherwise} \end{cases} \qquad (i,t) \in \mathcal{N} \setminus \mathcal{N}_k, k \in K, \tag{36}$$

$$\sum_{k \in K} y^k(\delta^+(s,0)) = |D|, \tag{37}$$

$$\sum_{k \in K} y^k(\delta^-(f,T)) = |D|, \tag{38}$$

$$y^k(\delta^+(i,t)) - y^k(\delta^-(i,t)) = 0, \qquad (i,t) \in \mathcal{N} \setminus \{(s,0),(f,T)\}, k \in K, \tag{39}$$

$$y_a^k \leq |D| x_a^k \qquad a \in A_k, k \in K, \tag{40}$$

$$y_a^k \leq x^k(\delta^-(i,t) \setminus \mathcal{A}_d) \qquad (i,t) \in \mathcal{N}^v, a \in \delta^+(i,t) \cap \mathcal{A}_d, k \in K, \tag{41}$$

$$\tau_a^d y_a^k + \tau_{a'}^d y_{a'}^k \leq r \qquad (i,t) \in \mathcal{N}_d, a \in \delta^-(i,t), a' \in \delta^+(i,t), k \in K, \tag{42}$$

$$u_i \leq \sum_{k \in K} x^k(\delta^-(\mathcal{N}_i^v) \cap \mathcal{A}_k) \leq |\mathcal{N}_k^i| u_i \qquad i \in N, \tag{43}$$

$$v_i \leq \sum_{k \in K} y^k(\delta^-(\mathcal{N}_d^i)) \leq |\mathcal{N}_d^i| v_i \qquad i \in N, \tag{44}$$

$$u_i + v_i \geq 1 \qquad i \in N, \tag{45}$$

$$z \geq \bar{t}_a x_a^k \qquad a \in \delta^-(f,T) \cap \mathcal{A}_k, k \in K, \tag{46}$$

$$z \geq \bar{t}_a y_a^k \qquad a \in \delta^-(f,T) \cap \mathcal{A}_d, k \in K, \tag{47}$$

$$\sum_{k \in K} y^k(\delta^-(i,t)) \leq 1 \qquad (i,t) \in \mathcal{N}_d, \tag{48}$$

$$\sum_{k \in K} x_a^k \leq 1 \qquad a \in A_k, \tag{49}$$

$$\sum_{k \in K} y_a^k \leq 1 \qquad a \in A_d, \tag{50}$$

$$x_a^k \in \{0,1\} \qquad a \in \mathcal{A}_k \cup \mathcal{A}_h, k \in K, \tag{51}$$

$$y_a^k \in \{0,1\} \qquad a \in \mathcal{A}_d, k \in K \tag{52}$$

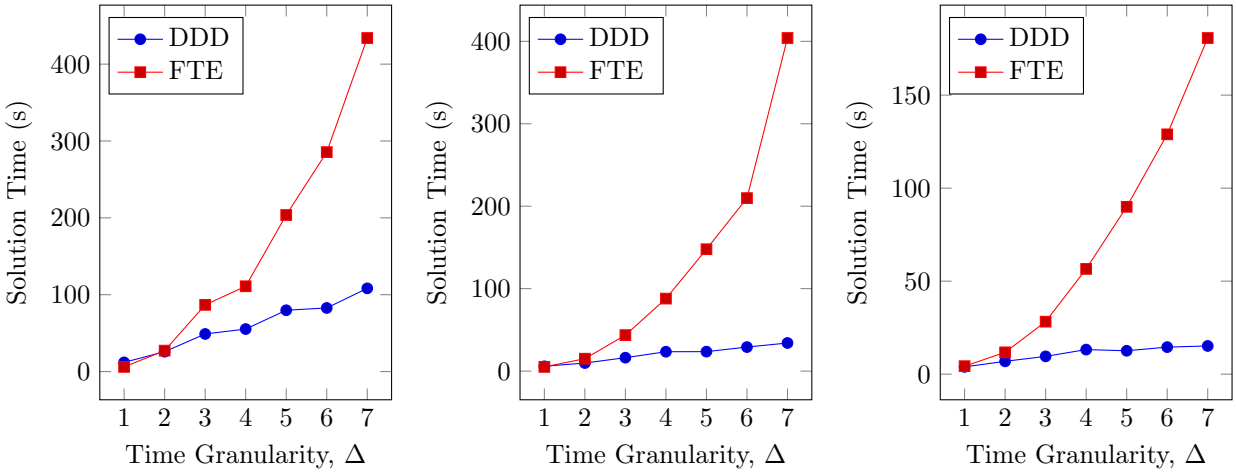
$$y_a^k \in \mathbb{Z}_+ \qquad a \in \mathcal{A}_k \cup \mathcal{A}_h, k \in K, \tag{53}$$

$$u_i \in \{0,1\} \qquad i \in N, \tag{54}$$

$$v_i \in \{0,1\} \qquad i \in N, \tag{55}$$

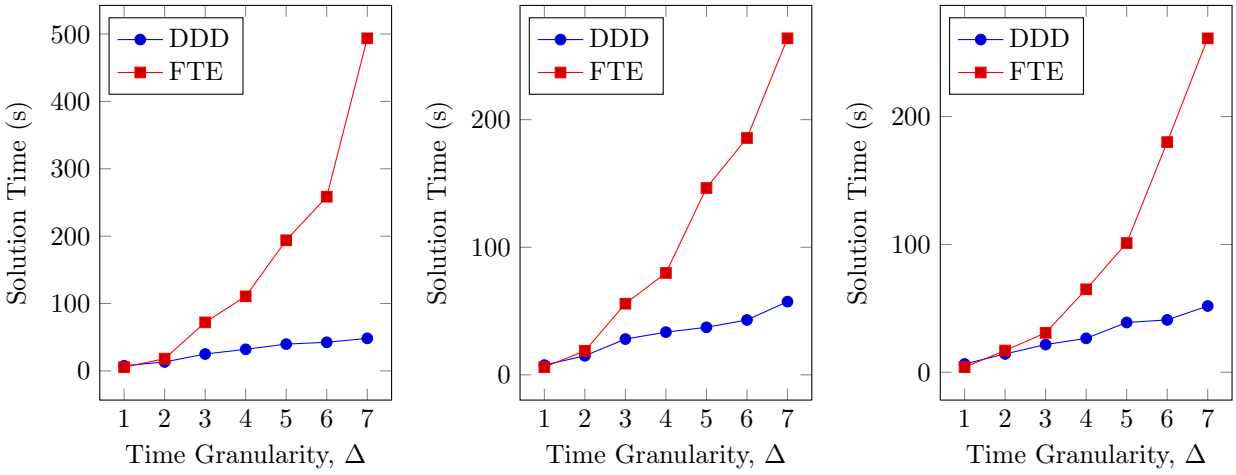
$$z \geq 0. \tag{56}$$

Appendix B Solution Time Details



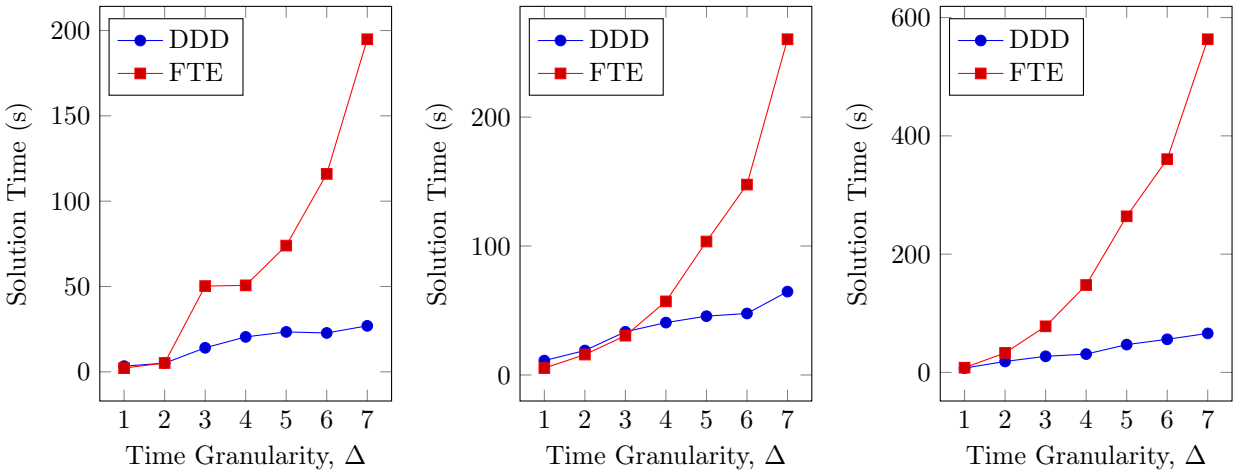
(a) $|D| = 1$ (b) $|D| = 2$ (c) $|D| = 3$

Figure 10: Average solution time, by number of drones



(a) Short endurance (b) Medium endurance (c) Long endurance

Figure 11: Average solution time, by drone endurance



(a) $\rho = 1$ (b) $\rho = 1.5$ (c) $\rho = 2$

Figure 12: Average solution time, by drone speed ratio