TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

A Variable Neighborhood Search Algorithm for Solving the Vehicle Routing Problem with Drones

Daniel Schermer, Mahdi Moeini, Oliver Wendt

Technical Report BISOR-02/2018

Daniel Schermer
Chair of Business Information Systems and Operations Research
Technische Universität Kaiserslautern, Germany
daniel.schermer@wiwi.uni-kl.de

Mahdi Moeini
Chair of Business Information Systems and Operations Research
Technische Universität Kaiserslautern, Germany
mahdi.moeini@wiwi.uni-kl.de

Oliver Wendt
Chair of Business Information Systems and Operations Research
Technische Universität Kaiserslautern, Germany
wendt@wiwi.uni-kl.de

Abstract

Drones have started to play an increasing role in logistic systems in both, academic research and practical context. In particular, drones have already been applied in various public and private service sectors including energy, agriculture, and emergency response. Recently, the Vehicle Routing Problem with Drones (VRPD) has been introduced as a variant of the Vehicle Routing Problem. In the case of the VRPD, a fleet of vehicles, each of them equipped with a set of drones, is tasked with delivering parcels to customers. The objective consists in designing feasible routes with minimal mission time. The drones may be launched from and retrieved by the vehicles and move at a velocity that might differ from the vehicle's speed. Furthermore, drones possess a limited flight endurance and small carrying capacity. The VRPD can be formulated as a Mixed Integer Linear Program (MILP) and, consequently, be solved by any standard MILP solver. With the aim of improving the performance of solvers, we introduce some sets of valid inequalities. Additionally, due to limited performance of the solvers in addressing large-scale instances, we address this issue by proposing an algorithm based on the well-known Variable Neighborhood Search (VNS) approach. In order to evaluate the performance of the introduced algorithm as well as the solver in solving the VRPD instances, we carried out extensive computational experiments.

Keywords: Vehicle Routing Problem; Drones; Last-mile Delivery; Valid Inequalities; Heuristics; Metaheuristics; Variable Neighborhood Search

1 Introduction

In recent years, drones have started to play in increasing role in logistic systems in both, academic research and practical context. Several companies in the logistic industry, such as Amazon Inc., Deutsche Post AG, UPS Inc. and the DPDgroup are actively investigating the potentials of drones in delivery applications (see, e.g., (DHL Trend Research 2014, 2016)). Furthermore, drones have been successfully applied in many public and private sectors including energy, agriculture and forestry, environmental protection, and emergency response (see (Carlsson and Song 2017)), and references therein). Most recently, several problems have been proposed in the academic literature that integrate drones into last-mile delivery, in an effort to lower costs and reduce delivery times (see, e.g., (Murray and Chu 2015, Wang et al. 2016, Agatz et al. 2018)). Drones can generally move faster between two locations than vehicles due to not being restricted to the road network or congestion. In addition, drones and their payload are far more lightweight than vehicles, which causes drones to consume less energy for the movement between two points. However, a drone's capacity is typically limited to just one or few parcels. Moreover, as drones rely on comparatively small batteries for powering their flight, their range of operation is more restricted compared to a vehicle powered by a fossil fuel or a much larger battery. Consequently, due to the complementary nature of vehicles and drones, they might be useful for last-mile delivery purposes.

In the context of the last-mile logistics, several drone-based optimization problems have recently been introduced in the literature. The *Vehicle Routing Problem with Drones* (VRPD), as proposed by Wang et al. (2016), is one of the representative problems and integrates drones into the classical *Vehicle Routing Problem* (VRP). In the case of the VRPD, a fleet of vehicles, each of them equipped with a set of drones, is tasked with delivering parcels to customers. As an extension of the VRP, the VRPD is NP-hard and, consequently, quite difficult to solve for large instances.

In this paper, we address the Vehicle Routing Problem with Drones and propose two approaches in order to solve it. The contributions of our work are threefold:
(1) We introduce a *mixed integer linear program* (MILP) formulation for the VRPD. The model might be solved by any standard MILP solver, e.g., Gurobi Optimizer and IBM CPLEX. They are able to handle small-sized instances of the problem.
(2) In order to improve the performance of MILP solvers in solving VRPD instances, we derive some sets of valid inequalities. These valid inequalities impose lower and upper bounds on the optimal value of a VRPD instance. We show that our proposed valid inequalities have a significant impact on the solver's performance in finding an optimal solution to small-scale instances with up to 12 nodes.
(3) The VRPD, as a generalization of the VRP, is an NP-hard optimization problem. Consequently, using the MILP formulation to obtain optima in a reasonable amount of time is only possible on small-sized instances. Therefore, in order to address large-scale instances of the VRPD, we propose a metaheuristic that is based on *Variable Neighborhood Search* (VNS). In particular, as components of our method, we propose two drone insertion operators. We show that our method is computationally effective and provides good solutions consistently.
In any case, through our numerical study, we show that the application of drones can significantly reduce the mission time, thus making them a valuable prospect for future last-mile delivery applications.

The remainder of this paper is organized as follows. We begin in Section 2 by providing a brief overview of the existing academic literature that is related to the application of drones in last-mile delivery. Then, Section 3 introduces the notation, mathematical model, and the proposed valid inequalities for the VRPD. In Section 4, we propose a metaheuristic approach that is based on *Variable Neighborhood Decomposition Search* as well as two drone insertion operators: *greedy insertion* and *savings insertion*. Detailed results on the numerical studies that were performed using both, the MILP formulation as well as the metaheuristic approach, are presented in Section 5. Finally, we draw a conclusion on this work and on future research implications in Section 6.

2 Related Works

In this section, we provide a short overview on the drone-related optimization problems. A more extensive literature review might be found in (Otto et al. 2018).

In Murray and Chu (2015), the authors introduced two new NP-hard problems that are related to delivery applications, where a vehicle (or truck) is working in tandem with a drone.

- In the *flying sidekick traveling salesman problem* (FSTSP), a set of customers is given, each of whom must be visited exactly once by either a drone or a vehicle. The drone travels along with a vehicle, both of them start from a common *distribution center* (DC) and must return to the same DC at the end of the tour. At any customer location (or at the DC), the drone may be launched from the vehicle, starting a *sortie*, which is described as follows: the drone begins its flight to make its way to serve a customer and will then be retrieved by the vehicle at a later customer location (or at the DC).

- In the *parallel drone scheduling TSP* (PDSTSP), it is assumed that the DC is in close proximity to most of the customers. In this case, it might be beneficial to let the drone make its deliveries independently from the vehicle, while the vehicle serves customers that are located far away from the DC. Whereas the vehicle continues on a predefined tour, the drone will continuously return to the DC to pick up a new parcel after each delivery.

Both problems, i.e., the FSTSP and PDSTSP, share the objective of minimizing the mission time. However, in the case of the FSTSP synchronization between the vehicle and the drone is required whereas no synchronization is required in the PDSTSP as the drone and the vehicle make their deliveries independently from each other. Murray and Chu (2015) introduced MILP formulations as well as two heuristic methods for solving the FSTSP and PDSTSP. The authors conclude that, due to the NP-hard nature of the proposed problems, only small-sized instances can be solved to optimality within a reasonable amount of time. Finally, given a problem instance, Murray and Chu (2015) highlight that it is difficult to assess a priori whether treating a problem as the FSTSP or the PDSTSP will yield better results with regards to the mission time.

The *Traveling Salesman Problem with Drone* (TSP-D), is another problem that shares many similarities with the FSTSP introduced by Agatz et al. (2018)[1]. The authors propose a MILP formulation as well as several heuristics (based on *route first, cluster second*) for solving the TSP-D. Additionally, Agatz et al. provide a worst-case approximation guarantee on their heuristic. Furthermore, they introduce a dynamic programming algorithm that is able to find the best assignment of vehicle and drone deliveries for a given delivery sequence. Agatz et al. performed a numerical study to assess the performance of their heuristics on various problem instances. The authors suggest to extend their model in order to take into account multiple vehicles (or drones) as well as to add different possibilities for recharging the drones.

Based on dynamic programming, Bouman et al. (2017) provide another exact solution method for the TSP-D. The authors highlight that their method is able to find the optimal solution for problem instances with 10 vertices much faster compared to the integer programming approach presented by Agatz et al. (2018). In particular, Bouman et al. (2017) also perform an in-depth analysis on the impact of adding an additional constraint that restricts the number of drone sorties. While the authors highlight that this constraint improves the time required by their method to compute the optimal solution, limiting the number of drone sorties may prevent the algorithm from finding the globally optimal solution of the TSP-D.

Yurek and Ozmutlu (2018) provide an iterative optimization algorithm for the TSP-D. The approach is based on decomposition of the problem into two components: (i) finding a tour and (ii) assigning drone operations to a given tour. First, they generate all possible solutions to the *Traveling Salesman Problem* (TSP). They keep track of the *global upper bound* (GUB) in their solution set and propose an iterative way of solving the drone assignment problem through a mathematical model that minimizes the waiting time of the truck at potential retrieval vertices. Yurek and Ozmutlu (2018) compare their results to (Murray and Chu 2015, Agatz et al. 2018) and report an improvement, by

[1] A closely related work was originally published in 2015 by the same authors.

being able to solve instances with 10 to 12 customer locations in shorter computation time. However, the authors believe that, due to the concept of the GUB, their approach is not well-suited to clustered problem instances.

Carlsson and Song (2017) worked on a close variant of the FSTSP. One of their key findings is the hypothesis that the potential benefit of using a drone in tandem with a vehicle is proportional to the square root of the relative velocity between the vehicle and the drone. Furthermore, they highlight the dependency of the potential benefit on the instance features. The authors propose a heuristic method based on convex optimization and use generated instances (with up to 500 vertices) as well as practical data (with up to 100 vertices) in order to support their hypothesis.

Ha et al. (2018) employed two heuristics to solve the FSTSP: *route first, cluster second* and *cluster first, route second*. The latter heuristic performed better according to the numerical results. The authors believe that if a drone travels at a velocity similar to the vehicle, then this can significantly improve the mission time. Furthermore, they suggest integrating local search procedures to refine the quality of their method.

Mathew et al. (2015) introduced the *Heterogeneous Delivery Problem* (HDP) which shares most features of the TSP-D and FSTSP. The authors propose a solution approach by reducing the HDP to the *Generalized Traveling Salesman Problem* (GTSP). Further, they suggest splitting the HDP into two traceable sub-problems: first they compute an optimal vehicle routing; then, convex optimization is applied to compute the specific deployment points for the drone. Mathew et al. (2015) also propose a variant of the HDP called *Multiple Warehouse Delivery Problem* (MWDP), where one drone serves customers from multiple warehouses. In this case, the authors suggest again their method for solving the MWDP.

Boone et al. (2015) highlighted the potential of applying the *Multiple Traveling Salesman Problem* (MTSP) to a drone-based logistic system. In the MTSP, multiple vehicles must serve customers with the objective of minimizing the mission time. Boone et al. suggest a variation of the K-means algorithm (MacQueen 1967) for creating clusters, then perform an initial nearest-neighbor heuristic and finally use a 2-opt algorithm for improving the routing.

Ferrandez et al. (2016) combined *K-means*, for clustering purposes, and a *Genetic Algorithm* with various *mutation* operators for finding valid tours to solve a variant of the FSTSP. In their variant, the authors allow multiple drones per vehicle. The customers are first clustered into k sets through K-means algorithm (MacQueen 1967) and then assigned if they are served by vehicle or drone. Ferrandez et al. (2016) also investigated the impact of various relative velocities and identified that the drones should be at least twice as fast as the vehicles to allow for a significant improvement in the mission time. Furthermore, the authors highlight the added benefits of employing multiple drones (per vehicle) in energy consumption.

Wang et al. (2016) introduced the VRPD as a generalization of the VRP. In the VRPD a fleet of vehicles, each vehicle equipped with a given number of drones, is tasked with delivering parcels to customers. The drones may be launched from the vehicle at the depot or at any customer location and might be retrieved by the vehicle at a different customer location (or at the depot, concluding its tour). When a drone is launched, it can serve exactly one customer before returning to the vehicle (or to the depot). The objective is to minimize the mission time, i.e., the time required to serve all customers such that the fleet has returned to the depot at the end of the mission. Wang et al. introduced several upper bounds on the amount of time that can be saved by employing drones compared to the classic VRP. The upper bounds are obtained by investigating the structure of optimal solutions and depend on the relative velocity of the drones compared to the vehicles and the number of drones per vehicle. Poikonen et al. (2017) refined the work of Wang et al. (2016) by considering the impact on the previously introduced bounds when integrating limited battery life, different distance metrics, and operational expenditures of deploying drones and vehicles in the objective function. Furthermore, Poikonen et al. (2017) proposed extending the VRPD model similar to the *close-enough traveling salesman problem* (CETSP). The authors suggest the possibility of launching and retrieving drones from arbitrary locations instead of being restricted to customer locations. They conclude their work by proposing that a computational heuristics and benchmark instances should be developed as a next step in research on the VRPD.

Daknama and Kraus (2017) introduced the *Vehicle Routing with Drones* (VRD) problem that shares most of the features of the VRPD (Wang et al. 2016). However, Daknama and Kraus allow a drone to be launched and collected by different vehicles. The authors propose a heuristic that consists of two nested local search procedures to solve the VRD problem. Furthermore, Daknama and Kraus conclude that the use of drones can significantly reduce the mission time. Finally, they claim that an increase in the number of drones generally leads to an improvement in the solution, however, with a decrease in the marginal benefit of each additional drone.

Pugliese, L.D.P. and Guerriero, F. (2017) introduced the *vehicle routing problem with drones and time windows* (VRPDTW), where the objective consists in minimizing the cost required to serve all customers, while respecting the time-windows restrictions. The authors provide a mathematical model for the VRPDTW and use a commercial solver for solving the problem on randomly generated instances with 5 and 10 customers. Pugliese and Guerriero conclude that the benefit of drones for last-mile delivery strongly depends on the marginal cost per mile of both drones and vehicles.

Dorling et al. (2017) propose the *Drone Delivery Problem* (DDP), where drones visit a DC several times to pick up parcels and deliver them to customers. This problem can be considered a generalization of the MTSP proposed by Boone et al. (2015). The authors derive two optimization problems: the *Minimum-Cost-DDP* (MC-DDP) and the *Minimum-Time-DDP* (MT-DDP), for minimizing the cost and delivery time, respectively. Dorling et al. (2017) propose solving both problems by a MILP solver as well as a *Simulated Annealing* (SA) algorithm. Finally, they verify if their SA implementation can reach near-optimal results for small-sized problems of the DDP.

Ulmer and Thomas (2017) propose the *same-day delivery routing problems with heterogeneous fleets* (SDDPHF), where drones and vehicles serve customers without a need for synchronization. The customers are not known in advance and, consequently, dynamic routing of the heterogeneous fleet based on the incoming requests is required. In fact, the previously introduced PDSTSP of Murray and Chu (2015) can be viewed as a relaxed version of the SDDPHF in which all customers are known a priori. Ulmer and Thomas (2017) use a newly introduced approach that they call *geographic districting* in order to partition the customers into sets that should be served by the drone and vehicle, respectively. Through computational experiments, the authors can show that the combination of vehicles and drones can reduce the delivery costs significantly.

Jiang et al. (2017) applied drone logistics to the *Vehicle Routing Problem with Time Windows* (VRPTW) and derived a MILP model. Their model of the VRPTW consists of assigning a swarm of drones to serve customers within predefined time frames. The authors use an adapted version of *Particle Swarm Optimization* for drone routing and conclude that the heuristic is well suited for solving this version of the VRPTW.

3 The Vehicle Routing Problem with Drones

In this paper, we are interested in studying the VRPD as proposed by (Wang et al. 2016, Poikonen et al. 2017). We recall that the VRPD asks for routing a fleet of vehicles, each vehicle equipped with a set of drones, and looks for minimizing the mission time, i.e., the time required to serve all customers using the vehicles and the drones such that, by the end of the mission, all vehicles and drones must be at the depot. According to the classification of Toth and Vigo (2002), we might classify the VRPD as a *Distance-Constrained Capacitated Vehicle Routing Problem* (DCVRP) with a set of *heterogeneous* vehicles and additional *synchronization* constraints:

- The *Capacitated Vehicle Routing Problem* (CVRP) deals with heterogeneous fleet of vehicles with a limited carrying capacity. In the case of the VRPD, the drones are limited to one parcel per delivery, whereas the vehicles have a higher carrying capacity.
- A variation of the VRP, where the capacity is unlimited but a distance constraint is imposed is called the *Distance-Constrained Vehicle Routing Problem* (DVRP). In the case of the VRPD, due to limited battery capacity, the drones might only travel a restricted distance.

The combination of the CVRP and DVRP, where both capacity and distance constraints are imposed, leads to the DCVRP. The heterogeneous fleet consists of drones and vehicles, to which different

constraints apply (e.g., different carrying capacities, distance constraints, possibly distance metrics, and velocities).

In this section, we formulate the VRPD as Mixed Integer Linear programming (MILP) model. We begin by providing the notation that we need for the formulation of the VRPD.

3.1 Notation and Mathematical Model

Suppose that a set of customer locations, each with an equal demand, and a fleet of homogeneous vehicles, each carrying the same amount of identical drones, are given. In the VRPD, we look for minimizing the mission time required to serve all customers using the vehicles and the drones such that, by the end of the mission, all vehicles and drones must be at the depot (Wang et al. 2016, Poikonen et al. 2017). Furthermore, we make the following assumptions (Murray and Chu 2015, Wang et al. 2016, Poikonen et al. 2017):

- A drone can serve exactly one customer per operation.
- A drone has a limited endurance of $\mathcal{E}$ distance units. After returning to the vehicle, the endurance of the drone is recharged instantaneously.
- The vehicles and drone do not necessarily follow the same distance metric. While the vehicles are restricted to the road network restrictions, the drones might be able to use a different trajectory for traveling between locations.
- At a customer location, the service time required to deliver a parcel by vehicle or drone is assumed to be negligible. In addition, the service time required to launch and retrieve a drone is assumed to be negligible.
- Without loss of generality, the average velocity of each vehicle is assumed to be equal to 1 and the relative velocity of each drone is assumed to be α times the velocity of the vehicle. Due to the absence of congestion or necessary stops (e.g., traffic lights, stop signs) in the route of a drone, we can assume that $\alpha \geq 1$.
- Drones may only be dispatched and picked up from vertices, i.e., at the depot or any other customer location. Furthermore, a drone may not be picked up at the same vertex from which it was launched. If a drone arrives at a retrieval location before its corresponding vehicle, then the drone must wait for the vehicle to arrive. Moreover, a drone must always return to the same vehicle from which it was launched.
- We assume that when a drone is launched, then its delivery will be successful. In other words, we consider a risk-free environment.

Inspired by the work of Pugliese, L.D.P. and Guerriero, F. (2017), we use the following notation in order to formulate the VRPD.

Assume that a complete and symmetric graph $G = (V, E)$ is given, where V is the set of vertices (or nodes) and E is the set of edges. The set V contains n nodes associated with the customers, named $V_N = \{1, \ldots, n\}$ and (in order to simplify the notation and the mathematical formulation) two extra nodes 0 and $n+1$ that both represent the same depot location at the start and end of each tour, respectively. Thus, $V = \{0, 1, \ldots, n, n+1\}$, where $0 \equiv n+1$.

We refer to $V_L = V \setminus \{n+1\}$ as the subset of nodes in V from which drones may be launched. Furthermore, $V_R = V \setminus \{0\}$ denotes the subset of nodes where the drones may be retrieved.

By the parameters d_{ij} and $\overline{d}_{ij}$ we define the distance required to travel from node i to node j by vehicle and drone, respectively. As G is assumed to be symmetric, $d_{ij} = d_{ji}$ and $\overline{d}_{ij} = \overline{d}_{ji} : \forall i, j \in V$. Furthermore, by definition, $d_{ii} = \overline{d}_{ii} = 0 : \forall i \in V$. Additionally, as the drone may not be limited to the road network, without loss of generality, we may assume that $\overline{d}_{ij} \leq d_{ij} : \forall i, j \in V$.

Using the parameters $v = 1$ and $\overline{v} = \alpha \cdot v$, we define the velocity of the vehicle and drone, respectively. Without loss of generality, we assume that both vehicles are moving at a constant velocity. Hence, the time required to traverse edge (i, j) is defined as $t_{ij} = d_{ij}/v$ and $\overline{t}_{ij} = \overline{d}_{ij}/\overline{v}$ for

the vehicles and drones, respectively. As we assumed that $\alpha \geq 1$ and $\bar{d}_{ij} \leq d_{ij} : \forall i, j \in V$, we may conclude that $\bar{t}_{ij} \leq t_{ij} : \forall i, j \in V$

We consider a limited set K of vehicles. Furthermore, each vehicle $k \in K$ holds an equal number of $D \in \mathbb{Z}$ drones assigned to it. Each drone may travel a maximum range of $\mathcal{E}$ distance units per operation, where a *drone-operation* is characterized by a triple (i, w, j) as follows: the drone is launched from a vehicle at a node $i \in V_L$, delivers a parcel to $w \in V_N$, and is retrieved by the vehicle from which it was launched and at node $j \in V_R$ ($i \neq w \neq j$ and $i \neq j$).

We use the following decision variables for modeling the VRPD:

x_{ij}^k :	$\forall i, j \in V, k \in K$:	Binary variables that state whether edge (i, j) belongs to the route of vehicle k.
y_{iwj}^k :	$\forall i, w, j \in V, k \in K$:	Binary variables that specify whether drone delivery (i, w, j) is performed by a drone associated to vehicle k.
p_{ij}^k :	$\forall i, j \in V, k \in K$:	Binary variables that indicate whether node i is served before but not necessarily consecutive to node j in the route of vehicle k.
u_i^k :	$\forall i \in V, k \in K$:	Integer variables with a lower bound of 0 that state the position of node i in the route of vehicle k, if customer i is served by vehicle k.
z_{aiwje}^k :	$\forall a, i, w, j, e \in V, k \in K$:	Binary variables that specify whether the drone delivery (a, w, e) is performed by a drone associated with vehicle k and node a precedes node i and node e follows node j in the route of vehicle k.
a_i^k :	$\forall i \in V, k \in K$:	Continuous variables with a lower bound of 0 that indicate the earliest time at which customer i is served by either the vehicle k or a drone associated with vehicle k. In the case that i is used as retrieval node a_i^k is the earliest time at which both the (latest) drone and the vehicle have arrived at node i.

The MILP formulation of the VRPD is given in (1) - (19) where (1) is the objective function and the constraints are given by (2) - (19).

$$\min \quad \tau \tag{1}$$

s.t.

$$\tau \geq a_{n+1}^k \quad : \quad \forall k \in K, \tag{2}$$

$$\sum_{j \in V_N} x_{0j}^k - \sum_{i \in V_N} x_{i,n+1}^k = 0 \quad : \quad \forall k \in K, \tag{3}$$

$$\sum_{i \in V_L} x_{ih}^k - \sum_{j \in V_R} x_{hj}^k = 0 \quad : \quad \forall k \in K, h \in V_N, \tag{4}$$

$$\sum_{j \in V_N} x_{0j}^k \leq 1 \quad : \quad \forall k \in K, \tag{5}$$

$$\sum_{i \in V_L} \sum_{k \in K} x_{ij}^k + \sum_{l \in V_L} \sum_{m \in V_R} \sum_{k \in K} y_{ljm}^k = 1 \quad : \quad \forall j \in V_N, \tag{6}$$

$$2y_{iwj}^k \leq \sum_{h \in V_L} x_{hi}^k + \sum_{l \in V_N} x_{lj}^k \quad : \quad \forall k \in K, i \in V_N, w \in V_N, j \in V_R, \tag{7}$$

$$y_{0,w,j}^k \leq \sum_{h \in V_L} x_{hj}^k \quad : \quad \forall k \in K, w \in V_N, j \in V_R, \tag{8}$$

$$u_i^k - u_j^k \leq (n+1)(1 - x_{ij}^k) \quad : \quad \forall k \in K, i \in V_L, j \in V_R, \tag{9}$$

$$u_j^k - u_i^k \geq 1 - (n+1)(1 - y_{iwj}^k) \quad : \quad \forall k \in K, i \in V_L, w \in V_N, j \in V_R, \tag{10}$$

$$u_i^k - u_j^k \geq 1 - (n+1)p_{ij}^k \quad : \quad \forall k \in K, i \in V_L, j \in V_R, \tag{11}$$

$$u_i^k - u_j^k \leq -1 + (n+1)(1 - p_{ij}^k) \quad : \quad \forall k \in K, i \in V_L, j \in V_R, \tag{12}$$

$$3z_{aiwje}^k \leq y_{awe}^k + p_{ai}^k + p_{je}^k \quad : \quad \forall k \in K, a \in V_L, i, w, j \in V_N, e \in V_R, \tag{13}$$

$$2 + z_{aiwje}^k \geq y_{awe}^k + p_{ai}^k + p_{je}^k \quad : \quad \forall k \in K, a \in V_L, i, w, j \in V_N, e \in V_R, \tag{14}$$

$$M(x_{ij}^k - 1) + a_i^k + t_{ij} \leq a_j^k \quad : \quad \forall k \in K, i \in V_L, j \in V_R, \tag{15}$$

$$M(y_{iwj}^k - 1) + a_i^k + \bar{t}_{iw} \leq a_w^k \quad : \quad \forall k \in K, i \in V_L, w \in V_N, j \in V_R, \tag{16}$$

$$M(y_{iwj}^k - 1) + a_w^k + \bar{t}_{wj} + \leq a_j^k \quad : \quad \forall k \in K, i \in V_L, w \in V_N, j \in V_R, \tag{17}$$

$$(\bar{d}_{iw} + \bar{d}_{wj}) y_{iwj}^k \leq \mathcal{E} \quad : \quad \forall k \in K, i \in V_L, j \in V_R, w \in V_N, \tag{18}$$

$$\begin{aligned} M(x_{ij}^k - 1) + \sum_{v \in V_N} y_{ivj}^k + \sum_{l \in V_N} \sum_{\substack{m \in V_R, \\ m \neq j}} y_{ilm}^k \\ + \sum_{\substack{p \in V_L, \\ p \neq i}} \sum_{q \in V_N} y_{pqj}^k + \sum_{a \in V_L} \sum_{w \in V_N} \sum_{e \in V_R} z_{aiwje}^k \leq D \quad : \quad \forall k \in K, i \in V_L, j \in V_R. \end{aligned} \tag{19}$$

Using the objective function represented by (1) along with constraint (2), we look for minimizing the arrival time a_{n+1}^k at which the last vehicle k (or drone that belongs to the vehicle k, where $k \in K$) arrives at the depot (i.e., node $n+1$).

Constraints (3) - (5) form the preservation of flow for the vehicles. More precisely, constraint (3) guarantees that the number of vehicles departing from the depot (i.e., node 0) is equal to the number of vehicles arriving at the depot (i.e,. node $n+1$). Constraint (4) ensures that for each node $h \in V_N$, the number of vehicles arriving at h must be equal to the number of vehicles departing from h. Due to constraint (5), each vehicle $k \in K$ may (but must not) start from the depot exactly once. Constraint (6) states that each node $j \in V_N$ is visited exactly once by either a vehicle or a drone.

Synchronization between routes of the vehicles and its drones is handled by constraints (7) and (8). That is, in the case of constraint (7), if a drone is set to perform operation (i, w, j) (i.e., $y_{iwj}^k = 1$), then vehicle $k \in K$ must visit nodes $i \in V_n$ and $j \in V_R$ at some point to allow a launch and retrieval of the drone at nodes i and j, respectively. Additionally, constraint (8) handles the synchronization between the vehicle and drone in the case that the drone is launched from the depot.

Constraints (9) - (10) link the variables u_i^k, for vehicle $k \in K$, to the decision variables x_{ij}^k and y_{iwj}^k, where $i \in V_L, j \in V_R$. Constraints (11) - (12) associate the variables p_{ij}^k with u_i^k and u_j^k. Constraints (13) - (14) determine a value on the variables z_{aiwje}^k depending on the decision variables p_{ij}^k and y_{iwj}^k.

Constraints (15) - (17) set lower bounds on the variables a_i^k. In particular, constraint (15) places lower bounds on the arrival time a_j^k in the case that $x_{ij}^k = 1$. Furthermore, constraints (16) and (17) put lower bounds on the arrival time a_w^k and a_j^k in the case that $y_{iwj}^k = 1$. The objective function guarantees that all these bounds are tight. Constraints (15) and (17) also ensure a synchronization on the lower bound of a_j^k (i.e., the lower bound will be determined by the latest arrival). Constraint (18) imposes a maximum distance (endurance) constraint on each drone operation (i, w, j). Finally, constraint (19) guarantees that during each transition of the vehicle from node i to node j, a maximum of D drones are in operation simultaneously. More precisely, the following drone operations may occur while a vehicle k travels from $i \in V_L$ to $j \in V_R$ (see also Figure 1):

- A drone may be launched from node i, make a delivery to $w \in V_N$, and be retrieved by the vehicle at node j (i.e., $y_{iwj}^k = 1$).
- A drone may be launched from node i and is set to be retrieved by the vehicle at $m \in V_R$, that is visited by the vehicle at some point after j (i.e., $y_{ilm}^k = 1$).
- A drone may have been launched from a previous vertex p, that was visited by the vehicle before i and is set to return to the vehicle at node j (i.e., $y_{pqj}^k = 1$).

- A drone may have been launched from $a \in V_L$, that is a predecessor of i in the vehicle's route, to perform a sortie (a, w, e) (i.e., $y^k_{awe} = 1$). The vehicle will retrieve the drone at node $e \in V_R$, that is a successor of node j in the vehicle's route.

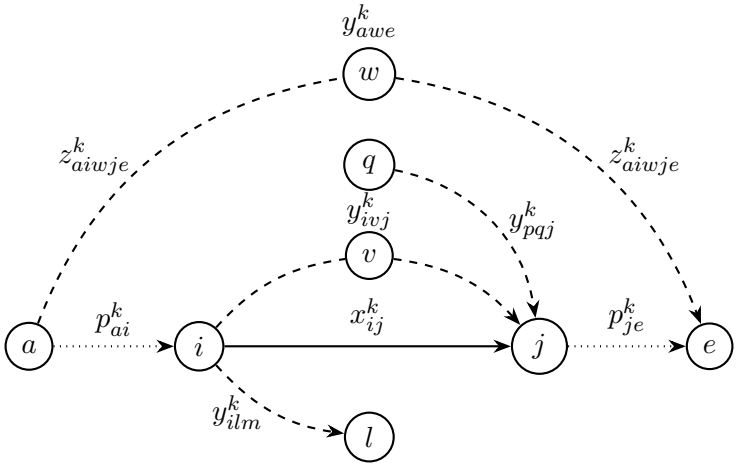

Figure 1: A visualization of constraint (19). The solid and dotted lines refer to the path of the vehicle k. A solid line indicates that the vertices i and j are adjacent on the route of the vehicle (i.e., $x^k_{ij} = 1$). In the case of a dotted line, the vertices (e.g., a and i) are not necessarily adjacent, i.e., further vertices might be visited in between the connection (i.e., $p^k_{ai} = 1$). The dashed lines indicate the path of a drone assigned to vehicle k. As p and m are arbitrary vertices that are visited by the vehicle before i and after j, respectively, they are not shown in the figure.

Any standard MILP solver, e.g., IBM Cplex and Gurobi, might be used to solve the model (2) - (19). However, due to the NP-hardness of the VRPD, there are computational challenges in solving the problem. In this paper, we address this issue by introducing some valid inequalities and by proposing a metaheuristic approach.

3.2 Valid Inequalities

In order to provide a higher performance for the MILP solvers in solving the VRPD, we derive valid inequalities that restrict the solution space of the problem while excluding none of the optimal solutions. Hence, the addition of valid inequalities can have a positive impact on the computational time that is required for solving the VRPD.

3.2.1 Lower Bounds on a^k_{n+1}

First, we derive two sets of valid inequalities that are used to place *lower bounds* (LB) on each variable a^k_{n+1}. Indeed, the valid inequalities are based on the earliest possible arrival time of the vehicle and drones that belong to $k \in K$ at node $n + 1$, respectively.

Proposition 1 *In the MILP (1) - (19), a^k_{n+1} is bounded by the earliest possible arrival time of vehicle k at node $n + 1$.*

Proof 1 *Let a^k_i be the earliest time at which node i is served by either the vehicle k or a drone associated with $k \in K$. In the case that i serves as a retrieval vertex, the time will be determined by the vehicle that arrives after the other one. The total time τ^k_T spent traveling on edges by vehicle $k \in K$ can be calculated as follows:*

$$\sum_{i \in V_L} \sum_{j \in V_R} x^k_{ij} t_{ij} = \tau^k_T \quad : \quad \forall k \in K. \tag{20}$$

In any VRPD solution, the earliest arrival time a^k_{n+1} of the vehicle $k \in K$ must be at least equal to or greater than the total time spent traveling on edges by the vehicle τ^k_T. In fact, as the vehicle k may

have to wait at vertices for the arrival of a drone, a^k_{n+1} might be larger than the time spent traveling by the vehicle. Consequently, we can derive the following valid inequality:

$$\sum_{i \in V_L} \sum_{j \in V_R} x^k_{ij} t_{ij} \leq a^k_{n+1} \quad : \quad \forall k \in K. \tag{21}$$

Therefore, we have proven Proposition 1. ■

In the special case of $D = 0$ or $y^k_{iwj} = 0 : \forall i \in V_L, w \in V_N, j \in V_R$, we know that $\tau^k_T = a^k_{n+1}$, because the vehicle will never have to wait at any vertex to synchronize with a drone. For other cases, the arrival time of the vehicle at node $n+1$ may exceed the time spent traversing the edges τ^k_T, as the vehicle may also have to spend time waiting at a node in order to synchronize with a drone.

Proposition 2 *In the MILP (1) - (19), a^k_{n+1} is bounded by the earliest possible arrival time of a drone that is associated with vehicle $k \in K$ at node $n+1$.*

Proof 2 *As introduced in Section 3.1, we have a given number of D drones that are associated with each vehicle $k \in K$. Let τ^k_i be the total time that the i-th drone, assigned to vehicle $k \in K$, spends traveling on edges. Then, for each $k \in K$, the total time spent traveling by the D drones can be calculated as follows:*

$$\sum_{i \in V_L} \sum_{w \in V_N} \sum_{j \in V_R} y^k_{iwj} (\bar{t}_{iw} + \bar{t}_{wj}) = \sum_{i=1}^{D} \tau^k_i \quad : \quad \forall k \in K. \tag{22}$$

Let $\tau^k_{max} = max(\tau^k_1, \ldots, \tau^k_D)$, i.e., τ^k_{max} is the longest time that a drone, associated to a vehicle $k \in K$, spends traveling. As a drone might also have to stay at a vertex in order to wait for the vehicle, we know that the earliest arrival time a^k_{n+1} must be at least equal to or greater than τ^k_{max}:

$$\tau^k_{max} \leq a^k_{n+1} \quad : \quad \forall k \in K. \tag{23}$$

Furthermore, we know that the average arrival time $\overline{\tau}^k_D$ of the set of drones D associated with vehicle $k \in K$ must always be less than or equal to the earliest arrival time τ^k_{max} in the given set of D drones:

$$\overline{\tau}^k_D = \frac{1}{D} \sum_{i=1}^{D} \tau^K_i \leq \tau^k_{max} \leq a^k_{n+1} \quad : \quad \forall k \in K. \tag{24}$$

By dividing both sides of Equation (22) with the number of drones, i.e., D, and based on the inequality in (24) we derive the set of valid inequalities in (25), that place a LB on the earliest arrival time a^k_{n+1}:

$$\frac{1}{D} \sum_{i \in V_L} \sum_{w \in V_N} \sum_{j \in V_R} y^k_{iwj} (\bar{t}_{iw} + \bar{t}_{wj}) \leq a^k_{n+1} \quad : \quad \forall k \in K. \tag{25}$$

This completes the proof of Proposition 2. ■

3.2.2 Upper Bounds on a^k_{n+1}

In this section, we derive one further set of valid inequalities that are used to place an *upper bound* (UB) on each variable a^k_{n+1}.

Proposition 3 *In the MILP (1) - (19), the latest possible arrival time a^k_{n+1}, at which the vehicle $k \in K$ and the last drone associated to it reach the node $n+1$, is bounded as follows:*

$$\sum_{i \in V_L} \sum_{j \in V_R} x^k_{ij} t_{ij} + \sum_{i \in V_L} \sum_{w \in V_N} \sum_{j \in V_R} y^k_{iwj} (\bar{t}_{iw} + \bar{t}_{wj}) \geq a^k_{n+1} \quad : \quad \forall k \in K. \tag{26}$$

Proof 3 *The earliest arrival time a^k_{n+1} of vehicle $k \in K$ or a drone, associated with vehicle k, depends on the time that either of them spend traveling and the time that either of them must wait at a vertex to synchronize with the other one.*

As we will later show in Section 4.4, the value of a^k_{n+1} is equal to the longest path that connects nodes 0 and $n+1$ through the directed acyclic sub-graph, determined by the decision variables x^k_{ij} and y^k_{iwj} (for all $k \in K$). The longest path in a directed acyclic graph is defined to be a simple path, i.e., it has no repeating vertices. If the graph is directed and acyclic, the longest path will never be longer than the sum of all edges in the sub-graph (see also (Cormen et al. 2009)). This fact is indeed described mathematically by inequalities (26). ■

In special cases, i.e., if $D = 0$ or all $y^k_{iwj} = 0 : \forall i \in V_L, w \in V_N, j \in V_R$, the bound described in (26) could be even tighter. The reason is the fact that, in these special cases, the vehicle will never have to wait for a drone and the earliest arrival time a^k_{n+1} will be equal to the total time spent traveling on edges by the vehicle (i.e., $\sum_{i \in V_L} \sum_{j \in V_R} x^k_{ij} t_{ij}$).

4 Variable Neighborhood Decomposition Search for the VRPD

In Section 3, we introduced a MILP formulation for the VRPD. The proposed MILP might be solved by any commercial MILP solver (e.g., IBM CPLEX, Gurobi Optimizer, etc.). However, the time required to optimally solve a VRPD grows considerably with an increase in the problem size. Indeed, from the model that was introduced in Section 3.1, we observe a strong growth in the number of decision variables and constraints with an increase in the problem size. In particular, we can observe a large increase in the amount of binary variables z^k_{aiwje} and the associated constraints that are required for modeling the VRPD.

In order to overcome the issue related to the size of the MILP formulation, we might investigate heuristic approaches. In fact, apart from exact methods, heuristics also play an important role in solving complex optimization problems. In this context, *metaheuristics* are solution methods that manage the interaction between the exploitation of local improvement procedures and exploration procedures that may be used to escape from local optima (Gendreau and Potvin 2010). Typically, we can categorize three types of metaheuristics (Toth and Vigo 2002, Labadie et al. 2016):

- metaheuristics that generate a sequence of solutions (e.g., *Simulated Annealing*, *Tabu Search*, *Variable Neighborhood Search*),
- metaheuristics that are based on a set of solutions (e.g, *Genetic Algorithms*, *Ant Colony Optimization*, *Particle Swarm Optimization*),
- hybrid metaheuristics that use various components.

This section will introduce an adaptation of *Variable Neighborhood Decomposition Search* (VNDS) for solving the VPRD.

4.1 Variable Neighborhood Search: Basic Concepts

Variable Neighborhood Search (VNS) is a metaheuristic framework that was proposed by Mladenović and Hansen (1997) for solving complex optimization problems. The authors propose a systematic change of increasingly distant neighborhoods of the current incumbent solution. The new neighborhoods are then explored by a local search method in order to identify the local optima. A new solution is accepted if and only if an improvement was made.

Let y be a feasible solution to an optimization problem min $f(y)$. Algorithm 1 details the abstract structure of the *Basic Variable Neighborhood Search* (BVNS) (Mladenović and Hansen 1997, Gendreau and Potvin 2010). More precisely, BVNS consists of several components and requires an initial solution y, as well as two integer parameters k_{max} and i_{max} that impose the maximum depth of the neighborhood and maximum number of iterations, respectively. In the following, we provide a short description of the components.

$y' \leftarrow Shake(y, k)$ A *shake* is a random move that is used to generate a new solution $y' \in \mathcal{N}_k(y)$. We refer to $\mathcal{N}_k(y)$ as the k-th neighborhood of y. Given a solution y, the neighborhood $\mathcal{N}_k(y)$ can be reached from y through exactly k shakes.

$y'' \leftarrow LocalSearch(y')$ A *local search* procedure is used to attempt improving the solution y' that was found in $\mathcal{N}_k(y)$ (i.e., the k-th neighborhood of y), which might lead to a new solution y''.

$y, k \leftarrow NeighborhoodChange(y, y'', k)$ Here, the operator determines if the new solution y'' replaces y as the incumbent solution. Algorithm 2 details the structure of the neighborhood change operator for a minimization problem. That is, if $f(y'') < f(y)$, then y'' will become the new incumbent solution y. Furthermore, we reset the parameter k to 1. If $f(y'') \geq f(y)$ we increase the value of the parameter k by 1.

Algorithm 1 BVNS(y, k_{max}, i_{max})

```
i ← 1;
repeat
    repeat
        y' ← Shake(y, k);
        y'' ← LocalSearch(y');
        y, k ← NeighborhoodChange(y, y'', k);
    until k = k_max
until i = i_max
return y;
```

Algorithm 2 NeighborhoodChange(y, y'', k)

```
if f(y'') < f(y) then
    y ← y'';
    k ← 1;
else
    k ← k + 1;
end if
return y, k;
```

Several extensions to BVNS have been proposed and an overview of the proposed methods is given in Gendreau and Potvin (2010). These extensions include *Variable Neighborhood Descent* (VND) by Brimberg et al. (2000) that searches the neighborhoods in a determenistic way, *Reduced VNS* (RVNS) by Mladenović et al. (2003) that skips the local search procedure, and *Skewed VNS* (SVNS) by Hansen et al. (2000) that allows for the exploration of solutions that are in distant neighborhoods of the incumbent solution. In this context, *Variable Neighborhood Decomposition Search* (VNDS) is another extensions of the VNS method that was introduced by Hansen et al. (2001). It extends the basic VNS into a nested two-level VNS scheme, based on a decomposition of the optimization problem.

Through our preliminary computational experiments (Schermer et al. 2018), we were determined that VNDS is a well-suited approach for solving the VRPD. In order to describe VNDS for the VRPD, we introduce the variables x and y that denote feasible solutions to a VRPD and VRP, respectively. The VRPD can be embedded as follows into the VNDS framework:

- We attempt to find a solution y to a VRP, where no drones are used, through VNS. It might be assumed that this decomposed sub-problem is easier to solve than the original VRPD. Furthermore, existing optimization techniques, that have been successfully applied to the VRP, can be used to solve the decomposed sub-problem.
- Once we have found a VRP solution y, we transform it to a VRPD solution x by inserting drones into the existing route of each vehicle $k \in K$.

- We design VNDS in a way to to manage the interaction of the sub-problems in a way, such that a good heuristic solution can be achieved.

Algorithm 3 details the abstract structure of the VNDS approach (Hansen et al. 2001, Gendreau and Potvin 2010) that has been adapted for the VRPD. The parameter k defines the depth of the neighborhood that will be explored and is initially set to 1. The parameter t_{max} determines the maximum run time. The structure of VNDS (Algorithm 3) consists of several components:

- $y \leftarrow Shake(y(x), k)$: A *Shake* is a random move that is used to generate a solution y of a VRP sub-problem that favors characteristics of the solution x of the VRPD problem.
- $y' \leftarrow BVNS(y, k)$: A BVNS heuristic is used to generate a new solution y' to the VRP sub-problem by improving the solution y (see Algorithm 1). The operators that we use during BVNS are introduced in Section 4.3.
- $x' \leftarrow LocalSearch(y', D)$: Once we have found a solution y' to the VRP sub-problem, we generate a solution x' to the VRPD problem by applying a local search operator. In our case, the local search operator is a drone insertion operator that improves a VRP solution by adding feasible drone sorties. The drone insertion operators that we use are introduced in detail in Section 4.4.
- $x, y(x), k \leftarrow NeighborhoodChange(x, x', y', k)$: This operator determines if the new solution x' replaces x as the incumbent solution. Algorithm 4 details the structure of the neighborhood change operator. If the objective value is improved, i.e., $f(x') < f(x)$, then x' will become the new incumbent solution. Furthermore, we reset the parameter k to 1 and keep track of y'. If $f(x') \geq f(x)$ we increment the value of the parameter k by 1.

Algorithm 3 VNDS($G, K, D, k_{max}, t_{max}$)

repeat
  $k \leftarrow 1$;
  repeat
    $y \leftarrow$ Shake$(y(x), k)$;
    $y' \leftarrow BVNS(y, k)$;
    $x' \leftarrow LocalSearch(y', D)$;
    $x, y(x), k \leftarrow NeighborhoodChange(x, x', y', k)$;
  until $k = k_{max}$
until $t > t_{max}$
return x;

Algorithm 4 NeighborhoodChange(x, x', y', k)

if $f(x') < f(x)$ **then**
  $x \leftarrow x'$;
  $y(x) \leftarrow y'$;
  $k \leftarrow 1$;
else
  $k \leftarrow k + 1$;
end if
return x, k;

4.2 Initialization

For generating an initial VRP solution, we use two-phase heuristic, called *route-first, cluster second* (RFCS)(for more details, see Toth and Vigo (2002) and references therein). The RFCS approach

works as follows: First, we build an initial tour using the *Nearest-Neighbor-Heuristic* (NNH) (see, e.g., Cormen et al. (2009)). More precisely, starting from the depot, we gradually construct a route by adding the closest unvisited vertex to the tour before returning to the depot. Afterwards, the tour is split equally, based on the number of available vehicles in the set K.

4.3 Shake and Local Search Operators for BVNS

This section is dedicated to explaining the operators used in the VNDS and BVNS algorithms (see Algorithms 1 and 3). In the case of VNDS, the shake operator (see Algorithm 3, line 4) will take the solution $y(x)$ to the VRP sub-problem that was either generated by RFCS during initialization or stored after performing a Neighborhood Change (see Algorithm 4, line 3).

Before introducing the BVNS-related operators, we discuss about some classical approaches in solving the VRP. Indeed, the classical heuristics that are used in improving VRP solutions might be classified in two categories (Toth and Vigo 2002):

- single-route improvement heuristics (also known as intraroute),
- multi-route improvement heuristics (also known as interroute).

Single-Route improvements focus on improving the routing of a single vehicle. Since the overall objective consists in minimizing the latest arrival time, that is determined by the last vehicle to return to the depot, single-route improvements are valuable. Perhaps, the most well-known heuristic for improving a single route is the *k-opt*, which is a generalization of the 2-opt operator (Lin and Kernighan 1973). Multi-Route improvements on the other hand try to improve the objective function by considering the routing of multiple vehicles at the same time. van Breedam classifies the multi-route improvements as follows, all of which can be considered as special cases of a 2-cyclic exchange (van Breedam 1994):

- *String Cross* (SC), where two edges are exchanged by crossing two edges of two different routes.
- *String Relocation* (SR), where one vertex is relocated from one route and added to another route.
- *String Exchange* (SE), where two vertices are exchanged between two different routes.
- *String Mix* (SM), which is a combination of SR and SE that selects the best move among the two.

For applying shake and local search moves during BVNS, we use the 2-opt, SR, and SE moves. In particular, all moves work by generating a *random number* to determine which vertices (and which routes in the case of SE and SR) might be affected by a 2-opt, SE, and SR.

- $y' \leftarrow Shake(y, k)$: In the case of a shake, we take the existing routes (belonging to each vehicle in K) and apply a random sequence of k operations (consisting of 2-opt, SR, and SE) to the routes. In order to provide a deep exploration of the solution space and to stir out of local minima, we always accept changes from shake operators even if the objective value is worse after applying a shake.
- $y'' \leftarrow LocalSearch(y')$: In the case of local search, we apply the same operations (i.e. 2-opt, SR, and SE) to the new routes. However, we apply a greedy search, that consists in applying a move only if it improves the current incumbent objective value.

4.4 Heuristics for Drone Insertion

This section is dedicated to introducing two heuristic procedures that can be used for incorporating drones in a VRP solution; hence generating a feasible VRPD solution.

Figure 2 shows a segment of a vehicle routing. If we assume that the number of drones $D = 0$, then the vehicle k arrives at the depot after finishing its tour; hence, as the vehicle never has to wait at any vertex, the time a^k_{n+1} is equal to the time required by the vehicle to serve all customers that

belong to its route. Now, consider that $D = 1$ and we are tasked with placing a single drone in a way, such that the arrival time in Figure 2 is reduced. Furthermore, assume that we know the arrival times at each vertex, $a_1, \ldots, a_6$, respectively.

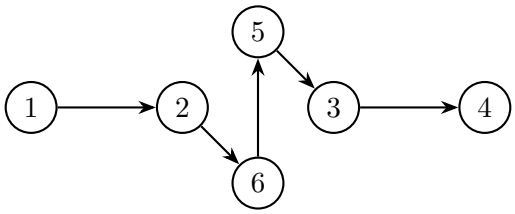

Figure 2: A segment of a vehicle routing.

Next, consider that we want to add the *drone operation* $(1, 5, 4)$. That is, as shown in Figure 3, the drone is launched from the vertex labeled 1, continues its flight to serve the vertex labeled 5, and will then be retrieved by the vehicle at the vertex labeled 4. Hence, the arrival time a'_4, at which both the vehicle and the drone have reached the node labeled 4, can be calculated as follows, where $a_4^{v,1}$ and $a_4^{d,1}$ are the earliest arrival time of the vehicle and the drone, respectively:

$$a_4^{d,1} = a_1 + \bar{t}_{1,5} + \bar{t}_{5,4} \tag{27}$$

$$a_4^{v,1} = a_4 - t_{6,5} - t_{5,3} + t_{6,3} \tag{28}$$

$$a'_4 = max(a_4^{d,1}, a_4^{v,1}) \tag{29}$$

Therefore, the sortie $(1, 5, 4)$ seems to be a reasonable choice only if $a'_4 < a_4$.

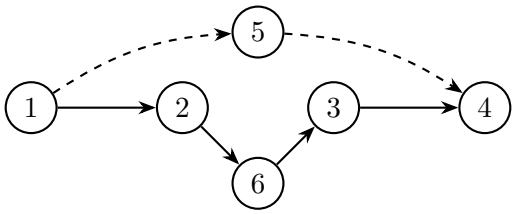

Figure 3: A segment of a VRPD routing with one drone performing the operation $(1, 5, 4)$.

Now, let us consider that $D = 2$, i.e., we have one more drone available. Given the structure of the graph, the operation $(2, 6, 3)$ seems like a reasonable choice for placing the drone, as shown in Figure 4. Furthermore, assume that we know the updated earliest arrival times at each vertex $a'_1, \ldots, a'_6$ of the graph depicted in Figure 3. The earliest arrival time a''_4, at which the vehicle and both drones have reached the vertex labeled 4, can be calculated as follows (where $a_i^{d,1}$ and $a_i^{d,2}$ are the earliest arrival time of the first and second drone and $a_i^{v,2}$ is the earliest arrival time of the vehicle at node i):

$$a_4^{d,1} = a'_1 + \bar{t}_{1,5} + \bar{t}_{5,4} \tag{30}$$

$$a_3^{v,2} = a'_2 - t_{2,6} - t_{6,3} + t_{2,3} \tag{31}$$

$$a_3^{d,2} = a'_2 + \bar{t}_{2,6} + \bar{t}_{6,3} \tag{32}$$

$$a_3'' = max(a_3^{d,2}, a_3^{t,2}) \tag{33}$$

$$a_4^{v,2} = a_3'' + t_{3,4} \tag{34}$$

$$a_4'' = max(a_4^{d,1}, a_4^{v,2}) \tag{35}$$

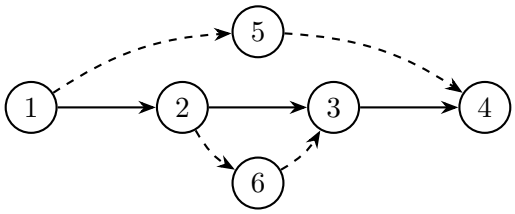

Figure 4: A segment of a VRPD routing with two drones performing the operations $(1, 5, 4)$ and $(2, 6, 3)$.

Clearly, adding the operation $(2, 6, 3)$ will improve the objective function only if the condition described in (36) holds.

$$a_4'' < a_4' = max(a_4^{d,1}, a_4^{v,1}) \leq max(a_4^{d,1}, a_4^{v,1}, a_4^{v,2}) \tag{36}$$

In other words, if the sequence of vertices $1, 5, 4$, that is performed by the first drone in Figure 3, is the longest path through the directed graph that connects vertices 1 and 4 (i.e., $a_4^{d,1} > a_4^{v,1}$)), then the earliest arrival time at that both the drone and the vehicle have reached the node labeled 4 in Figure 3 is determined by the drone as $a_4' = a_4^{d,1} = max(a_4^{d,1}, a_4^{v,1})$. Consequently, if $a_4' = a_4^{d,1} = max(a_4^{d,1}, a_4^{v,1})$, we will not be able to improve the objective function by reducing the arrival time of the vehicle at node 4 (i.e., find a drone sortie such that $a_4^{v,2} < a_4^{v,1}$), because the earliest arrival time a_4'' will always be determined by $a_4^{d,1}$. Further, if $a_4' = a_4^{v,1} = max(a_4^{d,1}, a_4^{v,1})$ then we may only improve the objective value if we reduce the arrival time of the vehicle by adding an additional drone sortie.

To sum up, we can draw the following conclusion: In any VRPD solution, the earliest time a_{n+1}^k at which the vehicle k and the latest drone, that is assigned to the vehicle k, arrive at the depot is determined by the longest path through the directed acyclic graph G^k (which is determined by the decision variables x_{ij}^k and $y_{iwj}^k : \forall k \in K$). Furthermore, an existing routing might only be improved through an additional drone operation, if the drone is assigned to a customer that previously belonged to the longest path through G^k and was served by the vehicle.

4.4.1 Greedy Insertion Heuristic

For inserting drones into a route, we propose a greedy heuristic operator named *Greedy Insertion Heuristic* (GIH). Algorithm 5 presents the pseudocde for the GIH and SIH (see Section 4.4.2). Let $G^k(V^k, A^k)$ be the directed acyclic graph that consists of a set of vertices $V^k \subset V$ and a set of arcs A^k that are determined by the decision variables x_{ij}^k and y_{iwj}^k. Given an existing VRP or VRPD solution, the GIH starts by calculating $a_i^k : \forall i \in V^k$. Then, we are able to identify which arcs and vertices, served by the vehicle, belong to the longest path through the directed acyclic graph G^k.

As we have established in Section 4.4, it can only be beneficial to assign a customer v_j to a drone sortie (v_i, v_j, v_k), only when v_j belongs to the longest path and is visited by the vehicle. Hence, the GIH attempts to continuously improve a_{n+1}^k by following the route of each vehicle and by inserting the *first feasible* sortie (v_i, v_j, v_k) into the existing solution that improves the arrival time. This is done by assigning a vertex v_j, that belongs to the longest path and is visited by the vehicle, to a drone. The procedure is repeated until all drones have been placed onto each vehicle's route or until no more feasible sorties are possible.

4.4.2 Savings Insertion Heuristic

The *Savings Insertion Heuristic* (SIH) is a another greedy heuristic operator that we propose for inserting drones into a tour. In the case of the SIH, a very similar procedure to the GIH takes place, i.e., we begin by identifying the longest path through the directed acyclic graph G^k. By following the route of the vehicle in an existing solution, for each vertex v_j, that belongs to the longest path through G^k and is served by the vehicle, the SIH attempts to improve a^k_{n+1} by inserting the sortie (v_i, v_j, v_k), that has promising *savings*.

Algorithm 5 shows the basic structure of the SIH. Compared to the GIH, the SIH, given a launch vertex v_i provides a more thorough search of good delivery and retrieval vertices v_j and v_k. However, when using the SIH, it is important to consider the effect of looking for the highest savings for a fixed launch vertex v_i. Indeed, the more customers are served by the vehicle, while the drone is performing a sortie (v_i, v_j, v_k), the more likely the possibility that it might be beneficial to use the drone for multiple shorter sorties, instead. In order to consider this effect, we use a heuristic estimator in the SIH. This estimator considers the number of the customers served by the vehicle during the drones' sorties. To this end, we introduce Equation (37) that estimates the savings s, depending on the change in arrival time $(a^k_k - a^{k'}_k)$ at the retrieval vertex v_k, when performing sortie (v_i, v_j, v_k). We then divide this value by e^d, where d is the number of vertices that are visited by the vehicle during the sortie (v_i, v_j, v_k) and e the Euler's number.[2]

$$s = \frac{a^k_k - a^{k'}_k}{e^d}. \tag{37}$$

Figure 5 shows how the vehicle routing that is shown in Figure 2 might be improved through the application of GIH and SIH, respectively. In this case, we assume that the vehicle is equipped with two drones.

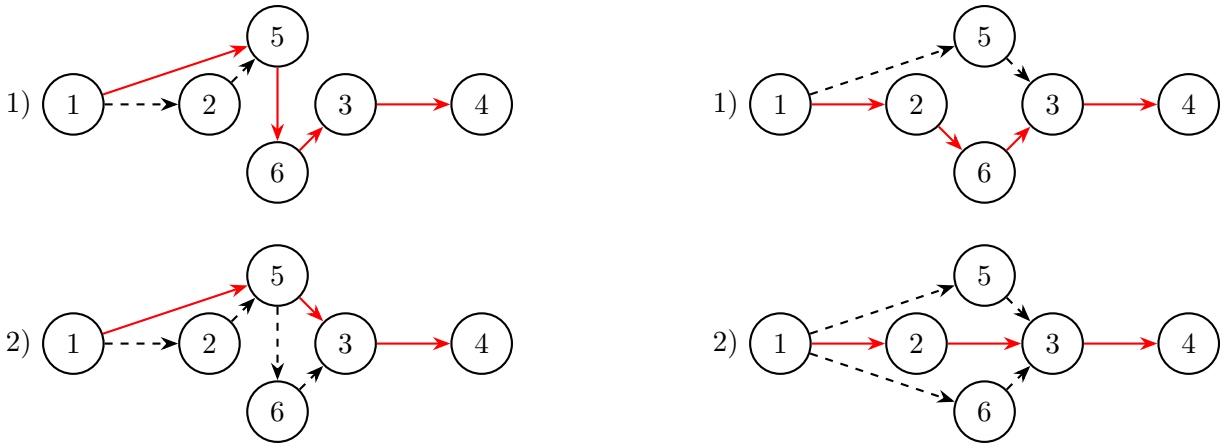

Figure 5: An improvement of the routing in Figure 2 that might be found through GIH (lefthand column) and SIH (righthand column) after two steps (indicated by 1) and 2)). In this case, we assume that two drones are available. In the solution found through GIH a single drone can perform both operations (dashed lines). In the solution found by SIH both drones are required to perform one operation each. In each case, we assume that the longest path is determined by the truck (solid lines colored in red).

[2]The number of customers d is weighted by the exponential function, as this produced the best results during our preliminary experiments compared to a linear or logarithmic dependency.

Algorithm 5 Drone Insertion Heuristics

$\pi^k :=$ ordered list of vertices as visited by vehicle k;
$G^k(V^k, A^k) :=$ directed graph determined by $x^k_{ij}, y^k_{iwj} : i \in V_L, w \in V_N, j \in V_R$;
$LP(G^k) :=$ longest path through G^k that connects v_0 with v_{n+1};
for each vehicle $k \in K$ and each drone assigned to k **do**
    update$(G^k, a^k_i, LP(G^k))$;
    for vertex $v_i \in \pi^k$ **do**
        for vertex $v_j \in \pi^k, j > i, v_j \in LP(G^k)$ that is not used for retrieval **do**
            for vertex $v_k \in \pi^k, k > j$ **do**
                if (v_i, v_j, v_k) is feasible **then**
                    if SIH selected **then**
                        calculate $s(v_i, v_j, v_k) = \dfrac{a^k_k - a^{k'}_k}{e^d}$;
                    end if
                    if GIH selected **then**
                        if sortie (v_i, v_j, v_k) improves a^k_k **then**
                            insert operation (v_i, v_j, v_k);
                            update$(G^k, a^k_i, LP(G^k), \pi^k)$;
                            continue with $v_i = v_k$;
                        end if
                    end if
                end if
            end for
        end for
        if SIH is selected and a feasible drone operation exists **then**
            insert operation (v_i, v_j, v_k) with the highest savings s;
            update$(G^k, a^k_i, LP(G^k), \pi^k)$;
            continue with $v_i = v_k$;
        end if
    end for
end for

5 Computational Study

This section is dedicated to our computational experiments and their numerical results. More precisely, in Section 5.1, the focus is on small-sized instances. In particular, we present the test settings in Section 5.1.1 and numerical results in Section 5.1.2. They are obtained by solving the MILP formulation of the VRPD, on small-scale instances, through Gurobi Solver 7.5.2. The large-scale instances have been solved by the VNDS algorithm presented in Section 4 and the results are reported in Section 5.2. We introduce the test settings in Section 5.2.1 and the numerical results in Section 5.2.2.

5.1 MILP model on Small-Scale Instances

This section details the results of our computational study on small-sized instances using the MILP formulation that was introduced in Section 3.

5.1.1 Implementation and Test Settings

The MILP formulation that was introduced in Section 3 is implemented and solved using the Java SE 8 interface of Gurobi Optimizer 7.5.2. We use the following test settings:

- We use two instances from the Solomon instances[3] for generating the graph $G = (V, E)$: *R101* and *RC101*. *R101* contains randomly generated coordinates, while *RC101* contains a mixture

[3]Available at `http://web.cba.neu.edu/~msolomon/problems.htm`

of random and clustered coordinates. Further, we limit ourselves to the first 11 vertices from each problem instance, i.e., we have 1 depot and 10 customer locations.[4]

- We test with $K = \{1\} \vee \{1, 2\}$, i.e., we have either one or two vehicles.
- We set $D \in \{1, 2\}$, i.e., we have either one or two drones per vehicle.
- We use $\alpha \in \{2, 3, 4\}$ (recall that α is the relative velocity of the drones).[5]
- We test with $\beta \in \{0.5, 0.75\}$ for instance *R101* and $\beta \in \{0.25, 0.5\}$ for instance *RC101*. We set the endurance $\mathcal{E}$ as follows: $\mathcal{E} = \beta \cdot max(D(G(V, E)))$ for each instance, where $max(D(G(V, E)))$ is the maximum value in the distance matrix of the graph. We use higher values of β in instance *R101* to allow for a suitable number of feasible drone operations.
- We use MILP (1) - (19) from Section 3.1. Additionally, we test the impact of adding the LB and UB that were introduced in Section 3.2.
- We use the Euclidean distance metric.
- We limit the run time of Gurobi Optimizer to 20 minutes.
- We run each experiment on a 3.1 GHz Quad Core CPU limited to the base clock speed with 8 GB of RAM available.

5.1.2 Numerical Results

Tables 1 and 2 show the numerical results of the experiments on the instances *R101* and *RC101*, respectively. Each table is divided into three further columns to show the difference between using the base model (i.e., MILP (1)-(19)) and the impact of adding the lower bounds and additionally adding the upper bounds (see also Section 3.2)[6]. Each table shows the best objective value τ that Gurobi Optimizer returned after run time. In order to highlight the potential benefit from using drones, the objective values (i.e., τ values) are shown relative to the optimal objective value of a problem, where no drone is available (i.e., $D = 0$). Furthermore, $\#N$ shows the number of nodes that have been explored by Gurobi Solver during the branch-and-cut phase. The column labeled *Gap* shows the relative gap between the objective bounds (i.e., the incumbent solution and the current objective bound provided by a leaf node of the search tree during the branch-and-cut phase). Finally, the column t shows the time in seconds that Gurobi required to solve the instance to optimality. In the case that no optimal solution could be found after 20 minutes, the column contains a T as an indication that Gurobi Optimizer encountered the preset run time limit.

Table 1 proves the effectiveness of the proposed VIEQ. Indeed, using the MILP without the VIEQ, no problem can be solved to proven optimality (i.e., with a gap equal to 0%) within the run time limit imposed on Gurobi Optimizer. Adding the LB has a two-sided positive impact: on the one hand the run time is reduced significantly, on the other hand the number of nodes that are searched during the branch-and-cut phase are also reduced by a similar magnitude. In some cases (in particular for $\beta = 0.50$), using the base MILP, the incumbent solution returned by Gurobi after run time is equal to the optimal solution, albeit, without any proof of optimality. Overall, on instance *R101*, the presence of a single drone can significantly improve the objective value, if the endurance parameter β is large enough. Furthermore, we have a clear indication that the marginal benefit from the application of a second drone is much lower, than that of the first one, as the objective values can not be improved in the same way.

[4]The instances that we used for our experiments are available at: `https://bisor.wiwi.uni-kl.de/fileadmin/bisor.wiwi.uni-kl.de/VRPD_Instances.zip`

[5]While there is no commercial parcel delivery drone in operation, current experimental drones have a maximum velocity of up to 43 . . . 70 kilometers per hour and an operational range of 10 . . . 30 kilometers (DHL International GmbH 2016, DPDgroup International Services GmbH & Co. KG 2016). Thus, our assumptions seem reasonable.

[6]The impact of adding the upper bounds, while no lower bounds were present, was negligible and is thus not further investigated here.

Table 2 highlights that instance *RC101*, that contains a set of clustered customers, is more difficult to solve.In particular, while the VIEQ help to provide tighter gaps and return better objective values after run time, almost no problem can be solved to optimality. Indeed, as *RC101* contains a set of clustered customers, the number of feasible drone operations is significantly larger, making this instance more difficult to solve. This behavior is also documented by the larger amount of explored nodes in the branch-and-cut tree during presence of the VIEQ compared to instance *R101*. Additionally, in contrast to the results in Table 1, the relative objective values are much larger. While the best relative objective values on instance *R101* are 67.83% and 72.42% for one and two vehicles, respectively, the best relative objective values on instance *RC101* are 86.90% and 83.36%.

Finally, based on our experiments, we report that the VRPD does not have an *optimal substructure*, making it particularly hard to solve the problem through a dynamic programming approach. This is further investigated in the Appendix.

Table 1: Results on Instance R101. The objective values τ are shown relative to the optimal objective values $\tau_1 = 173.042$ and $\tau_2 = 110.173$ of a problem with one or two vehicles, respectively, where no drones are available.

		MILP				MILP with LB				MILP with LB and UB			
β	α	τ [%]	#N	Gap [%]	t [s]	τ [%]	#N	Gap [%]	t [s]	τ [%]	#N	Gap [%]	t [s]
One vehicle equipped with one drone:													
.50	2	97.64	975656	100	T	97.64	66	0	1.59	97.64	68	0	1.75
	3	97.64	987453	100	T	97.64	206	0	1.59	97.64	73	0	1.72
	4	97.64	970392	100	T	97.64	80	0	1.75	97.64	71	0	1.84
.75	2	82.08	49495	100	T	78.16	1675	0	35.02	78.16	1434	0	32.44
	3	89.33	64444	100	T	73.78	583	0	19.05	73.78	555	0	18.84
	4	84.34	56448	100	T	73.61	575	0	20.27	73.61	764	0	29.41
One vehicle equipped with two drones:													
.50	2	97.64	1067860	100	T	97.64	72	0	1.73	97.64	66	0	1.84
	3	97.64	1026758	100	T	97.64	74	0	1.63	97.64	41	0	1.8
	4	97.64	1072514	100	T	97.64	62	0	1.86	97.64	67	0	1.7
.75	2	75.46	50995	100	T	71.28	394	0	17.24	71.28	826	0	22.77
	3	72.01	62969	100	T	67.83	163	0	18.89	67.83	325	0	18.69
	4	71.68	65164	100	T	67.83	113	0	16.22	67.83	491	0	18.24
Two vehicles equipped with one drone per vehicle:													
.50	2	100.06	659136	100	T	96.30	4308	0	7.48	96.30	4729	0	7.44
	3	96.30	579957	100	T	96.30	5560	0	12.09	96.30	4535	0	13.27
	4	96.30	533929	100	T	96.30	6654	0	13.56	96.30	4559	0	14.05
.75	2	85.44	41900	100	T	78.41	2356	0	269.65	78.41	3380	0	344.94
	3	84.43	37837	100	T	74.17	2360	0	322.85	74.17	2262	0	321.36
	4	79.18	30424	100	T	72.81	2023	0	216.09	72.81	2354	0	299.19
Two vehicles equipped with two drones per vehicle:													
.50	2	99.17	743606	100	T	96.30	5589	0	13.33	96.30	5054	0	12.09
	3	100.06	535470	100	T	96.30	4328	0	12.99	96.30	7170	0	15.97
	4	96.30	494188	100	T	96.30	4013	0	12.67	96.30	3690	0	12.8
.75	2	88.06	42850	100	T	74.10	3280	0	312.27	74.10	2263	0	374.07
	3	75.13	42347	100	T	72.42	1865	0	257.15	72.42	3871	0	84.38
	4	88.06	42142	100	T	72.42	3284	0	81.72	72.42	4314	0	92.35

Table 2: Results on Instance RC101. The objective values τ are shown relative to the optimal objective values $\tau_1 = 137.777$ and $\tau_2 = 95.885$ of a problem with one or two vehicles, respectively, where no drones are available.

		MILP				MILP with LB				MILP with LB and UB			
β	α	Z [%]	#N	Gap [%]	t [s]	Z [%]	#N	Gap [%]	t [s]	Z [%]	#N	Gap [%]	t [s]
One vehicle equipped with one drone:													
.25	2	95.89	32799	100	T	93.04	16753	20.65	T	93.27	15324	22.69	T
	3	94.10	23107	100	T	92.29	15802	20.18	T	93.12	14751	22.46	T
	4	93.25	23437	100	T	92.29	18867	19.35	T	92.29	15077	22.18	T
.50	2	96.75	5461	100	T	94.42	2275	33.53	T	92.64	2397	32.08	T
	3	96.39	3842	100	T	93.46	2315	32.60	T	93.12	2214	32.05	T
	4	96.09	5495	100	T	94.57	2287	33.70	T	93.37	2267	37.18	T
One vehicle equipped with two drones:													
.25	2	93.59	28026	100	T	91.49	19982	17.87	T	91.49	15986	18.49	T
	3	91.47	25876	100	T	91.33	18585	18.04	T	91.59	15211	20.01	T
	4	93.19	24511	100	T	91.33	20531	18.39	T	91.47	19773	19.81	T
.50	2	88.68	6783	100	T	89.77	2380	30.21	T	89.21	2128	29.02	T
	3	91.52	5598	100	T	87.61	2315	27.32	T	90.51	2346	29.90	T
	4	89.60	6446	100	T	86.90	2630	27.14	T	89.33	2382	28.15	T
Two vehicles equipped with one drone per vehicle:													
.25	2	92.23	20648	100	T	91.34	9347	19.88	T	91.34	10030	19.88	T
	3	98.92	25950	100	T	90.26	12091	18.93	T	90.26	11201	18.93	T
	4	96.17	21829	100	T	92.87	5384	21.20	T	90.20	10831	18.88	T
.50	2	100.78	6070	100	T	103.71	1208	32.54	T	102.62	1164	31.82	T
	3	99.86	5988	100	T	95.70	1139	25.72	T	106.87	1139	34.54	T
	4	94.33	7100	100	T	93.54	1037	25.20	T	93.98	1271	25.56	T
Two vehicles equipped with two drones per vehicle:													
.25	2	93.47	20136	100	T	88.55	11707	17.37	T	88.58	10892	17.39	T
	3	100	23745	100	T	88.33	10967	17.15	T	88.33	12395	17.15	T
	4	88.52	25234	100	T	88.33	11652	0.00	1162.7	88.33	11441	17.15	T
.50	2	99.28	5071	100	T	88.07	1134	20.56	T	88.52	1071	20.91	T
	3	90.40	5672	100	T	92.35	1109	26.83	T	90.07	1228	22.32	T
	4	93.03	5522	100	T	83.36	1094	16.08	T	84.17	1097	16.88	T

5.2 Metaheuristic on Large-Scale Instances

As we have shown in Section 5.1, using the MILP approach for solving the VRPD is only feasible on small instances. For practical purposes, it might be necessary to solve large VRPD instances in a short time span. To this end, we compare the VNDS metaheuristic with both drone insertion operators (see Section 4.4) on larger test instances.

5.2.1 Implementation and Test Settings

We implemented the VNDS approach, that is described in Section 4, in Java SE 8, and tested the algorithm by using the following test settings:

- We used five instances from the TSPLIB[7]: *att532*, *berlin52*, *ch150*, *kroA200*, and *rd400*, ranging from 52 to 532 vertices, for generating the graph $G = (V, E)$. The first vertex in each instance was used as the depot location and the remaining vertices were considered customer locations.[8]
- We test with $K = \{1\} \vee \{1, 2\} \vee \{1, 2, 3\}$ and $D \in \{1, 2, 3\}$.
- We use $\alpha \in \{2, 3, 4\}$ (recall that α is the relative velocity of the drones).
- We test with $\beta \in \{0.1, 0.25, 0.5\}$, where the parameter β is used as follows: Let $max(A(G(V, E)))$ be the maximum value in the distance matrix of the graph G. Then, we set the endurance $\mathcal{E} = \beta \cdot max(A(G(V, E)))$ for each instance.
- We use the Euclidean distance metric.
- We test with both drone insertion heuristics, GIH and SIH[9], that were introduced in Section 4.4. As a reference value, we also test with $D = 0$.
- We set $k_{max} = 10$.
- We set i_{max} during BVNS to $\lceil 0.25 \cdot |V_N| \rceil$
- We run our heuristic 5 times per set of test settings (i.e., instance, α, β, ...).
- The heuristic is given a maximum run time of 10 minutes. The heuristic terminates early, in the case that the incumbent objective value remains unchanged for one minute.
- We run our experiments on a 3.1 GHz Quad Core CPU limited to the base clock speed with 8 GB of RAM available.

[7] Available at `https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

[8] The instances that we used for our experiments are available at: `https://bisor.wiwi.uni-kl.de/fileadmin/bisor.wiwi.uni-kl.de/VRPD_Instances.zip`

[9] Figure 6 show sample output using the SIH.

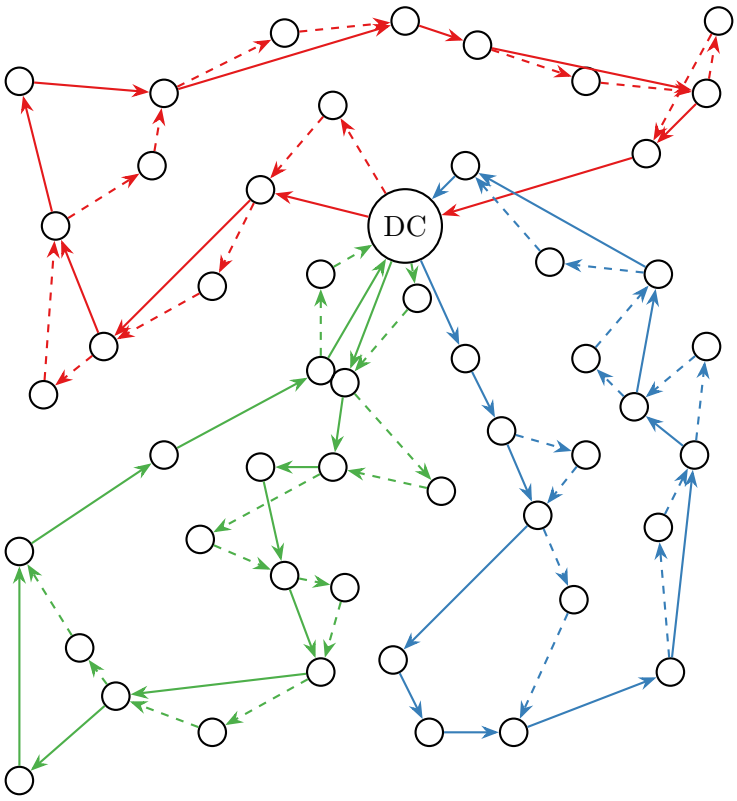

Figure 6: Sample output using the SIH on a problem instance with 50 customer locations with three vehicles, each vehicle equipped with one drone that has a limited flight endurance. The vehicles' and drones' paths are shown using solid and dashed lines, respectively.

5.2.2 Numerical Results

Tables 3 - 5 contain the results of the metaheuristic in solving the large-scale VRPD instances. More precisely, each table is divided into multiple columns and rows that differentiate different parameters (e.g., α, β, instance, number of vehicles and drones per vehicle) from each other. Tables 3 and 4 present the objective values for the GIH and SIH, respectively. In the column that represents the case where no drone is used, the objective values τ are shown. In contrast to the τ values that are real numbers, for the sake of comparing the results, the remaining objective values (i.e., in presence of drones) are reported in percentage relative to the first column of the same row. In Table 5, the shares of customers that are served by a drone are shown as a percentage of all customers $|V_N|$.

Overall, based on the numerical results reported in Tables 3 - 5, we can make the following observations. In most of the instances, any of the drone insertion heuristics provide solutions that reduce the mission time significantly. Furthermore, for a fixed value of parameter β, an increase in the relative velocity α results in an improved relative objective value τ and a higher share of drone deliveries in most cases. In particular when α is increased from 2 to 3, a noticeable improvement is visible in most cases. Increasing the value of α from 3 to 4 does not have significant influence on the objective value and the share of drone deliveries remain almost unchanged. Indeed, we observe that, after reaching a certain threshold, the arrival times may always be determined by the vehicle, i.e., no improvement by any further speeding up of the drone(s). The same behavior can be observed for a fixed relative velocity α. Here, an increase in β, in most cases, yields a better objective value and higher quota of using drones in deliveries. In particular, on some instances (e.g., the VRPD instances based on the berlin52, ch150, kroA200) a significant increase in the objective value is visible when a certain threshold is reached such as when β is increased from 0.1 to 0.25. For other cases (e.g., att532 with $\beta = 0.25$ or 0.5), the objective value shows little sensitivity with respect to a change in β. In these cases, when the endurance is already sufficiently large, it may not be possible to improve the objective value by further increasing the endurance. Finally, we observe a decreasing marginal benefit of each additional drone. As an example, consider instance berlin52 for the case of one vehicle,

$\alpha = 4, \beta = 0.5$ in Tables 3 and 4. A single drone allows for a significant reduction in the mission time leading to objective values of 63.67% and 61.12%. When increasing the number of drones from one to two and in particular from two to three the relative change in the objective value becomes increasingly smaller. This indicates that it becomes gradually more difficult to utilize further drones efficiently.

With the objective of comparing GIH versus SIH, Table 6 presents some statistics on the performance of these heuristics. More precisely, the table shows the average savings compared to the cases where no drone is used (calculated as $100\% - \tau[\%]$) and the number of times either heuristic dominated the other one depending on the number of vehicles and drones per vehicle.

According to the numerical results, we can emphasize that the SIH outperforms the GIH consistently over different combinations of vehicles and drones per vehicle. Indeed, the SIH provides much higher savings on average, even though, based on Table 5, the share of drone deliveries in solutions found through either drone insertion heuristic is often very similar. From the 405 cases recorded in Table 6, the GIH produced better results in only 54 cases whereas the SIH produced better results in the remaining 351 cases. Furthermore, the SIH also lead to better average savings of 19.83% compared to the average savings of 13.34% achieved through the GIH.

Finally, we are interested in analyzing the performance of SIH and GIH versus Gurobi Optimizer (tuned by valid inequalities) on instances where an optimal solution is known (see Tables 1 and 2 for the optimally solved cases). To this end, Table 7 shows the best objective values produced by the SIH and GIH, and the time t (in seconds) after which the solution was found. The results of the heuristics are compared to those provided by the MILP solver. We observe that, overall, both heuristics produce competitive results and identify optimal or near-optimal solutions in all cases.

When we conduct a detailed study on the differences between the optimal solutions achieved through the MILP formulation and the near-optimal solutions provided by the metaheuristic, we can identify two reasons why the metaheuristic fails to achieve the optimal solutions in some cases. In fact, on the one hand, the routing of the vehicle might not be optimal in the solution found by the metaheuristic.Indeed, as we have identified in Section 5.1.2 and it is also illustrated in the Appendix, in an optimal VRPD solution, the routing of the vehicle might exhibit intersecting arcs. On the other hand, the routing of the drone might not be optimal. In particular, a future algorithm may consider local search procedure that investigates changes to the route of the vehicle after drones have been placed. This might help identify solutions such as the one depicted in Figure 7 (see the Appendix). Furthermore, a more elaborated drone insertion heuristic might consider additional features when calculating the estimated savings, hence, allowing for an improved placement of drones in some additional cases.

Table 3: Numerical results of the metaheuristic on large-scale instances for different numbers of vehicles, drones, relative velocities α, and parameters β using the GIH. The objective values τ are shown as absolute values in the case of no drone and relative to the absolute value in the same row for the remaining cases.

Instance	β	*No Drone*	*One drone per vehicle*			*Two drones per vehicle*			*Three drones per vehicle*		
			$\alpha = 2.0$	$\alpha = 3.0$	$\alpha = 4.0$	$\alpha = 2.0$	$\alpha = 3.0$	$\alpha = 4.0$	$\alpha = 2.0$	$\alpha = 3.0$	$\alpha = 4.0$
One vehicle											
att532	0.1	93813.03	97.60%	96.37%	96.84%	95.06%	93.88%	93.43%	92.3%	90.1%	90.1%
	0.25		93.40%	91.55%	88.78%	86.24%	84.92%	82.28%	81.7%	82.2%	82.2%
	0.5		94.61%	92.79%	87.52%	83.89%	87.43%	81.57%	81.8%	83.3%	83.3%
berlin52	0.1	7716.69	97.74%	96.44%	96.81%	98.27%	96.15%	98.78%	97.2%	97.5%	97.5%
	0.25		83.10%	84.29%	85.12%	80.39%	79.74%	84.10%	79.8%	78.2%	78.2%
	0.5		63.31%	60.88%	63.67%	53.55%	59.03%	56.09%	53.1%	55.1%	55.1%
ch150	0.1	6697.48	97.14%	99.13%	98.48%	100.60%	93.93%	96.66%	95.7%	95.4%	95.4%
	0.25		74.95%	72.97%	72.92%	66.22%	66.69%	67.78%	64.4%	67.2%	67.2%
	0.5		72.95%	73.73%	69.83%	63.99%	60.26%	68.31%	63.5%	58.5%	58.5%
kroA200	0.1	29866.99	95.24%	99.84%	92.62%	91.10%	91.18%	92.68%	94.1%	87.7%	87.7%
	0.25		81.64%	81.17%	79.13%	74.19%	70.36%	77.06%	74.0%	72.6%	72.6%
	0.5		87.99%	83.65%	80.24%	78.90%	75.00%	80.49%	77.4%	71.6%	71.6%
rd400	0.1	16282.64	96.13%	96.30%	90.77%	91.27%	94.61%	91.48%	89.5%	90.8%	90.8%
	0.25		87.37%	84.46%	85.75%	78.58%	79.32%	80.53%	78.4%	77.0%	77.0%
	0.5		94.62%	90.13%	83.64%	82.64%	82.12%	81.34%	79.3%	80.1%	80.1%
Two vehicles											
att532	0.1	53070.52	104.29%	98.42%	97.60%	98.37%	96.93%	95.88%	97.1%	95.5%	95.5%
	0.25		97.55%	96.21%	93.91%	90.00%	85.54%	92.03%	85.6%	90.1%	90.1%
	0.5		97.09%	100.12%	92.62%	92.04%	90.44%	87.64%	86.2%	87.3%	87.3%
berlin52	0.1	4416.70	101.61%	106.12%	106.45%	105.92%	103.87%	105.24%	107.7%	98.0%	98.0%
	0.25		98.23%	98.23%	95.73%	98.94%	91.12%	83.56%	88.4%	100.8%	100.8%
	0.5		78.40%	76.66%	82.24%	64.50%	71.41%	73.31%	62.8%	68.4%	68.4%
ch150	0.1	3834.22	96.68%	97.58%	99.57%	98.24%	101.08%	96.69%	97.3%	99.9%	99.9%
	0.25		79.99%	83.71%	81.38%	74.21%	75.61%	78.07%	75.1%	77.7%	77.7%
	0.5		81.51%	76.01%	76.60%	69.22%	71.17%	75.44%	71.9%	68.6%	68.6%
kroA200	0.1	17129.32	97.49%	94.11%	96.19%	96.07%	101.61%	99.74%	96.6%	91.4%	91.4%
	0.25		85.51%	81.04%	86.44%	81.85%	76.37%	79.07%	79.2%	76.2%	76.2%
	0.5		91.29%	85.56%	82.84%	82.48%	79.33%	86.15%	76.0%	79.6%	79.6%
rd400	0.1	8896.51	92.90%	96.43%	95.80%	94.36%	90.56%	91.77%	94.3%	94.4%	94.4%
	0.25		87.48%	88.67%	86.28%	79.88%	79.95%	82.68%	80.3%	79.2%	79.2%
	0.5		94.01%	91.59%	85.03%	81.66%	78.97%	84.74%	79.6%	78.6%	78.6%
Three Vehicles											
att532	0.1	37980.07	102.98%	107.00%	102.75%	101.41%	100.11%	100.00%	98.7%	98.8%	98.8%
	0.25		94.19%	93.08%	97.41%	93.61%	87.37%	89.29%	87.8%	95.3%	95.3%
	0.5		100.17%	101.69%	95.76%	89.27%	94.14%	88.65%	87.8%	87.0%	87.0%
berlin52	0.1	3198.58	99.19%	103.05%	104.11%	100.76%	101.04%	101.25%	102.4%	105.1%	105.1%
	0.25		96.11%	100.70%	100.05%	88.70%	95.71%	101.14%	94.4%	96.0%	96.0%
	0.5		79.92%	68.72%	79.93%	68.14%	67.10%	80.11%	64.5%	78.6%	78.6%
ch150	0.1	2760.36	99.54%	97.06%	100.62%	97.92%	101.11%	104.59%	102.8%	102.4%	102.4%
	0.25		82.96%	84.39%	87.75%	80.61%	79.05%	82.95%	75.8%	76.4%	76.4%
	0.5		78.42%	76.45%	77.10%	74.48%	74.12%	81.87%	75.7%	74.8%	74.8%
kroA200	0.1	12378.97	94.49%	97.88%	95.11%	95.46%	93.88%	99.47%	95.0%	93.8%	93.8%
	0.25		88.20%	80.90%	86.59%	83.92%	74.65%	85.87%	78.7%	81.2%	81.2%
	0.5		88.74%	86.71%	81.97%	84.95%	81.59%	83.90%	75.9%	71.8%	71.8%
rd400	0.1	6205.54	103.36%	97.62%	99.43%	93.47%	97.23%	96.06%	95.4%	96.0%	96.0%
	0.25		89.20%	88.49%	87.20%	83.33%	81.90%	81.67%	82.7%	79.6%	79.6%
	0.5		97.08%	91.38%	85.71%	82.90%	84.11%	82.42%	86.5%	79.2%	79.2%

Table 4: Numerical results of the metaheuristic on large-scale instances for different numbers of vehicles, drones, relative velocities α, and parameters β using the SIH. The objective values τ are shown as absolute values in the case of no drone and relative to the absolute value in the same row for the remaining cases.

Instance	β	*No Drone*	*One drone per vehicle*			*Two drones per vehicle*			*Three drones per vehicle*		
			$\alpha = 2.0$	$\alpha = 3.0$	$\alpha = 4.0$	$\alpha = 2.0$	$\alpha = 3.0$	$\alpha = 4.0$	$\alpha = 2.0$	$\alpha = 3.0$	$\alpha = 4.0$
One Vehicle											
att532	0.1	93813.03	86.70%	85.22%	81.49%	85.08%	80.89%	81.12%	79.9%	79.2%	80.6%
	0.25		84.95%	81.85%	80.91%	80.34%	73.60%	72.36%	72.0%	71.6%	69.5%
	0.5		84.26%	82.85%	84.60%	76.63%	75.01%	73.03%	70.9%	68.2%	69.6%
berlin52	0.1	7716.69	95.85%	95.30%	95.27%	95.09%	95.06%	95.55%	96.0%	94.9%	94.7%
	0.25		85.17%	80.85%	80.65%	81.30%	76.60%	85.13%	81.7%	77.9%	79.4%
	0.5		69.06%	61.75%	61.12%	64.28%	53.25%	54.44%	61.3%	53.1%	49.4%
ch150	0.1	6697.48	92.95%	96.17%	91.16%	91.99%	92.50%	93.58%	94.4%	91.9%	93.3%
	0.25		76.41%	69.81%	68.32%	72.95%	60.22%	60.63%	63.8%	62.6%	58.8%
	0.5		76.46%	69.84%	68.42%	69.29%	59.91%	57.95%	63.6%	60.3%	57.9%
kroA200	0.1	29866.99	87.83%	83.55%	81.47%	82.14%	79.56%	77.60%	81.9%	78.5%	75.3%
	0.25		78.07%	73.60%	67.77%	70.72%	62.10%	62.66%	74.4%	68.6%	64.8%
	0.5		79.57%	74.60%	74.07%	69.31%	67.28%	62.99%	68.9%	62.7%	61.2%
rd400	0.1	16282.64	84.25%	81.82%	85.52%	81.23%	77.25%	84.23%	80.1%	82.0%	83.1%
	0.25		81.54%	77.90%	74.18%	73.96%	68.62%	67.93%	70.1%	67.1%	68.8%
	0.5		82.18%	79.54%	77.86%	75.86%	72.81%	70.25%	72.4%	70.1%	70.8%
Two Vehicles											
att532	0.1	53070.52	88.20%	88.76%	85.42%	89.33%	84.84%	85.27%	83.4%	81.1%	82.8%
	0.25		87.74%	84.67%	84.60%	84.32%	75.75%	80.82%	79.3%	73.9%	76.4%
	0.5		88.71%	82.90%	84.20%	76.65%	77.20%	78.93%	76.8%	76.2%	79.0%
berlin52	0.1	4416.70	97.55%	94.65%	100.78%	99.80%	97.78%	94.45%	100.2%	98.1%	94.1%
	0.25		94.95%	91.00%	97.44%	79.41%	91.15%	94.76%	94.5%	83.9%	86.3%
	0.5		77.38%	67.24%	67.20%	71.93%	64.65%	63.20%	66.0%	66.8%	66.8%
ch150	0.1	3834.22	97.54%	91.78%	96.28%	96.44%	94.06%	92.90%	93.7%	96.6%	93.5%
	0.25		82.71%	74.20%	69.32%	70.26%	70.23%	68.58%	74.5%	66.6%	66.8%
	0.5		82.87%	75.85%	69.70%	67.67%	65.59%	68.70%	70.3%	64.8%	67.2%
kroA200	0.1	17129.32	88.36%	86.56%	86.16%	85.11%	84.91%	86.48%	85.5%	84.6%	82.6%
	0.25		75.15%	79.89%	76.44%	75.91%	71.00%	69.57%	74.2%	71.2%	69.4%
	0.5		83.41%	80.19%	78.52%	75.29%	72.72%	72.86%	70.6%	69.6%	63.2%
rd400	0.1	8896.51	89.38%	89.32%	86.61%	86.87%	85.05%	81.69%	82.3%	84.4%	82.4%
	0.25		84.25%	78.68%	79.23%	74.49%	75.18%	69.63%	74.8%	68.3%	70.6%
	0.5		87.02%	84.19%	79.33%	78.35%	71.70%	73.80%	74.1%	73.6%	72.4%
Three Vehicles											
att532	0.1	37980.07	91.89%	88.93%	91.72%	89.13%	88.27%	85.82%	88.8%	87.1%	87.1%
	0.25		88.16%	91.10%	80.72%	87.57%	81.43%	80.56%	74.5%	80.0%	78.6%
	0.5		88.74%	82.97%	84.30%	83.33%	77.33%	80.91%	83.0%	76.7%	79.6%
berlin52	0.1	3198.58	102.14%	97.55%	100.93%	103.52%	97.23%	95.69%	102.7%	99.0%	104.7%
	0.25		101.38%	100.33%	92.98%	90.14%	95.20%	92.69%	93.3%	93.3%	93.3%
	0.5		89.02%	77.35%	66.99%	71.43%	64.45%	65.15%	70.6%	65.5%	61.2%
ch150	0.1	2760.36	99.15%	92.71%	99.07%	99.90%	99.48%	93.21%	100.9%	97.1%	96.0%
	0.25		85.37%	81.48%	78.55%	78.60%	79.28%	76.93%	78.2%	76.6%	77.4%
	0.5		81.66%	79.28%	78.60%	74.01%	72.17%	72.35%	76.5%	69.7%	71.2%
kroA200	0.1	12378.97	96.71%	90.54%	90.56%	93.29%	89.43%	92.41%	95.5%	89.8%	87.9%
	0.25		80.39%	81.95%	77.68%	84.42%	78.93%	76.21%	77.2%	73.3%	71.3%
	0.5		89.18%	78.46%	83.50%	79.42%	76.41%	77.15%	80.5%	74.0%	70.0%
rd400	0.1	6205.54	90.14%	86.08%	86.94%	85.13%	82.94%	85.30%	87.9%	85.4%	85.2%
	0.25		84.02%	79.39%	75.64%	80.12%	76.54%	72.53%	74.5%	70.9%	66.4%
	0.5		82.89%	82.40%	80.63%	71.41%	77.12%	70.80%	74.0%	72.1%	73.9%

Table 5: Average share of customers (shown as percentages) that are served by a drone. The values are shown for the GIH and SIH. The latter is shown in parentheses.

Instance	β	*One drone per vehicle*			*Two drones per vehicle*			*Three drones per vehicle*		
		$\alpha = 2.0$	$\alpha = 3.0$	$\alpha = 4.0$	$\alpha = 2.0$	$\alpha = 3.0$	$\alpha = 4.0$	$\alpha = 2.0$	$\alpha = 3.0$	$\alpha = 4.0$
One vehicle										
att532	0.1	31.0 (30.5)	33.3 (32.1)	35.5 (33.6)	42.7 (45.2)	44.4 (44.7)	47.5 (45.5)	48.7 (50.3)	48.3 (49.3)	52.3 (47.2)
	0.25	32.4 (32.9)	33.9 (34.2)	35.6 (35.1)	48.5 (47.7)	50.8 (50.2)	50.8 (51.5)	56.0 (55.7)	56.2 (55.7)	57.7 (56.6)
	0.5	31.1 (32.2)	34.1 (34.5)	35.4 (34.8)	48.8 (49.6)	51.1 (50.8)	51.3 (52.2)	56.7 (57.1)	57.8 (58.5)	58.1 (58.4)
berlin52	0.1	20.0 (16.5)	18.8 (22.0)	20.4 (20.0)	24.3 (22.4)	25.1 (24.3)	24.7 (23.9)	23.5 (24.7)	23.1 (23.5)	24.3 (24.3)
	0.25	31.8 (33.7)	35.3 (38.8)	36.9 (36.9)	42.4 (45.5)	45.1 (44.3)	46.7 (44.3)	45.9 (47.5)	46.3 (47.8)	49.8 (48.2)
	0.5	39.6 (39.6)	45.9 (44.7)	46.7 (46.3)	56.5 (52.9)	59.2 (57.6)	60.0 (61.2)	60.8 (61.2)	63.9 (62.0)	63.5 (65.1)
ch150	0.1	26.6 (28.5)	26.3 (25.5)	27.7 (27.4)	32.3 (31.1)	30.5 (25.5)	29.3 (26.4)	31.8 (30.7)	31.7 (28.9)	32.2 (31.7)
	0.25	38.3 (37.0)	42.4 (42.4)	44.4 (43.4)	54.8 (55.4)	57.7 (56.4)	57.0 (57.3)	60.1 (58.3)	60.8 (59.9)	61.3 (60.0)
	0.5	36.8 (36.1)	39.5 (38.4)	40.0 (40.3)	54.8 (55.2)	56.5 (56.6)	58.4 (58.3)	60.4 (60.5)	61.9 (63.0)	63.6 (63.1)
kroA200	0.1	32.8 (32.4)	36.6 (37.4)	35.0 (37.4)	46.4 (44.5)	44.7 (45.8)	44.6 (44.8)	49.9 (47.9)	46.6 (48.6)	48.5 (49.5)
	0.25	36.1 (35.7)	38.4 (40.1)	39.5 (38.8)	54.0 (53.6)	55.9 (55.2)	56.4 (57.6)	58.9 (58.4)	61.6 (60.1)	61.4 (60.9)
	0.5	36.0 (35.9)	37.4 (39.0)	37.9 (38.7)	54.5 (53.8)	55.0 (55.8)	55.5 (55.6)	59.8 (60.0)	61.8 (61.6)	62.5 (62.3)
rd400	0.1	30.3 (30.5)	33.1 (36.4)	33.0 (33.3)	46.4 (42.0)	42.8 (44.1)	41.6 (39.5)	49.1 (47.0)	44.8 (43.6)	48.8 (46.6)
	0.25	34.9 (34.5)	36.9 (35.8)	36.8 (37.1)	52.3 (52.4)	54.0 (54.3)	53.8 (54.2)	58.1 (57.2)	58.1 (58.6)	59.8 (57.1)
	0.5	34.3 (34.4)	35.7 (35.8)	37.1 (36.5)	51.7 (51.8)	53.6 (53.0)	54.0 (53.7)	58.4 (58.3)	59.8 (59.1)	59.7 (59.5)
Two vehicles										
att532	0.1	29.8 (31.0)	33.6 (33.0)	35.4 (34.6)	42.2 (40.5)	40.7 (46.4)	43.9 (42.7)	50.2 (49.2)	50.9 (50.2)	46.4 (46.9)
	0.25	30.0 (31.5)	34.1 (33.6)	33.6 (35.3)	47.6 (48.2)	51.1 (49.6)	50.8 (50.8)	55.2 (54.6)	56.5 (57.6)	54.9 (57.4)
	0.5	31.1 (31.7)	34.0 (33.9)	35.4 (34.5)	48.1 (48.2)	49.8 (51.7)	50.9 (51.6)	56.6 (56.3)	57.4 (58.9)	58.8 (58.9)
berlin52	0.1	18.4 (17.3)	16.1 (15.7)	16.1 (17.6)	18.8 (20.4)	22.4 (22.4)	22.7 (22.7)	22.4 (22.0)	22.0 (23.9)	22.4 (24.3)
	0.25	30.6 (32.2)	33.3 (33.7)	33.3 (35.7)	40.8 (40.0)	40.0 (42.4)	40.4 (40.8)	42.4 (43.5)	44.3 (45.1)	41.6 (45.1)
	0.5	43.1 (41.6)	42.0 (44.3)	43.9 (45.5)	54.9 (55.3)	58.0 (58.0)	58.0 (60.4)	60.0 (61.2)	62.4 (60.0)	59.6 (61.6)
ch150	0.1	22.1 (23.6)	27.2 (23.4)	25.2 (24.0)	28.5 (27.7)	25.1 (25.9)	27.9 (30.1)	29.9 (29.7)	29.4 (30.6)	30.9 (29.9)
	0.25	37.3 (39.3)	43.6 (42.6)	40.8 (44.8)	53.4 (52.2)	55.8 (55.6)	54.9 (57.0)	56.6 (57.2)	59.7 (60.7)	58.5 (59.7)
	0.5	36.5 (36.5)	39.7 (41.1)	42.6 (42.7)	54.6 (54.4)	56.5 (56.8)	57.3 (57.2)	62.3 (60.7)	64.2 (61.7)	61.6 (62.7)
kroA200	0.1	35.3 (35.2)	34.5 (36.5)	38.2 (36.2)	42.2 (42.1)	42.8 (42.0)	40.4 (39.1)	44.9 (44.9)	46.8 (44.8)	46.4 (45.2)
	0.25	37.3 (37.1)	39.2 (38.8)	40.2 (40.3)	52.9 (55.1)	56.9 (55.4)	56.3 (56.8)	58.4 (58.1)	61.8 (60.7)	59.6 (61.0)
	0.5	35.8 (34.6)	38.1 (37.1)	41.2 (38.4)	52.9 (52.0)	55.3 (54.1)	56.5 (54.7)	61.4 (59.5)	62.2 (61.1)	61.9 (62.1)
rd400	0.1	32.4 (31.9)	35.4 (33.4)	37.4 (32.2)	42.4 (43.9)	42.3 (40.7)	45.0 (43.8)	44.5 (47.1)	48.4 (47.5)	46.5 (47.7)
	0.25	35.5 (35.0)	36.5 (36.5)	37.5 (37.7)	51.5 (52.4)	52.9 (53.7)	54.7 (54.3)	58.0 (57.3)	58.7 (59.8)	58.5 (58.9)
	0.5	33.7 (34.6)	35.7 (35.4)	36.4 (36.7)	51.3 (52.1)	53.3 (53.2)	53.9 (53.4)	58.0 (58.6)	59.3 (59.1)	59.4 (60.3)
Three Vehicles										
att532	0.1	27.8 (23.9)	29.5 (32.3)	26.9 (29.2)	41.2 (41.1)	42.5 (44.1)	44.8 (45.9)	50.1 (46.9)	47.5 (43.0)	45.1 (45.8)
	0.25	29.3 (28.8)	34.3 (30.7)	35.4 (33.8)	48.5 (46.1)	50.2 (47.0)	51.1 (50.3)	54.4 (54.4)	56.6 (56.4)	56.0 (55.4)
	0.5	31.1 (31.9)	33.2 (33.7)	34.2 (34.6)	48.2 (48.9)	49.9 (50.6)	50.1 (49.5)	56.8 (54.8)	56.8 (57.3)	58.4 (58.3)
berlin52	0.1	16.5 (17.6)	16.5 (14.9)	16.1 (16.2)	21.2 (21.6)	18.1 (21.2)	22.0 (22.0)	22.1 (20.8)	22.7 (21.2)	23.5 (21.2)
	0.25	31.4 (30.4)	34.8 (32.2)	31.0 (32.9)	38.4 (40.4)	40.8 (42.7)	45.6 (42.0)	46.3 (42.0)	44.3 (44.3)	45.5 (44.3)
	0.5	42.4 (39.6)	43.9 (41.6)	45.1 (45.1)	56.5 (56.9)	55.7 (58.8)	56.9 (58.4)	58.0 (59.2)	61.2 (63.1)	61.6 (62.7)
ch150	0.1	22.7 (20.0)	23.6 (24.6)	23.1 (20.4)	28.2 (27.2)	26.2 (26.2)	25.4 (26.6)	27.4 (27.4)	28.7 (29.4)	28.1 (30.5)
	0.25	38.7 (40.3)	43.8 (43.5)	44.2 (43.5)	52.9 (52.1)	54.1 (54.2)	53.8 (54.6)	57.0 (56.4)	57.3 (57.2)	58.0 (58.1)
	0.5	39.7 (37.4)	42.7 (40.7)	44.3 (43.1)	54.4 (55.4)	58.0 (58.3)	56.9 (57.9)	60.5 (61.1)	61.1 (61.7)	63.9 (62.4)
kroA200	0.1	34.9 (30.3)	31.2 (35.9)	35.9 (35.4)	38.8 (39.7)	42.4 (40.6)	42.4 (40.5)	42.8 (41.3)	43.1 (43.0)	43.7 (44.6)
	0.25	37.2 (38.4)	40.9 (41.4)	42.0 (42.0)	54.2 (53.5)	56.0 (54.2)	57.1 (56.6)	59.1 (57.8)	60.1 (60.2)	60.6 (60.4)
	0.5	35.7 (35.7)	37.6 (39.9)	39.7 (39.4)	53.1 (54.2)	55.0 (56.0)	56.5 (56.4)	59.7 (60.4)	62.7 (62.6)	63.9 (62.4)
rd400	0.1	35.0 (33.0)	33.5 (31.8)	37.7 (36.4)	41.2 (43.2)	47.2 (42.1)	42.9 (44.9)	47.3 (45.9)	47.5 (47.6)	46.8 (47.5)
	0.25	35.6 (36.1)	36.9 (36.8)	37.8 (38.4)	52.1 (52.1)	54.2 (53.1)	53.8 (54.9)	58.7 (58.4)	59.7 (59.7)	59.1 (58.4)
	0.5	34.2 (33.7)	35.6 (36.2)	37.2 (36.3)	51.8 (51.7)	53.0 (53.1)	53.8 (54.0)	58.5 (58.1)	59.7 (60.1)	60.5 (59.3)

Table 6: A comparison of the numerical results obtained through GIH and SIH.

Vehicles	Drones	Average savings GIH	Average savings SIH	#(GIH better than SIH)	#(SIH better than GIH)
1	1	13.48%	19.72%	5	40
	2	18.40%	25.24%	5	40
	3	20.51%	26.72%	6	39
2	1	8.60%	15.64%	4	41
	2	13.03%	20.55%	3	42
	3	14.46%	21.93%	3	42
3	1	7.75%	13.11%	12	33
	2	11.31%	17.21%	7	38
	3	12.53%	18.40%	9	36
Total		13.34%	19.83%	54	351

Table 7: Comparison of the metaheuristic to the MILP solver on instances where an optimal solution was achieved. The table shows the instance specifications, the optimal objective values found through the MILP solver after t seconds (see Tables 1 and 2) and the objective values returned by the metaheuristic after t seconds.

Instance	Vehicles	Drones	α	β	MILP with LB	t [s]	SIH	t [s]	GIH	t [s]
R101	1	1	2	0.5	168.96	1.59	172.48	0.009	172.48	0.008
R101	1	1	3	0.5	168.96	1.59	172.48	0.009	172.48	0.030
R101	1	1	4	0.5	168.96	1.75	172.48	2.902	172.48	5.419
R101	1	1	2	0.75	135.24	35.02	**135.24**	0.278	**135.24**	0.087
R101	1	1	3	0.75	127.67	19.05	**127.67**	0.026	**127.67**	0.437
R101	1	1	4	0.75	127.38	20.27	**127.38**	0.372	**127.38**	2.132
R101	1	2	2	0.5	168.96	1.73	172.48	0.082	172.48	0.101
R101	1	2	3	0.5	168.96	1.63	172.48	0.323	172.48	6.124
R101	1	2	4	0.5	168.96	1.86	172.48	0.046	172.48	0.036
R101	1	2	2	0.75	123.35	17.24	**123.35**	0.137	**123.35**	2.489
R101	1	2	3	0.75	117.38	18.89	**117.38**	2.262	**117.38**	0.244
R101	1	2	4	0.75	117.38	16.22	**117.38**	1.242	**117.38**	1.039
R101	2	1	2	0.5	106.09	7.48	109.25	0.003	109.25	1.275
R101	2	1	3	0.5	106.09	12.09	109.25	0.468	109.25	1.236
R101	2	1	4	0.5	106.09	13.56	109.25	0.249	109.25	0.284
R101	2	1	2	0.75	86.39	269.65	97.02	4.788	97.02	4.581
R101	2	1	3	0.75	81.72	322.85	97.02	4.946	97.02	0.221
R101	2	1	4	0.75	80.22	216.09	97.02	5.581	97.02	2.938
R101	2	2	2	0.5	106.09	13.33	107.63	3.398	107.63	2.375
R101	2	2	3	0.5	106.09	12.99	107.63	1.080	107.63	2.288
R101	2	2	4	0.5	106.09	12.67	107.63	0.394	107.63	2.643
R101	2	2	2	0.75	81.63	312.27	82.57	2.927	82.57	0.379
R101	2	2	3	0.75	79.79	257.15	**79.79**	2.813	**79.79**	0.388
R101	2	2	4	0.75	79.79	81.72	**79.79**	0.107	**79.79**	1.303
RC101	2	2	4	0.5	84.69	1162.70	87.39	0.001	91.56	0.001

6 Conclusion

In this paper, we have studied the Vehicle Routing Problem with Drones, which consists of a combination of multiple trucks and drones in a last-mile delivery setting. We adapt a mathematical model and provide several valid inequalities to solve the problem optimally. Through our proposed valid inequalities, we are able to obtain optimal solutions through Gurobi Solver with significantly reduced runtime. Moreover, for solving large-sized instances, we propose a fast heuristic based on Variable Neighborhood Decomposition Search and two heuristic drone insertion operators that provide near-optimal solutions for small-sized problems. We provide an extensive computational study that considers the impact on the objective value in a given problem instance by using multiple vehicles, multiple drones, and several different drone parameters such as their relative velocity and flight endurance. Overall, through our computational study, we are able to show that significant savings in terms of mission time are possible through the combination of trucks and drones in last-mile delivery.

This topic provides a rich research prospect for the future. As a natural extension, it might be of interest to allow drones to be retrieved and / or launched on arcs. Another extension might consider a time-dependent recharge model. Finally, it might be of interest to consider further metrics in the objective function. On the one hand, it might be of interest to consider operational cost per mile or per delivery. On the other hand, in the context of *green VRPs*, it might also be of interest to consider reduced emissions.

Appendix: A note on the property of optimal substructure

It is also interesting to investigate whether the VRPD has an *optimal substructure*. Given a finite set S of feasible solutions x, we say that an optimization problem $\min f(x)$ has an optimal substructure, if the optimal solution $x^* \in S$ incorporates features, such that $x' \subset x^*$ is an optimal solution to a related subproblem $\min g(x)$ (Cormen et al. 2009). This property is particularly useful for solving optimization problems through dynamic programming approaches by constructing a solution to $f(x)$ through optimal solutions to its subproblems $g(x)$.

Dijkstra's algorithm is an exemplary application of solving the shortest-path problem through a dynamic programming approach by exploiting the optimal substructure of the shortest-path problem (Dijkstra 1959). In our case, applying the concept of an optimal substructure to a VRPD problem might be interpreted as follows: given an optimal solution to a VRPD problem, is it also an optimal solution to its subproblems? Let the routing of the vehicles be a subproblem to the VRPD. Then, in a given solution, if we remove the customers that are served by the drone, the remaining customers must be served in an optimal order, for the VRPD to exhibit an optimal substructure. Unfortunately, this property is not exhibited by the VRPD. With this in mind, Figure 7 shows an optimal solution to a VRPD problem with 5 vertices. In particular, depending on the values of the parameters α and β, we can see that two edges in the route of the vehicle intersect each other. It is well known that an optimal route to a basic symmetric traveling salesman or vehicle routing problem in the Euclidean metric will always be a simple polygon.

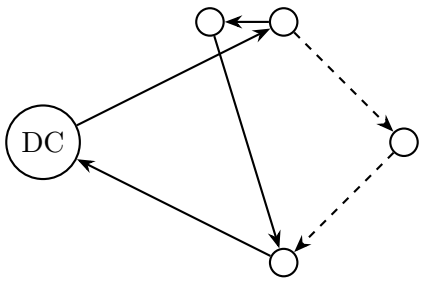

Figure 7: An optimal solution to a VRPD problem with a single vehicle (solid lines) and drone (dashed lines) for a given set of values α and β. Clearly, if one disregards the customer that is served by the drone, this solution is not optimal to the vehicle routing subproblem, as the route could be improved by removing the intersection.

References

N. Agatz, P. Bouman, and M. Schmidt. Optimization approaches for the traveling salesman problem with drone. *Transportation Science Articles in Advance*, pages 1–17, 2018. doi: 10.1287/trsc.2017.0791. URL `https://doi.org/10.1287/trsc.2017.0791`.

N. Boone, A. Sathyan, and K. Cohen. Enhanced approaches to solving the multiple traveling salesman problem. In *AIAA Infotech @ Aerospace*, pages 889–897, Reston, Virginia, 2015. American Institute of Aeronautics and Astronautics.

P. Bouman, N. Agatz, and M. Schmidt. Dynamic programming approaches for the traveling salesman problem with drone. *ERIM Report Series Reference No. ERS-2017-011-LIS*, pages 1–20, 2017.

J. Brimberg, P. Hansen, N. Mladenović, and E. D. Taillard. Improvements and comparison of heuristics for solving the uncapacitated multisource weber problem. *Operations Research*, 48(3):444–460, 2000.

J. G. Carlsson and S. Song. Coordinated logistics with a truck and a drone. *To appear in Management Science*, pages 1–31, 2017.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 2009.

R. Daknama and E. Kraus. Vehicle routing with drones. *arXiv preprint arXiv:1705.06431*, pages 1–24, 2017.

DHL International GmbH. Successful trial integration of dhl parcelcopter into logistics chain: Press release. `http://www.dhl.com/en/press/releases/releases_2016/all/parcel_ecommerce/successful_trial_integration_dhl_parcelcopter_logistics_chain.html`, 2016. Accessed: 2019-04-20.

DHL Trend Research. Unmanned aerial vehicle in logistics: A dhl perspective on implications and use cases for the logistics industry, 2014.

DHL Trend Research. Logistics trend radar: Delivering insight today. creating value tomorrow!, 2016.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski. Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85, 2017. ISSN 2168-2216.

DPDgroup International Services GmbH & Co. KG. The dpdgroup drone - parcel delivery 2.0: Datasheet. `https://www.dpd.com/home/insights/delivery_drones`, 2016. Accessed: 2019-04-20.

S. Ferrandez, T. Harbison, T. Weber, R. Sturges, and R. Rich. Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management*, 9(2): 374, 2016. ISSN 2013-0953.

M. Gendreau and J.-Y. Potvin. *Handbook of metaheuristics*, volume 2. Springer, 2010.

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

P. Hansen, B. Jaumard, N. Mladenović, and A. Parreira. Variable neighborhood search for weighted maximum satisfiability problem: Les cahiers du gerad 62: G-2000-62, 2000.

P. Hansen, N. Mladenović, and D. Perez-Britos. Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4):335–350, 2001. ISSN 13811231.

X. Jiang, Q. Zhou, and Y. Ye. Method of task assignment for uav based on particle swarm optimization in logistics. In *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, ISMSI '17, pages 113–117, New York, NY, USA, 2017. ACM.

N. Labadie, C. Prins, and C. Prodhon. *Metaheuristics for Vehicle Routing Problems*. John Wiley & Sons, 2016.

S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1 (Univ. of Calif. Press, 1967)*, pages 281–297, 1967.

N. Mathew, S. L. Smith, and S. L. Waslander. Planning paths for package delivery in heterogeneous multirobot teams. *IEEE Transactions on Automation Science and Engineering*, 12(4):1298–1308, 2015. ISSN 1545-5955.

N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11): 1097–1100, 1997. ISSN 03050548.

N. Mladenović, J. Petrović, V. Kovačević-Vujčić, and M. Čangalović. Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *European Journal of Operational Research*, 151(2):389–399, 2003. ISSN 03772217.

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

A. Otto, N. Agatz, J. Campbell, B. Golden, and E. Pesch. Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. *Networks*, 0(0):1–48, 2018. doi: 10.1002/net.21818.

S. Poikonen, X. Wang, and B. Golden. The vehicle routing problem with drones: Extended models and connections. *Networks*, 70(1):34–43, 2017. ISSN 00283045.

Pugliese, L.D.P. and Guerriero, F. Last-mile deliveries by using drones and classical vehicles. *Optimization And Decision Science*, 217:557–565, 2017.

D. Schermer, M. Moeini, and O. Wendt. Algorithms for solving the vehicle routing problem with drones. *Lecture Notes in Artificial Intelligence (Springer)*, 10751:352–361, 2018.

P. Toth and D. Vigo. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2002.

M. Ulmer and B. W. Thomas. Same-day delivery with a heterogeneous fleet of drones and vehicles, 2017.

A. van Breedam. *An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a selection of problems with Vehicle-related, Customer-related, and Time-related Constraints Contents*. Ph.d. dissertation, University of Antwerp, 1994.

X. Wang, S. Poikonen, and B. Golden. The vehicle routing problem with drones: Several worst-case results. *Optimization Letters*, 11(4):679–697, 2016. ISSN 1862-4472.

E. E. Yurek and H. C. Ozmutlu. A decomposition-based iterative optimization algorithm for traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, 91:249–262, 2018.