A Competitive Heuristic Algorithm for Vehicle Routing Problems with Drones

Xuan Ren[1], Aurélien Froger[2], Ola Jabali[3], Gongqian Liang[1]

[1] School of Management, Northwestern Polytechnical University, 710129, Xi'an, China
[2] Université de Bordeaux, CNRS, INRIA, Bordeaux INP, IMB, UMR 5251, F-33400 Talence, France
[3] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milan, Italy

Abstract

We propose a heuristic algorithm capable of handling multiple variants of the vehicle routing problem with drones (VRPD). Assuming that the drone may be launched from a node and recovered at another, these variants are characterized by three axes, 1) minimizing the transportation cost or minimizing the makespan, 2) the drone is either allowed or not allowed to land while awaiting recovery, and 3) single or multiple trucks each equipped with a drone. In our algorithm, we represent a VRPD solution as a set of customer sequences and evaluate it via local search procedures solving for each sequence a problem that we refer to as the fixed route drone dispatch problem (FRDDP). Given a sequence of customers to be served by a single truck and its drone, the FRDDP selects a subset of customers to be served by the drone and determines drone launch and recovery nodes, while ensuring that each such customer is positioned between these two nodes in the initial sequence. We introduce a heuristic dynamic program (HDP) to solve the FRDDP with reduced computational complexity compared to an exact solution algorithm for the problem. We reinforce our algorithm by developing filtering strategies based on the HDP. We benchmark the performance of our algorithm on nine benchmark sets pertaining to four VRPD variants resulting in 932 instances. Our algorithm computes 651 of 680 optimal solutions and identifies 189 new best-known solutions.

Keywords— Routing, Heuristics, Drones, Dynamic programming, Variable neighborhood search

1 Introduction

With the rapid growth of e-commerce, the global last-mile delivery market is projected to grow at a compound annual growth rate of 8.16% from 2022 to 2031 (Singh and Mutreja, 2022). To gain competitive advantages, last-mile delivery companies have to actively deal with the increasing demand for quantity and quality of service. Unmanned aerial vehicles (UAVs), generally known as drones, have shown significant market potential in last-mile delivery as an emerging technology (Sah et al., 2021).

Compared to trucks, a major advantage of drones is that they are not restricted by road infrastructure (Poikonen and Campbell, 2021). As traffic congestion is one of the most critical challenges in urban areas, drones may complete delivery tasks faster than trucks, especially in metropolises (Salama and Srinivas, 2022). Furthermore, drones are flexible and easy to operate, which may decrease labor costs (Otto et al., 2018). Besides, since they are battery-powered, drones are relatively energy-efficient (Kirschstein, 2020; Salama and Srinivas, 2020). However, drones still have shortcomings such as a small payload capacity, limited flight range, and susceptibility to interference, so they cannot meet the needs of modern logistics on their own. As a result, the current projected use of drones in last-mile delivery is in collaboration with trucks (Campbell et al., 2017).

Murray and Chu (2015) were the first to investigate the collaboration between a truck and a drone in a logistics context. They proposed a new variant of the traveling salesman problem (TSP) called the flying sidekick traveling salesman problem (FSTSP). In particular, a scenario where a truck delivers goods to customers together with a single drone is considered. The drone may be dispatched to serve a customer while the truck serves other customers simultaneously. Once the drone completes its delivery task, it must return to the truck. The temporal alignment between the drone and the truck along a route can be viewed as a form of *synchronization* between two vehicles (Drexl, 2012). Murray and Chu (2015) considered a makespan minimization objective, that is the maximum arrival time to the depot between the truck and the drone. As such, the makespan includes the time required for serving customers, as well as the potential waiting time induced by the synchronization between the truck and the drone.

The FSTSP has attracted increasing attention in the past years. We refer the reader to the surveys of Macrina et al. (2020); Li et al. (2021); Chung et al. (2020); Moshref-Javadi and Winkenbach (2021). Notably, a number of variants have been proposed. The generalization of the FSTSP to multiple trucks, each equipped with one or more drones, is commonly referred to as the vehicle routing problem with drones (VRPD). This problem, which can be seen as an extension of the vehicle routing problem (VRP), has been studied by several authors (e.g., Poikonen et al., 2017; Wang et al., 2017; Tamke and Buscher, 2021; Schermer et al., 2019a; Zhou et al., 2023). In addition to the objective of minimizing makespan, several authors have considered the VRPD with a cost minimization objective (e.g., Sacramento et al., 2019; Wang and Sheu, 2019). The considered costs typically include those related to both trucks and drones.

Due to the NP-hardness of the VRPD and its variants (Murray and Chu, 2015), existing exact approaches can only solve small-sized instances (e.g., Tamke and Buscher, 2021; Roberti and Ruthmair, 2021; Zhou et al., 2023). Therefore, a fair amount of the literature on the VRPD and its variants proposes solution methods based on meta-heuristics, such as adaptive large neighborhood search (ALNS) (Sacramento et al., 2019), genetic algorithm (Ha et al., 2020), and variable neighborhood search (VNS) (Freitas and Penna, 2020).

The objective of this paper is to propose an efficient solution method capable of handling several variants of the VRPD. In particular, we study the VRPD with two different objective functions: minimization of transportation cost and minimization of the makespan. Hereinafter, we refer to these two problems as VRPD-C and VRPD-M. Furthermore, our method can address situations where the drone is either allowed or not allowed to land while awaiting recovery. Lastly, our method may be used for single truck as well as multiple truck variants of the VRPD-C and VRPD-M. In our algorithm, we represent a VRPD solution as a set of customer sequences and evaluate it using local search procedures solving for each sequence a problem that we refer to as the fixed route drone dispatch problem (FRDDP). Given a sequence of customers to be served by a single truck and its drone, the FRDDP selects a subset of customers for drone service and determines drone launch and recovery nodes for each selected customer, while ensuring that each such a customer is positioned between these two nodes in the initial sequence. We note that the FRDDP is instrumental to evaluating possible solutions in most local search-based heuristic algorithms for the VRPD and its variants. Given the vast landscape of such variants, we believe that our algorithmic contribution for the FRDDP may significantly accelerate the solution process of a number of VRPD configurations. We summarize the main contributions of this paper as follows.

(i) We study the FRDDP with two different objective functions, that is minimizing the transportation cost (FRDDP-C) and minimizing the makespan (FRDDP-M). For each problem, we propose a heuristic dynamic program (HDP) with a computational complexity of $\mathcal{O}(n^2)$ (where n is the number of customers contained in a single route). This is in contrast to exact dynamic programs (DPs) for the FRDDP with a computational complexity of $\mathcal{O}(n^3)$ (e.g., Agatz et al., 2018).

(ii) We propose an efficient hybrid variable neighborhood search (HVNS) algorithm that embeds our HDP to solve the VRPD. The HVNS combines a shaking step, variable neighborhood descent (VND), and a simulated annealing-based acceptance criterion. To accelerate the search in the VND, we also develop a sub-heuristic dynamic program (subHDP) procedure to filter non-promising moves. The subHDP utilizes the principles of the HDP by adapting the computations performed for the HDP to approximate objective function savings.

(iii) We demonstrate the effectiveness of our algorithm via a large set of computational experiments. We first compare the performance of the HVNS using our HDP and subHDP with the HVNS using an exact DP. Given the same computational runtime budget, the HVNS using our HDP and subHDP outperforms the HVNS using an exact DP in terms of the objective function value, especially when the number of customers increases. Moreover, given the same number of iterations, the solution quality obtained by the HVNS using an exact DP is slightly better than that of the HVNS using our HDP and subHDP, while requiring on average more than 30 times the computational time. Furthermore, we compare our method with state-of-the-art algorithms on the VRPD with both objective functions. For the VRPD-C, out of a total of 112 instances, our algorithm identifies 48 new best-known solutions and matches 56 solutions in comparison with the benchmark results from Sacramento et al. (2019) and Rave et al. (2023). For the VRPD-M, out of a total of 820 instances, our algorithm yields 615 out of 644 optimal solutions and identifies 141 new best-known solutions in comparison with the benchmark results from El-Adle et al. (2021), Tamke and Buscher (2021), Roberti and Ruthmair (2021), and Kitjacharoenchai et al. (2019).

The remainder of this paper is organized as follows. In Section 2 we review the related literature. We define the treated problems and considered assumptions in Section 3. Considering that the FRDDP is a key component of our solution method, we formally introduce the FRDDP and the HDPs for both objective functions in Section 4. In Section 5 we describe the HVNS along with the subHDP, and present its computational results in Section 6. Finally, in Section 7 we draw conclusions and discuss possible future developments of our work.

2 Related Literature

The introduction of the FSTSP by Murray and Chu (2015) has paved the way for subsequent contributions on the collaborative use of trucks and drones for parcel delivery. In this section, we present the research most relevant to this work, specifically focusing on VRPD studies that consider the synchronization between a truck and its assigned drone. Such synchronization refers to the truck and its paired drone working in tandem during the delivery process. We excluded works addressing emerging VRPD variants, such as multiple drones carried by a truck, or drones not working in tandem with a truck. Despite these variants being undoubtedly valuable and contributing to the border landscape of VRPD research, our main goal is to investigate the problem characteristics, objective functions, and solution methods pertaining to synchronized collaboration. For a comprehensive survey of models, applications, and emerging VRPD variants, the reader is referred to Macrina et al. (2020); Li et al. (2021); Chung et al. (2020); Moshref-Javadi and Winkenbach (2021). In particular, we first discuss problem characteristics and objective functions in Section 2.1, and then describe the main solution methods used in solving the resulting problems in Section 2.2.

2.1 Problem characteristics and objective functions

Agatz et al. (2018) present a variant of the FSTSP called the traveling salesman problem with drone (TSPD). Similar to Murray and Chu (2015), the authors assume that the drone has limited capacity and can only serve one customer at each dispatch. However, they also assume that the drone can be launched from the same location multiple times, while the truck remains stationary for the drone to return. This is called a *cyclic* operation (Schermer et al., 2019b) or a *loop* (Roberti and Ruthmair, 2021). We observe that the FSTSP is characterized by a slightly different assumption, as loops are prohibited. In addition, Agatz et al. (2018) and Bouman et al. (2018a) also allow the truck to revisit nodes for drone recovery, which is prohibited in most of the literature (e.g., Murray and Chu, 2015; Sacramento et al., 2019; Roberti and Ruthmair, 2021; Dell'Amico et al., 2021). Roberti and Ruthmair (2021) study several variants of the TSPD, one of which requires that the drone cannot land and wait (NoLW) to be recovered by the truck. Since the limited battery capacity of the drone impacts its flight endurance, this assumption implies that the truck must recover the drone before its battery runs out. Thus, the time between launching and recovering the drone is more constrained. A similar assumption is also used by Murray and Chu (2015); Sacramento et al. (2019); Rave et al. (2023). However, this is not the case in several works (e.g., Agatz et al., 2018; El-Adle et al., 2021; Tamke and Buscher, 2021), where the drone is allowed to land and wait (LW) for recovery.

Wang et al. (2017) and Poikonen et al. (2017) introduce the VRPD by extending the problem from a single truck to multiple trucks. Both works analyze several worst-case scenarios and introduce bounds on the best possible time savings of using trucks and drones, as opposed to using trucks alone. Schermer et al. (2019b) formulate a mixed integer linear program (MILP) model along with valid inequalities for the VRPD, where cyclic operations are allowed, i.e., drones can be launched and recovered at the same node.

As one of the main advantages of using drones is their higher speed when compared to trucks, most of the aforementioned studies considered makespan minimization as an objective function. However, as operational costs also play an important role in logistics, several authors considered optimizing them in VRPDs. Ha et al. (2018) were the first to consider a cost minimization objective in a TSPD model. The considered costs include the transportation costs (of both the truck and the drone), as well as a waiting time penalty for the truck and for the drone. Sacramento et al. (2019) minimize the transportation cost of trucks and drones for the VRPD. Wang and Sheu (2019) include fixed cost of deploying trucks in the VRPD.

With exception of Ha et al. (2018); Roberti and Ruthmair (2021); Tamke and Buscher (2021), the aforementioned contributions primarily deal with a single problem. Considering non-cyclic operations, our method is capable of handling multiple VRPD variants, characterized by the following three axes: 1) objective functions of minimizing the transportation cost or minimizing the makespan, 2) the drone is allowed to land while awaiting recovery (i.e., LW) or not allowed (i.e., NoLW), and 3) single or multiple trucks each equipped with a single drone. Among the possible combinations of these axes, four variants have been extensively studied in the literature. We compare the performance of our algorithm on those four variants in Section 6.

2.2 Solutions methods

Compact MILP models for the FSTSP and its variants are capable of solving very small instances typically not exceeding 12 customers (e.g., Murray and Chu, 2015; Sacramento et al., 2019). Bouman et al. (2018a) develop a dynamic program with three steps for the TSPD, which solves instances with up to 20 nodes to optimality. El-Adle et al. (2021) propose a mixed integer program (MIP) model for the TSPD, which is enhanced by a series of bound improvement strategies. The authors are able to solve instances with 32 customers. Several authors focused on developing exact methods for the FSTSP and its variants, such as branch-and-bound algorithms (Poikonen et al., 2019), branch-and-price (B&P) algorithms (Roberti and Ruthmair, 2021; Zhou et al., 2023), branch-and-cut (B&C) algorithms (Tamke and Buscher, 2021), and branch-price-and-cut algorithms (Zhen et al., 2023). Among these works, notably, Roberti and Ruthmair (2021) solve instances with up to 40 nodes to optimality.

To solve instances with a larger number of customers, several works propose metaheuristic approaches. Ha et al. (2020) develop a hybrid genetic algorithm (HGA) to solve two versions of the TSPD with minimizing cost and minimizing makespan objective functions. The authors test their HGA on three benchmark sets (Murray and Chu, 2015; Ha et al., 2018; Freitas and Penna, 2020), showing that their HGA outperforms other methods in the literature in terms of solution quality. Moshref-Javadi et al. (2020) present a hybrid tabu search-simulated annealing algorithm for the case of a single truck equipped with multiple drones. The authors solve real-world-size problem instances with up to 159 customers. Sacramento et al. (2019) propose an ALNS with several destroy operators and repair operators for the VRPD. The authors test their ALNS on instances with up to 200 customers considering realistic values for the used parameters.

The heuristics in the aforementioned works generally employ local search operators that do not alter the given assignment of each customer to the truck or to the drone. However, there are a series of works that reoptimize these assignments for each evaluated solution. Murray and Chu (2015) propose a route and re-assign heuristic. The route step constructs a TSP tour containing all the customers served by the truck. Then the re-assign step assigns a subset of customers to be served by the drone by evaluating the achievable time saving of the drone insertion. When the drone is selected to serve a customer, the algorithm determines its launch node and recovery node, ensuring that the former is positioned before the latter in the TSP tour.

Such a re-assign step is slightly restricted by Agatz et al. (2018), who consider a potential visit of a customer by the drone to be such that the drone can only be launched at a node preceding the customer and recovered at a node succeeding the customer. The re-assign step proposed by Agatz et al. (2018) is the case of what we refer to as the FRDDP. The authors propose a greedy heuristic and an exact method based on a DP for the FRDDP. Ha et al. (2018) introduce a greedy randomized adaptive search procedure (GRASP) that embeds an exact DP to solve the FRDDP, which is only used once in each iteration of the GRASP to obtain a starting solution, and is not considered during the improvement phase. Najy et al. (2022) consider a FRDDP in an inventory-routing problem with a single truck and a single drone. In particular, they solve the FRDDP as a shortest path problem in a network that defines arcs between customers that are possibly served by the drone. Schermer et al. (2019b) decompose the VRPD into two subproblems. The first subproblem consists of allocating customers to routes and sequencing these customers within each route, while the second subproblem consists of an independent FRDDP for each route. The authors solve the first subproblem with a classical heuristic for VRPs and the second subproblem through a MILP model.

The existing heuristics for the FSTSP and its variants can be broadly classified into two main categories. The first includes methods purely applying local search operators on the truck and drone solutions (e.g., Ha et al., 2020; Sacramento et al., 2019; Freitas and Penna, 2020; Rave et al., 2023), whereas the second applies local search operators to truck routes, and evaluates those truck routes by inserting drone visits (i.e., solving an appropriately defined version of the FRDDP). However, exactly solving the FRDDP at every evaluation of a local search move is computationally expensive. To handle this issue, our method uses a heuristic DP during the local search in combination with filtering procedures on non-promising moves. These features dramatically improve the local search.

3 Problem Definition

In this section, we define the treated problems along with their considered assumptions. We begin by introducing common notation primarily following that of Sacramento et al. (2019). We then discuss the main modeling assumptions for all treated problems.

The VRPD is defined on a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \{0, 1, ..., n+1\}$ is the set of nodes, nodes 0 and $n+1$ represent the departure and arrival depot, and $\mathcal{C} = \{1, 2, ..., n\}$ is the set of nodes representing customers. The set of arcs is denoted by $\mathcal{A} = \{(i,j) : i \in \mathcal{N} \backslash \{n+1\}, j \in \mathcal{N} \backslash \{0\}\}$. We consider a fleet set $\mathcal{H}$ containing m homogeneous trucks. Each truck $h \in \mathcal{H}$ has a capacity of Q and carries a single drone, with a capacity of Q'. Each customer $i \in \mathcal{C}$ has a demand q_i that must be served in a single visit (i.e., split deliveries are not allowed). Customer i is considered to be *drone-eligible* if $q_i \leqslant Q'$. We define $\mathcal{C}' \subseteq \mathcal{C}$ as the set of drone-eligible customers that are possible to be served by the drone.

Considering that trucks and drones have different speeds, the travel times for a truck and for a drone in arc $(i,j) \in \mathcal{A}$ are denoted by τ_{ij} and τ'_{ij}. The service times of a truck and of a drone at node $i \in \mathcal{N}$ are s_i and s'_i. The drone has a flight endurance T', and needs $\hat{\tau}$ and $\check{\tau}$ time to be launched and recovered.

Due to capacity limitations, the drone can only serve one customer at each dispatch. We define a drone *sortie* as $\langle i,j,k \rangle$, where the drone takes off from the truck at node i, serves customer j, and subsequently returns to the same truck at node k. In such cases, we refer to node i and node k as a *launch node* and a *recovery node*. Let $\Delta = \{\langle i,j,k \rangle : i \in \mathcal{N} \backslash \{n+1\} \wedge j \in \mathcal{C}' \wedge k \in \mathcal{N} \backslash \{0\} \wedge i \neq j \wedge j \neq k \wedge k \neq i \wedge \hat{\tau} + \tau'_{ij} + s'_j + \tau'_{jk} + \check{\tau} \leqslant T'\}$ be the set of all possible drone sorties. A drone sortie is deemed possible (and inserted in Δ) if the flight endurance of the drone is respected. If a drone does not have any delivery task, it remains on its truck. The duration of each route is defined as the maximum between the time the truck returns to the depot and the time its drone is recovered at the depot. The maximum duration of

Notation	Type	Description
x_{ij}^h	Binary	1, if truck $h \in \mathcal{H}$ travels from node $i \in \mathcal{N}\backslash\{n+1\}$ to node $j \in \mathcal{N}\backslash\{0\}$; 0, otherwise
y_{ijk}^h	Binary	1, if the drone on truck $h \in \mathcal{H}$ performs the sortie $\langle i,j,k \rangle \in \Delta$; 0, otherwise
p_i^h	Integer	the position of node $i \in \mathcal{N}$ in the route of truck $h \in \mathcal{H}$
w_{ij}^h	Binary	1, if node $j \in \mathcal{N}\backslash\{0\}$ is visited after node $i \in \mathcal{N}\backslash\{n+1\}$ in the route of truck $h \in \mathcal{H}$; 0, otherwise
t_i^h	Continuous	ready time of truck $h \in \mathcal{H}$ at node $i \in \mathcal{N}$ to launch a drone, serve a customer or arrive at the depot
$t_i'^h$	Continuous	ready time of the drone on truck $h \in \mathcal{H}$ to start serving customer $i \in \mathcal{C}'$, the time at which it starts being launched from $i \in \mathcal{N}\backslash\{n+1\}$, the time at which it terminates being recovered at $i \in \mathcal{N}\backslash\{0\}$.

Table 1: Decision variables

each route is T.

In the VRPD-C, each arc $(i,j) \in \mathcal{A}$ is associated with two non-negative transportation costs, c_{ij} for a truck and c'_{ij} for a drone. The objective is to minimize the total transportation cost of both trucks and drones. In the VRPD-M, the objective is to minimize the makespan, which represents the maximum arrival time at the depot of all the trucks and their paired drones.

In what follows we summarize the main modeling assumptions of the treated problems.

(i) The truck and its carried drone are paired. Thus, the drone can only be launched from and recovered at its paired truck. This assumption is primarily motivated by safety reasons. Furthermore, we assume that after the drone returns to its truck, its battery is assumed to be immediately replaced by a full one. These two assumptions are commonly used in the literature (e.g., Sacramento et al., 2019; Rave et al., 2023; Roberti and Ruthmair, 2021).

(ii) For the synchronization between the truck and its paired drone, we consider either the NoLW or the LW assumption. The NoLW assumption is considered in many works (e.g., Sacramento et al., 2019; Rave et al., 2023; Roberti and Ruthmair, 2021). It entails that if the drone waits for recovery, it waits while hovering in the air (i.e., consuming flight endurance T') until the truck arrives at the recovery location. The LW assumption implies that the drone is allowed to land and wait for recovery (e.g., Agatz et al., 2018; El-Adle et al., 2021).

(iii) The truck and the drone must depart from and return to the depot. As assumed in many papers (e.g., Sacramento et al., 2019; Rave et al., 2023; Roberti and Ruthmair, 2021), the drone can only be launched from a node $i \in \mathcal{N}\backslash\{n+1\}$ and recovered at a node $k \in \mathcal{N}\backslash\{0\}$ which is different from i. In other words, the truck must recover the drone at a node other than the one from which it was launched. This assumption is primarily motivated by the difficulty of having trucks parked for a long time in urban areas.

(iv) As we consider the drone launch time and recovery time, as well as the truck service time, we assume that when the drone is launched from a truck to a customer node, it must be launched before the truck starts serving the customer. Furthermore, when the truck recovers the drone at a customer, it must do so before serving this customer, even if it arrives at this customer earlier than the drone. This assumption prioritizes the drone's tasks and is considered in Sacramento et al. (2019) and Rave et al. (2023). Notably, this assumption is not required when drone launch time, drone recovery time, and truck service time are ignored as in Tamke and Buscher (2021); El-Adle et al. (2021); Roberti and Ruthmair (2021).

In Section A of the supplementary material, we formulate every variant of the VRPD considered in this work as a MILP model using the decision variables listed in Table 1. The model for the VRPD-C corresponds to that introduced by Sacramento et al. (2019).

4 Fixed Route Drone Dispatch Problem

Our VRPD heuristic algorithm (described in Section 5) explores local search neighborhoods based on a representation of every solution as a set of truck routes, i.e., sequences of customer to be served by either a truck or its paired drone. Evaluating such sequences in our heuristic amounts to solving the fixed route drone dispatch problem (FRDDP). Given a sequence of customers to be served by a single truck and its drone, the FRDDP determines drone-served customers and the drone launch and recovery nodes. The drone should be launched at a node positioned before the customer it serves in the sequence and recovered at a node positioned after the customer it serves. Efficiently solving the FRDDP is essential to the overall effectiveness of our heuristic.

We present two exact dynamic programs and develop heuristic dynamic programs to address the FRDDP with both considered objectives. In the following, we first present common definitions and notation in Section 4.1 to describe the FRDDP. We also discuss how to address LW and NoLW assumptions. We then present two exact dynamic programs in Section 4.2. We note that the dynamic programming formulation for the FRDDP-M with the LW assumption was previously proposed by Agatz et al. (2018). In Section 4.3, we propose two heuristic dynamic programs for solving the FRDDP-C and FRDDP-M.

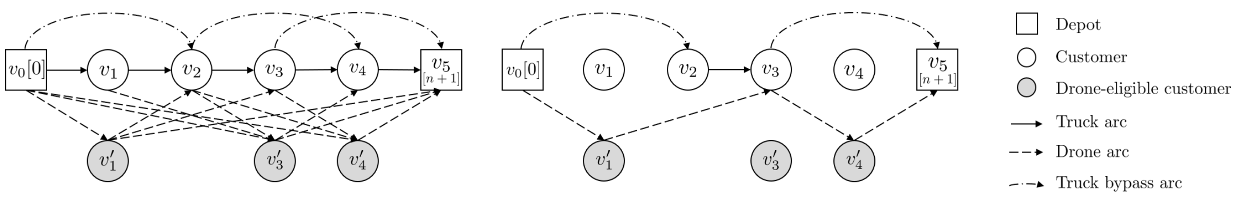
4.1 Problem statement and notation

Let $\mathcal{R} = (v_0, v_1, ..., v_{\bar{n}-1}, v_{\bar{n}})$ denote the sequence of nodes visited by a truck, where v_0 and $v_{\bar{n}}$ represent the departure and arrival depot (i.e., $v_0 = 0$ and $v_{\bar{n}} = n+1$) and v_e for $e \in \{1, ..., \bar{n}-1\}$ is a customer node (i.e., $v_e \in \mathcal{C}$). We refer to $\mathcal{R}$ as a *truck route.*

Given a truck route $\mathcal{R}$, we define a directed acyclic graph $\mathcal{G}_{\mathcal{R}} = (\mathcal{N}_{\mathcal{R}}, \mathcal{A}_{\mathcal{R}})$ for the FRDDP. The node set $\mathcal{N}_{\mathcal{R}} = \{v_e \in \mathcal{R} \cup \mathcal{R}'\}$ includes two distinct sets of nodes: a set $\mathcal{R}$ consisting of all nodes from the original route $\mathcal{R}$ and a new set $\mathcal{R}'$ consisting of all drone-eligible customers in $\mathcal{R}$. To distinguish between the customers in $\mathcal{R}$ and $\mathcal{R}'$, we use $v'_f \in \mathcal{R}'$ when referring to a drone-eligible customer v_f. Specifically, $\mathcal{R}' = \{v'_f \in \mathcal{R} : \exists \langle v_e, v_f, v_g \rangle \in \Delta\}$. Thus, v'_f is possibly neither launched nor recovered from a node in $\mathcal{R}$. The set of arcs $\mathcal{A}_{\mathcal{R}}$ is defined as follows:

$$\mathcal{A}_{\mathcal{R}} = \left\{ \begin{array}{ll} (v_e, v_{e+1}) : v_e \in \mathcal{R} \backslash \{n+1\} & \text{(1a)} \\ (v_e, v'_f) : 0 \leqslant e < f < \bar{n}, \exists k \in \mathcal{N} \backslash \{0\} \text{ with } \langle v_e, v_f, k \rangle \in \Delta & \text{(1b)} \\ (v'_f, v_g) : 0 < f < g \leqslant \bar{n}, \exists i \in \mathcal{N} \backslash \{n+1\} \text{ with } \langle i, v_f, v_g \rangle \in \Delta & \text{(1c)} \\ (v_e, v_{e+2}) : v_e \in \mathcal{R} \backslash \{n+1, v_{\bar{n}-1}\} \wedge v'_{e+1} \in \mathcal{R}' & \text{(1d)} \end{array} \right\}$$

We provide an example of $\mathcal{G}_{\mathcal{R}}$ given a truck route $\mathcal{R} = (v_0 = 0, v_1, v_2, v_3, v_4, v_5 = n+1)$ in Figure 1 (a). The nodes in route $\mathcal{R}$ are marked with white circles (customers) or white squares (the departure and arrival depot) and all drone-eligible customers of set $\mathcal{R}' = (v'_1, v'_3, v'_4)$ are marked with gray circles. Arcs are defined according to conditions (1). The *truck arcs* between adjacent nodes in $\mathcal{R}$, as described in conditions (1a), are marked with solid arrows. The *drone arcs*, described in conditions (1b)–(1c) and marked with dashed arrows, are between a node v_e and a drone-eligible customer v'_f, and between a drone-eligible customer v'_f and a node v_g. As previously mentioned, under conditions (1b)–(1c) nodes k and i may not be in route $\mathcal{R}$. However, the combination of conditions (1b)–(1c) ensures that drone arcs represent feasible launch and recovery arcs. The *truck bypass arcs* (v_e, v_{e+2}), described in condition (1d) and marked with dot-dash arrows, model the situations where the truck traverses directly from node v_e to node v_{e+2} when customer v'_{e+1} is served by the drone. This is due to the fact that the drone can only serve one customer at each dispatch. Note that if customer v'_{e+1} is not drone-eligible, there is no truck bypass arc between node v_e and v_{e+2}. Therefore, since customer v_2 cannot be served by a drone, there is no arc between customer v_1 and v_3.



(a) A graph for a truck route $\mathcal{R}$ (b) A complete route given a truck route $\mathcal{R}$

Figure 1: Illustration of $\mathcal{G}_{\mathcal{R}}$ and a complete route for a truck route $\mathcal{R}$ serving four customers

We define an FRDDP solution as a *complete route* and denote it by $\overline{\mathcal{R}} = (\mathcal{R}^t, \mathcal{R}^d)$, where $\mathcal{R}^t$ is a sequence of nodes starting at 0 and ending at $n+1$, containing all the customers of $\mathcal{R}$ that are served by the truck, and $\mathcal{R}^d$ contains a sequence of drone sorties. Figure 1 (b) shows a complete route for the truck route $\mathcal{R}$ given in Figure 1 (a). The drone serves customer v_1 after being launched at the depot. The truck departs from the depot to serve customer v_2, bypassing customer v_1. Then, the drone is recovered by the truck at customer v_3 and launched again to serve customer v_4 before being recovered at the depot, while the truck returns to the depot after serving customer v_3 bypassing customer v_4. Accordingly, we have $\mathcal{R}^t = (0, v_2, v_3, n+1)$ and $\mathcal{R}^d = (\langle 0, v_1, v_3\rangle, \langle v_3, v_4, n+1\rangle)$.

For notational convenience, in this section, we denote by c_{ef} and c'_{ef} the cost of the truck and the drone traveling from v_e to v_f in truck route $\mathcal{R}$. We then define C_{ef} as the total truck cost incurred if the truck visits the nodes of subsequence $(v_e, v_{e+1}, ..., v_f)$. The value C_{ef} is computed as follows.

$$C_{ef} = \begin{cases} 0 & e = f \\ c_{ef} & f = e+1 \\ C_{e,f-1} + c_{f-1,f} & f > e+1 \end{cases} \tag{2}$$

Let τ_{ef} and τ'_{ef} be the travel time of the truck and the drone from v_e to v_f in truck route $\mathcal{R}$, and let s_e and s'_e be the service time of the truck and the drone serving customer v_e in $\mathcal{R}$. We then define T_{ef} as the sum of the truck's travel and service time for a subsequence $(v_e, v_{e+1}, ..., v_f)$ as follows.

$$T_{ef} = \begin{cases} 0 & e = f \\ s_e + \tau_{ef} & f = e+1 \\ T_{e,f-1} + s_{f-1} + \tau_{f-1,f} & f > e+1 \end{cases} \tag{3}$$

Given a truck route $\mathcal{R}$, we define $\langle v_e, v_f, v_g\rangle$ with $v_e, v_f, v_g \in \mathcal{R}$ as a possible drone sortie if it satisfies the following conditions:

$$(v_e, v'_f) \in \mathcal{A}_{\mathcal{R}} \tag{4a}$$
$$(v'_f, v_g) \in \mathcal{A}_{\mathcal{R}} \tag{4b}$$
$$\langle v_e, v_f, v_g \rangle \in \Delta \tag{4c}$$
$$\hat{\tau} + \sum_{l=e}^{g-1} (\tau_{l,l+1} + s_l) + \check{\tau} \leqslant T' \tag{4d}$$

Condition (4d) ensures the NoLW assumption where the drone must be recovered by the truck within the drone endurance T'. In the case of the LW assumption, we remove condition (4d).

4.2 Exact dynamic programs

In this section, we first introduce an exact dynamic program (EDP) for the FRDDP-C, and then present an EDP for FRDDP-M. Let $\omega^{\mathsf{E}}_{\mathsf{C}}(v_g)$ and $\omega^{\mathsf{E}}_{\mathsf{M}}(v_g)$ denote the minimum cost and minimum makespan up to node v_g when visited by a truck while the drone is either on the truck or has just been recovered at v_g.

The EDP for the FRDDP-C is formulated in equation (5). Two cases are considered when computing $\omega^{\mathsf{E}}_{\mathsf{C}}(v_g)$ (where $g \geqslant 2$). In the first case, the truck arrives from node v_{g-1} while carrying the drone. In the second case, the drone is recovered by the truck at node v_g, having served one of the previous nodes of the route. This case is illustrated in Figure 2, where unvisited customers and untraveled arcs are marked in gray. The computation considers every possible couple of nodes (v_e, v'_f) in $\mathcal{G}_{\mathcal{R}}$ that can form a feasible drone sortie $\langle v_e, v_f, v_g \rangle$ (i.e., a drone sortie satisfying conditions (4a)–(4d)), and includes the minimum cost up to node v_e, the cost of the drone $c'_{ef} + c'_{fg}$, and the cost of the truck $C_{e,f-1} + c_{f-1,f+1} + C_{f+1,g}$ bypassing customer v_f. The computational complexity of this EDP is $\mathcal{O}(\bar{n}^3)$.

$$\omega^{\mathsf{E}}_{\mathsf{C}}(v_g) = \begin{cases} 0 & g = 0 \\ c_{01} & g = 1 \\ \min\Big\{\omega^{\mathsf{E}}_{\mathsf{C}}(v_{g-1}) + c_{g-1,g}, \min\limits_{v_e, v'_f : \langle v_e, v_f, v_g \rangle \text{ subject to (4a)–(4d)}} \big\{\omega^{\mathsf{E}}_{\mathsf{C}}(v_e) & \\ \quad + c'_{ef} + c'_{fg} + C_{e,f-1} + c_{f-1,f+1} + C_{f+1,g}\big\}\Big\} & g \in \{2, \ldots, \bar{n}\} \end{cases} \tag{5}$$

An EDP for the FRDDP-M is presented in equation (6). Slightly different from the FRDDP-C, truck and drone time synchronization is explicitly handled in the FRDDP-M. Specifically, the minimum makespan at node v_g is the maximum between the arrival time of the truck and the arrival time of the drone at that node. The computational complexity of this EDP is $\mathcal{O}(\bar{n}^3)$.

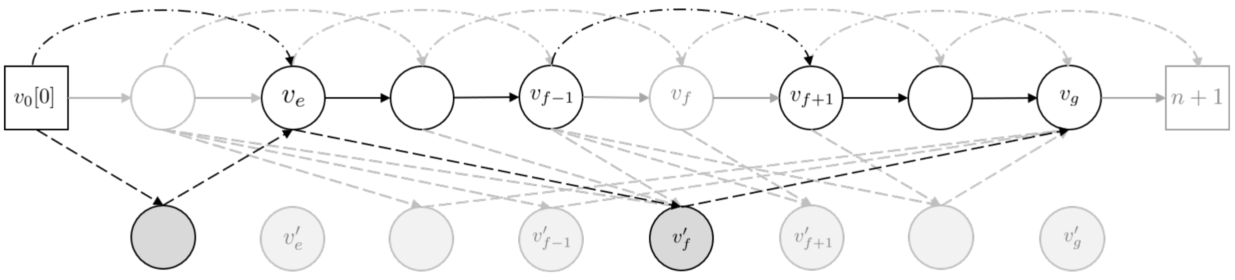


Figure 2: Illustration of the computation performed in the EDP

$$\omega^{\mathsf{E}}_{\mathsf{M}}(v_g) = \begin{cases} 0 & g = 0 \\ \tau_{01} & g = 1 \\ \min\Big\{\omega^{\mathsf{E}}_{\mathsf{M}}(v_{g-1}) + T_{g-1,g}, \min\limits_{v_e, v'_f : \langle v_e, v_f, v_g\rangle \text{ subject to (4a)–(4d)}} \Big\{\omega^{\mathsf{E}}_{\mathsf{M}}(v_e) + \hat{\tau} & \\ \quad + \max\{\tau'_{ef} + s'_f + \tau'_{fg}, T_{e,f-1} + s_{f-1} + \tau_{f-1,f+1} + T_{f+1,g}\} + \check{\tau}\Big\}\Big\} & g \in \{2, \ldots, \bar{n}\} \end{cases} \tag{6}$$

4.3 Heuristic dynamic programs

In this section, we develop novel heuristic dynamic programs for both objective functions of the FRDDP. In Section 4.3.1, we introduce the HDP for the FRDDP-C, and in Section 4.3.2, we introduce the HDP for the FRDDP-M.

4.3.1 FRDDP-C

We define $\omega^{\mathsf{H}}_{\mathsf{C}}(v_g)$ as the minimum cost up to node v_g when visited by a truck while the drone is either on the truck or has just been recovered at v_g. We define $\omega^{\mathsf{H}}_{\mathsf{C}}(v'_f)$ as the minimum cost up to the drone-eligible customer v'_f, where the cost for the truck up to v_{f-1} is included. The values $\omega^{\mathsf{H}}_{\mathsf{C}}(v_g)$ and $\omega^{\mathsf{H}}_{\mathsf{C}}(v'_f)$ are computed recursively by equations (7) and (8). We denote by $p_{\mathsf{C}}(v'_f)$ the index of the launch node leading to the minimum cost when computing $\omega^{\mathsf{H}}_{\mathsf{C}}(v'_f)$.

$$\omega^{\mathsf{H}}_{\mathsf{C}}(v_g) = \begin{cases} 0 & g = 0 \\ c_{01} & g = 1 \\ \min\Big\{\omega^{\mathsf{H}}_{\mathsf{C}}(v_{g-1}) + c_{g-1,g}, \min\limits_{v'_f : \langle v_{p_{\mathsf{C}}(v'_f)}, v_f, v_g\rangle \text{ subject to}} (4b) - (4d) \Big\{\omega^{\mathsf{H}}_{\mathsf{C}}(v'_f) & \\ \quad + c'_{fg} + c_{f-1,f+1} + C_{f+1,g}\Big\}\Big\} & g \in \{2, \ldots, \bar{n}\} \end{cases} \tag{7}$$

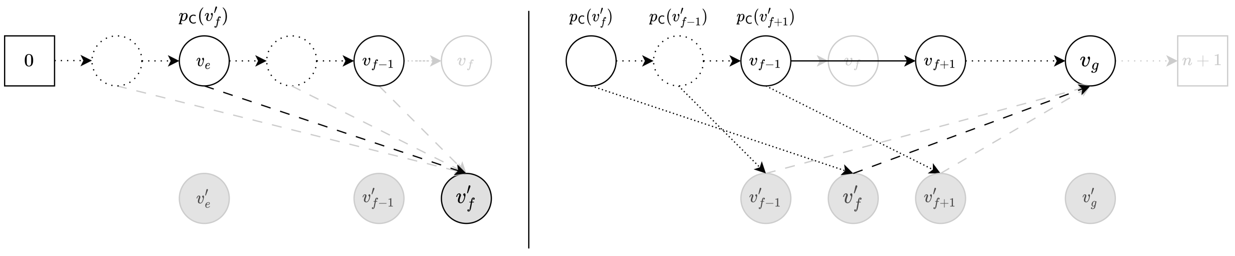
$$\omega^{\mathsf{H}}_{\mathsf{C}}(v'_f) = \min_{v_e : (v_e, v'_f) \in \mathcal{A}_{\mathcal{R}}} \Big\{\omega^{\mathsf{H}}_{\mathsf{C}}(v_e) + c'_{ef} + C_{e,f-1}\Big\} \qquad v'_f \in \mathcal{R}' \tag{8}$$

where, $$p_{\mathsf{C}}(v'_f) = \operatorname*{arg\,min}_{e : (v_e, v'_f) \in \mathcal{A}_{\mathcal{R}}} \Big\{\omega^{\mathsf{H}}_{\mathsf{C}}(v_e) + c'_{ef} + C_{e,f-1}\Big\} \qquad v'_f \in \mathcal{R}' \tag{9}$$

Figure 3 illustrates how the HDP functions. Similar to Figure 2, the unvisited customers and untraveled arcs are shown in grey. Instead of considering each possible drone sortie preceding v_g like in the EDP, the HDP decomposes this computation into two components. The first component, computed in equation (8), accounts for each feasible launch node v_e for serving v'_f with a drone, i.e., $(v_e, v'_f) \in \mathcal{A}_{\mathcal{R}}$ (as shown in Figure 3 (a) marked by dashed arrows). The cost $\omega^{\mathsf{H}}_{\mathsf{C}}(v'_f)$ is set as the minimum cost of launching the drone from a node v_e and the cost of serving all customers between v_e and v_{f-1} by the truck. Based on this computation, the launch node of v'_f is fixed in equation (9). The second component is computed in equation (7). Similar to the EDP, we consider that the truck may arrive at v_g while carrying the drone or recovers the drone at v_g. However, the computation of the latter case in the HDP is different. Each possible previous drone-eligible customer v'_f and its fixed launch node $v_{p_{\mathsf{C}}(v'_f)}$ (according to equation (9) and connected by densely dotted arrows in Figure 3 (b)) are considered. Specifically, we check if $\langle v_{p_{\mathsf{C}}(v'_f)}, v_f, v_g\rangle$ satisfies conditions (4b)–(4d).

To compute the minimum cost at customer v_g in the case there is a recovery of the drone, we add c'_{fg}

(the cost of the drone traveling from v'_f to v_g), and $c_{f-1,f+1} + C_{f+1,g}$ (the cost of the truck traversing from v_{f-1} to v_{f+1} bypassing customer v_f and visiting all the customers between v_{f+1} and v_g) to $\omega^{\mathsf{H}}_{\mathsf{C}}(v'_f)$ (see equation (7)). Since the launch node $v_{p_{\mathsf{C}}(v'_f)}$ is fixed in equation (9), the computational complexity of the HDP is $\mathcal{O}(\bar{n}^2)$.



(a) Considering the customer v'_f served by the drone

(b) Considering the customer v_g served by the truck (as the recovery node for the drone)

Figure 3: Illustration of the computation performed in the HDP

The reduced complexity of the HDP, when compared to the EDP, comes at the expense of compromising optimally. There are two main reasons for this. First, a sortie $\langle v_{p_{\mathsf{C}}(v'_f)}, v_f, v_g\rangle$ may not be feasible, thus would be discarded when checking the conditions (4b)–(4d) in equation (7). However, there may exist a feasible drone sortie $\langle v_l, v_f, v_g\rangle$ with $v_l \neq v_{p_{\mathsf{C}}(v'_f)}$, which was eliminated in the computation of $\omega^{\mathsf{H}}_{\mathsf{C}}(v'_f)$ in equation (8). The second reason for the HDP not finding an optimal solution is due to the fact that while $\langle v_{p_{\mathsf{C}}(v'_f)}, v_f, v_g\rangle$ may be feasible, there may exist an alternative drone sortie $\langle v_l, v_f, v_g\rangle$ with $v_l \neq v_{p_{\mathsf{C}}(v'_f)}$ with a lower cost. This comes from the fact that $\omega^{\mathsf{H}}_{\mathsf{C}}(v'_f)$ is computed without knowing where the drone will be recovered.

4.3.2 FRDDP-M

In this section, we formulate an HDP for the FRDDP-M in equations (10) and (11) that share similar ideas with the ones introduced in Section 4.3.1. We define $\omega^{\mathsf{H}}_{\mathsf{M}}(v_g)$ as the minimum makespan up to node v_g when visited by a truck and the drone is either on the truck when arriving at v_g or has just been recovered at v_g, and denote by $p_{\mathsf{M}}(v'_f)$ the index of the launch node leading to the minimum makespan at v'_f.

Different from the HDP for the FRDDP-C, the estimation of the minimum makespan at node v'_f is computed by considering each possible previous node v_e in $\mathcal{R}$ to find the one that minimizes the maximum between the arrival time of the drone at customer v'_f and the arrival time of the truck at node v_{f-1}. This computation is only performed to fix the launch node $v_{p_{\mathsf{M}}(v'_f)}$ of the drone when it serves customer v_f (in equation (11)). When computing $\omega^{\mathsf{H}}_{\mathsf{M}}(v_g)$ for the case where the drone is recovered at v_g after serving customer v'_f, we add to $\omega^{\mathsf{H}}_{\mathsf{M}}(v_{p_{\mathsf{M}}(v'_f)})$ the maximum between the travel time of the truck and the travel time of the drone between the launch node $v_{p_{\mathsf{M}}(v'_f)}$ and v_g, along with the launch time and the recovery time. The computational complexity of this HDP is $\mathcal{O}(\bar{n}^2)$. Its heuristic nature stems from the same reasons discussed in Section 4.3.1.

$$\omega^{\mathsf{H}}_{\mathsf{M}}(v_g) = \begin{cases} 0 & g = 0 \\ \tau_{01} & g = 1 \\ \min\Big\{\omega^{\mathsf{H}}_{\mathsf{M}}(v_{g-1}) + T_{g-1,g}, \min\limits_{v'_f:\langle v_{p_{\mathsf{M}}(v'_f)}, v_f, v_g\rangle \text{ subject to (4b)–(4d)}} \Big\{\omega^{\mathsf{H}}_{\mathsf{M}}(v_{p_{\mathsf{M}}(v'_f)}) + \\ \qquad \widehat{\tau} + \max\{\tau'_{p_{\mathsf{M}}(v'_f),f} + s'_f + \tau'_{fg}, T_{p_{\mathsf{M}}(v'_f),f-1} + s_{f-1} + \tau_{f-1,f+1} + T_{f+1,g}\} + \check{\tau}\Big\}\Big\} & g \in \{2, ..., \bar{n}\} \end{cases} \tag{10}$$

$$\text{where} \quad p_{\mathsf{M}}(v'_f) = \mathop{\arg\min}_{e:(v_e,v'_f)\in\mathcal{A}_{\mathcal{R}}} \Big\{\omega^{\mathsf{H}}_{\mathsf{M}}(v_e) + \widehat{\tau} + \max\Big\{\tau'_{ef}, T_{e,f-1}\Big\}\Big\} \qquad v'_f \in \mathcal{R}' \tag{11}$$

5 Solution Method

We propose a hybrid variable neighborhood search (HVNS) heuristic to solve the VRPD. The general scheme of the algorithm is described in Section 5.1. The basic structure of our HVNS is a VNS, which combines diversification by a perturbation procedure (usually known as shaking) and intensification by a local search mechanism (Mladenović and Hansen, 1997). Our shaking phase (Section 5.2) is performed through the use of destroy and repair operators. Our local search (Section 5.3) is a VND method with seven operators. Throughout the algorithm, we evaluate solutions via the HDP described in Section 4. To further accelerate the algorithm, we introduce a subHDP procedure that is used to filter non-promising moves (Section 5.4). Furthermore, to avoid being trapped in a local optimum, we use a simulated annealing acceptance criterion (in Section C of the supplementary material).

In the HVNS, we represent a VRPD solution using the concept of truck routes and complete routes introduced in Section 4. Specifically, we define a *truck solution* and a *complete solution* of the VRPD as $\mathcal{S} = \{\mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_m\}$ and $\overline{\mathcal{S}} = \{\overline{\mathcal{R}}_1, \overline{\mathcal{R}}_2, ..., \overline{\mathcal{R}}_m\}$ where $\mathcal{R}_l$ and $\overline{\mathcal{R}}_l$ are respectively the l-th truck route and complete route.

In what follows, we let $T_{\mathcal{R}}$ denote the duration of truck route $\mathcal{R}$, referring to the total time of the truck serving all the customers in $\mathcal{R}$, e.g., $T_{\mathcal{R}} = \sum_{(i,j)\in\mathcal{R}}(s_i + \tau_{ij})$. A *duration variation* is defined as the difference in duration between two truck routes $T_{\mathcal{R}_1}$ and $T_{\mathcal{R}_2}$, e.g., $T_{\mathcal{R}_1} - T_{\mathcal{R}_2}$. We let $T_{\overline{\mathcal{R}}}$ denote the duration of complete route $\overline{\mathcal{R}}$, that is the maximum time of arrival at the depot between that of the truck and that of its paired drone after serving all the customers in $\overline{\mathcal{R}}$. The maximum duration T of each route (defined in Section 3) translates into the feasibility condition $T_{\overline{\mathcal{R}}} \leqslant T$.

5.1 HVNS framework

Algorithm 1 outlines the general framework of the HVNS. The algorithm is initialized with a nearest neighborhood heuristic to generate a truck solution $\mathcal{S}$ (line 2). We build each truck route $\mathcal{R} \in \mathcal{S}$ by first selecting the nearest unvisited customer to the depot, and by then iteratively inserting the nearest unvisited customer as long as neither the duration nor the capacity of the truck is violated. In the VRPD-M, as long as there are unused trucks, we select the truck route containing more than one customer with the longest duration and split it into two. Precisely, let T_r the longest duration of the truck route be split. We maintain the customers up to position i in the truck route, and we create a new truck route with customers starting at position $i + 1$. This position i is such that the duration up to i is less than or equal to $T_r/2$ and the duration up to $i + 1$ is greater than $T_r/2$. After $\mathcal{S}$ is generated, we initialize λ, a parameter in the VND to determine whether we should consider a move or not (see details in Section 5.3). We solve the FRDDP using the HDP for each truck route in $\mathcal{S}$ to obtain the initial complete solution $\overline{\mathcal{S}}$ (line 3). In the first iteration of the algorithm (line 6–7), the HVNS calls the VND to improve the complete solution $\overline{\mathcal{S}}$ and obtains the value of λ^{avg} and λ^{max}, which determine the range of λ. In the remaining iterations, the HVNS first perturbs the incumbent complete solution $\overline{\mathcal{S}}$ with a `Shaking` procedure that generates a new truck solution $\mathcal{S}^{\mathsf{SH}}$ (line 9). Then, it obtains a new complete solution $\overline{\mathcal{S}}^{\mathsf{SH}}$ by running the `HeuristicDP` procedure to insert drone dispatches in $\mathcal{S}^{\mathsf{SH}}$ (line 10). From this solution, the algorithm gets a new local optimal truck solution $\mathcal{S}^{\mathsf{VND}}$ using the VND (line 11). The `ExactDP` procedure uses the EDP formulation from Section 4.2 to optimally insert drone dispatches and obtain a complete solution $\overline{\mathcal{S}}^{\mathsf{VND}}$ (line 12). To increase diversification, we use an acceptance criterion based on simulated annealing (SA) with a parameter κ representing the number of iterations since the last improvement, which is also used to update the parameter λ (line 13). The algorithm iterates until a maximum time limit $\mathsf{T}^{\mathsf{max}}$ is reached. In the entire algorithm, we denote the current computational time by T, which is initialized at the beginning of the HVNS.

Algorithm 1: The framework of the HVNS algorithm

```
   Data: Maximum time limit $\mathsf{T}^{\mathsf{max}}$
1  $\mathsf{T} \leftarrow 0, \mathsf{I} \leftarrow 0, \overline{\mathcal{S}}^* \leftarrow \varnothing$;
2  Generate an initial truck solution $\mathcal{S}$ with a nearest neighborhood heuristic, initialize $\lambda = \lambda_0 T_r$;
3  $\overline{\mathcal{S}} \leftarrow$ HeuristicDP ($\mathcal{S}$) ;
4  $\kappa \leftarrow 0$, if $\overline{\mathcal{S}}$ is feasible then $\overline{\mathcal{S}}^* \leftarrow \overline{\mathcal{S}}$;
5  while $\mathsf{T} \leqslant \mathsf{T}^{\mathsf{max}}$ do
6      if $\mathsf{I} = 0$ then
7          $(\mathcal{S}^{\mathsf{VND}}, \lambda^{\mathsf{avg}}, \lambda^{\mathsf{max}}) \leftarrow$ VariableNeighborhoodDescent$(\mathcal{S}, \overline{\mathcal{S}}, \lambda, \mathsf{I}, \mathsf{T})$;   ▷ see Section 5.3, Algorithm 2
8      else
9          $\mathcal{S}^{\mathsf{SH}} \leftarrow$ Shaking$(\overline{\mathcal{S}})$ ;                                   ▷ see Section 5.2
10         $\overline{\mathcal{S}}^{\mathsf{SH}} \leftarrow$ HeuristicDP ($\mathcal{S}^{\mathsf{SH}}$) ;
11         $(\mathcal{S}^{\mathsf{VND}}, \lambda^{\mathsf{avg}}, \lambda^{\mathsf{max}}) \leftarrow$ VariableNeighborhoodDescent$(\mathcal{S}^{\mathsf{SH}}, \overline{\mathcal{S}}^{\mathsf{SH}}, \lambda, \mathsf{I}, \mathsf{T})$ ;   ▷ see Section 5.3,
            Algorithm 2
12     $\overline{\mathcal{S}}^{\mathsf{VND}} \leftarrow$ ExactDP ($\mathcal{S}^{\mathsf{VND}}$) ;
13     $(\overline{\mathcal{S}}, \overline{\mathcal{S}}^*, \lambda, \kappa) \leftarrow$ SA AcceptanceCriterion$(\overline{\mathcal{S}}, \overline{\mathcal{S}}^*, \overline{\mathcal{S}}^{\mathsf{VND}}, \lambda, \lambda^{\mathsf{avg}}, \lambda^{\mathsf{max}}, \kappa)$ ;   ▷ see Algorithm 5 in Section
        C of the supplementary material
14     $\mathsf{I} \leftarrow \mathsf{I} + 1$;
   Result: $\overline{\mathcal{S}}^*$
```

5.2 Shaking

The shaking phase borrows the ideas of destroying and repairing solutions from the LNS literature. We define six destroy operators to remove customers from the given incumbent complete solution $\overline{\mathcal{S}}$ and three repair operators to insert those customers to form a new truck solution $\mathcal{S}^{\mathsf{SH}}$. We also use an operator to split one route into two.

Among the six destroy operators in our algorithm, the first four are widely used in the VRP literature (e.g., Demir et al. (2012); Macrina et al. (2019); Chen et al. (2021)), whereas the last two are specific for the VRPD and are adapted from Kitjacharoenchai et al. (2020). Each time the HVNS calls `Shaking` (Algorithm 1, line 9), a destroy operator is randomly chosen with an equal probability. The destroy phase takes a complete solution as input and returns a partial truck solution by removing customers from the incumbent complete solution and storing them in a removal list $\mathcal{L}$.

We define two types of destroy operators. The first type (1-4) sorts all customers in a non-increasing order based on a specific rule and subsequently removes β customers. If a removed customer is a launch node or a recovery node in a drone sortie, then the customer that is served by the drone in this sortie will also be removed. The second type (5-6) removes all customers that meet a specific rule. The six destroy operators are as follows.

1. **Random destroy** This operator randomly removes customers from $\mathcal{S}$ until β customers have been removed.

2. **Worst travel time destroy** This operator aims at removing customers that incur high truck travel times given their current position in a route. For this purpose, we sort all truck travel arcs in $\overline{\mathcal{S}}$ in descending order with respect to their travel time. We then iteratively delete customers associated with the arcs that have the highest travel times, until β customers have been removed.

3. **Cluster destroy** This operator randomly selects a seed customer, removes it first, and then removes the $\beta - 1$ customers that can be reached from the seed customer with the least truck travel time.

4. **Worst waiting-time destroy** Since the truck and the drone may wait for each other at a recovery node, this operator focuses on drone sorties that result in the longest waiting times. For this purpose, we sort all drone sorties in $\overline{\mathcal{S}}$ in descending order of their waiting time. We then iterate through these drone sorties and delete customers until β customers have been deleted. For a chosen drone sortie $\langle i, j, k\rangle$ where $i, k \in \mathcal{C}$, this operator removes not only the nodes i, j, and k, but also all customers served by the truck between i and k.

5. **One route destroy** This operator removes all the customers of the truck route with the fewest customers in solution $\overline{\mathcal{S}}$.

6. **Drone destroy** This operator removes all customers that are served by the drone in the complete solution $\overline{\mathcal{S}}$.

After the destroy phase removed a set $\mathcal{L}$ of customers, the repair phase inserts these customers into the partial truck solution $\mathcal{S}$ to form a new truck solution $\mathcal{S}^{\mathsf{SH}}$. We define three repair operators. The `Shaking` procedure randomly selects one of them with an equal probability. As long as $\mathcal{L}$ is not empty, the repair operator randomly removes one customer i from $\mathcal{L}$ and then inserts it to a specific position in the partial truck solution $\mathcal{S}$. The three repair operators exclusively consider truck-feasible capacity insertions and are as follows.

1. **Random repair** This operator inserts i at a random position in $\mathcal{S}$.

2. **Least travel time repair** This operator inserts i in a position with the least increase in travel time, i.e., $\tau_{ji} + \tau_{ik} - \tau_{jk}$ for every arc (j, k) in $\mathcal{S}$.

3. **Nearest polar angle repair** Considering the cartesian plane, this operator identifies the customer with the smallest difference in polar angle compared to the polar angle of the customer to be inserted. Let a_i be the polar angle of customer i. This operator inserts customer i at the position after customer $j = \arg\min_{l \in \mathcal{C}\setminus\mathcal{L}}\{|a_i - a_l|\}$ in $\mathcal{S}$.

In some cases, creating an additional route lowers the value of the objective function. The **Split** operator randomly selects one truck route, then splits it into two truck routes in a random position. If the incumbent best solution in the HVNS has not been improved during the last $\mathsf{N}^{\mathsf{max}}$ iterations, the `Shaking` procedure calls the **Split** operator after the repair operators have built a new truck solution.

5.3 Variable neighborhood descent

The VND relies on a list of operators $(\mathcal{K}_\vartheta)_{\vartheta=1,\ldots,\vartheta_{max}}$. Its general scheme is presented in Algorithm 2. At each iteration, a new solution is generated by calling the procedure `Search` (line 4) considering the ϑ-th operator. As soon as an improved neighboring solution (with respect to $\mathcal{S}^{\mathsf{inc}}$) is found, the procedure `Search` stops searching the current operator and restarts the search with $\mathcal{K}_1$ (line 6). Otherwise, the VND moves to the next operator $\mathcal{K}_{\vartheta+1}$ (line 9). A local optimum is reached when the last operator fails to improve the incumbent solution. During the VND, we may accept an improving infeasible solution (i.e., a complete solution having at least one complete route whose total duration exceeds the maximum duration T or containing more than m trucks). For this reason, we store every improving feasible (truck) solution returned by the procedure `Search` (line 7).

Algorithm 2: VariableNeighborhoodDescent($\mathcal{S}^{\mathsf{inc}}, \overline{\mathcal{S}}^{\mathsf{inc}}, \lambda, \mathsf{I}, \mathsf{T}$)

```
Data: An incumbent truck solution $\mathcal{S}^{\mathsf{inc}}$, its corresponding complete solution $\overline{\mathcal{S}}^{\mathsf{inc}}$, a duration variation
      threshold $\lambda$, the HVNS iteration number $\mathsf{I}$, the current computational time $\mathsf{T}$, the maximum time
      limit $\mathsf{T}^{\mathsf{max}}$, the list of operators $(\mathcal{K}_\vartheta)_{\vartheta=1,\ldots,\vartheta_{max}}$
1 $\vartheta \leftarrow 1, \mathcal{S} \leftarrow \mathcal{S}^{\mathsf{inc}}$ ;
2 if $\mathsf{I} = 0$ then $\lambda^{\mathsf{avg}} \leftarrow 0, \lambda^{\mathsf{max}} \leftarrow 0$;
3 while $\vartheta \leqslant |\mathcal{K}|$ or $\mathsf{T} \leqslant \mathsf{T}^{\mathsf{max}}$ do
4     $(\mathcal{S}^{\mathsf{new}}, \overline{\mathcal{S}}^{\mathsf{new}}, \lambda^{\mathsf{avg}}, \lambda^{\mathsf{max}}, improved) \leftarrow$ Search($\mathcal{S}^{\mathsf{inc}}, \overline{\mathcal{S}}^{\mathsf{inc}}, \lambda, \lambda^{\mathsf{avg}}, \lambda^{\mathsf{max}}, \mathsf{I}, \mathcal{K}_\vartheta$);    ▷ see Algorithm 3 and
       Algorithm 4
5     if improved then
6         $\mathcal{S}^{\mathsf{inc}} \leftarrow \mathcal{S}^{\mathsf{new}}, \overline{\mathcal{S}}^{\mathsf{inc}} \leftarrow \overline{\mathcal{S}}^{\mathsf{new}}, \vartheta \leftarrow 1$ ;
7         if $\overline{\mathcal{S}}^{\mathsf{new}}$ is feasible then $\mathcal{S} \leftarrow \mathcal{S}^{\mathsf{new}}$;
8     else
9         $\vartheta \leftarrow \vartheta + 1$;
Result: $\mathcal{S}, \lambda^{\mathsf{avg}}, \lambda^{\mathsf{max}}$
```

We use seven local search operators that are based either on intra-route moves or on inter-route moves. The moves are defined for solutions represented as truck solutions. Five of them are well-known for solving VRPs (e.g., (Vidal, 2022)), and include the inter-route and intra-route versions of 1-1 swap (exchanging two nodes) and 2-opt (exchanging two arcs in the route, which corresponds to reverse a section of the sequence), and the intra-route version of 1-0 relocate (removing a node and reinserting it in a different position). The remaining two operators are drone-related: 3-1 swap and 3-0 relocate. Rather than moving a single node, they move together three consecutive nodes i, j and k of a truck route with the condition that $\langle i, j, k\rangle$ is a drone sortie in the incumbent complete solution. The 3-1 swap operator exchanges a single node and a drone sortie that belongs to the same route. Similarly, the 3-0 relocate operator moves a drone sortie to another position in the same route.

We present our `Search` procedure in Algorithm 3. Specifically, we differentiate between intra-route moves (shown in Algorithm 3) and inter-route moves (shown in Algorithm 4 in Section B of the supplementary material). The general scheme is the same for both types of moves, except that for an inter-route move the total customer demand within each route should be satisfied first (line 4 in Algorithm 4). Given an incumbent truck solution $\mathcal{S}^{\mathsf{inc}}$, we define a *neighboring truck route* as a truck route resulting from a move an incumbent route $\mathcal{R}^{\mathsf{inc}} \in \mathcal{S}^{\mathsf{inc}}$. We refer to every inter-route move as a pair of neighboring truck routes $(\mathcal{R}_1^{\mathsf{new}}, \mathcal{R}_2^{\mathsf{new}})$ generated from a pair of truck routes $(\mathcal{R}_1^{\mathsf{inc}}, \mathcal{R}_2^{\mathsf{inc}})$ in $\mathcal{S}^{\mathsf{inc}}$. Similarly, we refer to every intra-route move as a single neighboring truck route $\mathcal{R}^{\mathsf{new}}$ generated from a truck route $\mathcal{R}^{\mathsf{inc}}$ in $\mathcal{S}^{\mathsf{inc}}$.

Given an intra-route move, we compute the duration $T_{\mathcal{R}^{\mathsf{new}}}$ of the neighboring truck route $\mathcal{R}^{\mathsf{new}}$ from $\mathcal{R}^{\mathsf{inc}}$. Since the drone could perform a subset of delivery tasks for the truck, and $\hat{\tau}$ as well as $\check{\tau}$ are usually negligible compared to the travel times, adding drone sorties generally reduces the route duration. For this reason, we allow $T_{\mathcal{R}^{\mathsf{new}}}$ to be greater than T, but not greater than μT, with $\mu \geqslant 1$ (line 4). The value of μ should be set considering the speed ratio between the drone and the truck. If the drone is faster than the truck, the route duration will be inevitably reduced. However, if the drone is slower than the truck, the route duration may increase or decrease, depending on the speed ratio between the drone and the truck. To further restrict the search space, we also limit the duration variation $T_{\mathcal{R}^{\mathsf{new}}} - T_{\mathcal{R}^{\mathsf{inc}}}$ between the truck routes $\mathcal{R}^{\mathsf{new}}$ and $\mathcal{R}^{\mathsf{inc}}$ by parameter λ (line 5), which is initialized by multiplying parameter λ_0 by the longest route duration of a truck route in the initial solution (see Algorithm 1, line 2).

We subsequently compute an approximation of the objective function value savings of a move (line 6) using the procedure described in Section 5.4. If such savings are promising, we use the HDP to solve the FRDDP to obtain the complete route $\overline{\mathcal{R}}^{\mathsf{new}}$ (line 7). We then update solutions $\mathcal{S}^{\mathsf{new}}$ and $\overline{\mathcal{S}}^{\mathsf{new}}$ if a better objective value is achieved (lines 8–9). Furthermore, when $\mathsf{I} = 0$ (line 10), we store the maximum duration variation λ^{max} and an average duration variation λ^{avg} by accounting for all duration variations associated with improving moves. We use ς to count the number of improving solutions computed in the VND. Each time an improving solution is found, `Search` stops and returns it.

Algorithm 3: Search($\mathcal{S}^{\mathsf{inc}}, \overline{\mathcal{S}}^{\mathsf{inc}}, \lambda, \lambda^{\mathsf{avg}}, \lambda^{\mathsf{max}}, \mathsf{I}, \mathcal{K}_\vartheta$)

Data: Incumbent truck solution $\mathcal{S}^{\mathsf{inc}}$, its complete solution $\overline{\mathcal{S}}^{\mathsf{inc}}$, duration threshold μ, duration variation threshold λ, the average duration variation threshold λ^{avg}, the maximum duration variation threshold λ^{max}, the current iteration number I, an operator $\mathcal{K}_\vartheta$

```
1  $improved \leftarrow \mathsf{false}, \varsigma \leftarrow 0, \mathcal{S}^{\mathsf{new}} \leftarrow \mathcal{S}^{\mathsf{inc}}, \overline{\mathcal{S}}^{\mathsf{new}} \leftarrow \overline{\mathcal{S}}^{\mathsf{inc}}$ ;
2  if $\mathcal{K}_\vartheta$ defines intra-route moves then
3      for each neighboring truck route $\mathcal{R}^{\mathsf{new}}$ generated from $\mathcal{R}^{\mathsf{inc}} \in \mathcal{S}^{\mathsf{inc}}$ do
4          if $T_{\mathcal{R}^{\mathsf{new}}} \leqslant \mu T$ then
5              if $T_{\mathcal{R}^{\mathsf{new}}} - T_{\mathcal{R}^{\mathsf{inc}}} \leqslant \lambda$ then
6                  if subHDP $(\mathcal{R}^{\mathsf{new}}, \mathcal{R}^{\mathsf{inc}}) > 0$ then
7                      $\overline{\mathcal{R}}^{\mathsf{new}} \leftarrow$ HeuristicDP $(\mathcal{R}^{\mathsf{new}})$;
8                      if $F(\overline{\mathcal{R}}^{\mathsf{new}}) < F(\overline{\mathcal{R}}^{\mathsf{inc}})$ then
9                          $\overline{\mathcal{S}}^{\mathsf{new}} \leftarrow \overline{\mathcal{R}}^{\mathsf{new}} \cup \overline{\mathcal{S}}^{\mathsf{inc}} \backslash \{\overline{\mathcal{R}}^{\mathsf{inc}}\}$; $\mathcal{S}^{\mathsf{new}} \leftarrow \mathcal{R}^{\mathsf{new}} \cup \mathcal{S}^{\mathsf{inc}} \backslash \{\mathcal{R}^{\mathsf{inc}}\}$;
10                         if $\mathsf{I} = 0$ then update $\lambda^{\mathsf{max}}$, $\lambda^{\mathsf{avg}} \leftarrow \big(\lambda^{\mathsf{avg}}\varsigma + (T_{\mathcal{R}^{\mathsf{new}}} - T_{\mathcal{R}^{\mathsf{inc}}})\big)/(\varsigma + 1)$, $\varsigma \leftarrow \varsigma + 1$;
11                         else $\lambda^{\mathsf{avg}} \leftarrow \lambda$;
12                         $improved \leftarrow \mathsf{true}$;   break;
13 else if $\mathcal{K}_\vartheta$ defines inter-route moves then
14     See Algorithm 4 in Section B of the supplementary material
```

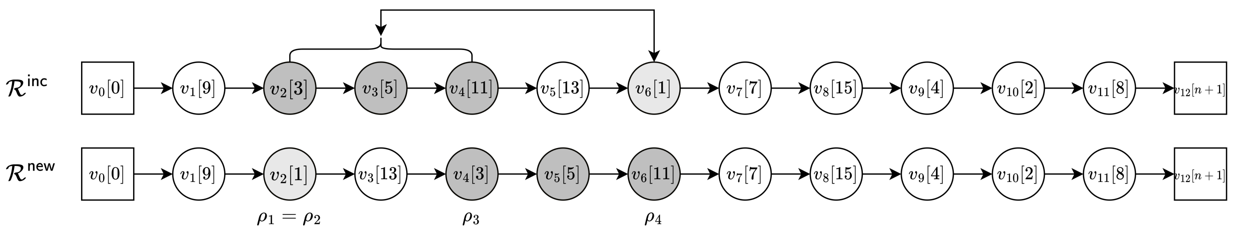
Result: $\mathcal{S}^{\mathsf{new}}, \overline{\mathcal{S}}^{\mathsf{new}}, \lambda^{\mathsf{avg}}, \lambda^{\mathsf{max}}, improved$

5.4 Filtering moves with a sub-heuristic dynamic program

Although our HDP is an order of magnitude faster than the EDP, calling the HDP for each neighboring truck route associated with an unfiltered move is computationally expensive. To address this, we develop the subHDP, which is a more time-efficient procedure to filter non-promising moves by approximating the potential saving in the objective function value of each move. The calculation of these savings is based on solving the FRDDP only for a partial route using the principles of the HDP. Specifically, the saving is estimated as the difference between the objective values of a partial route of the new route and those of the partial route of the original route. When the value of savings is positive, we further approximate the objective function value of new routes by executing the HDP to reevaluate if the move is improving (in Algorithm 3, line 7 and Algorithm 4, line 8). Note that as HDP and subHDP are heuristics, they may fail to detect improving moves.

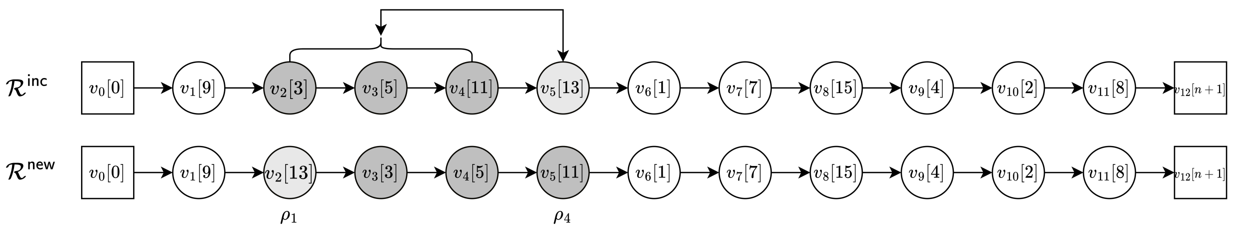
Given a neighboring truck route $\mathcal{R}^{\mathsf{new}}$ of $\mathcal{R}^{\mathsf{inc}} \in \mathcal{S}^{\mathsf{inc}}$, we extract a number of consecutive nodes from $\mathcal{R}^{\mathsf{new}}$, which we refer to as a *subroute*. The subHDP calculates an approximation of the objective function value savings induced by the subroute. The general scheme of the subHDP is as follows: for an inter-route (resp. intra-route) move: 1) Create one subroute (resp. one or two subroutes) for each of the two neighboring truck routes (resp. for the single neighboring truck route); 2) Calculate the objective function value for each created subroute; 3) Calculate the approximate savings induced by the move.

As the extracted number of consecutive nodes from $\mathcal{R}^{\mathsf{new}}$ may lead to overlaps, we distinguish between two cases. The case where a move results in inconsecutive node changes in a route, and the case where a move results in consecutive node changes in a route. In the first case, one or two subroutes are created, whereas in the second case, a single subroute is created. The latter is always the case for inter-route operators. Figure 4 shows two different examples of an intra-route 3-1 swap move: three consecutive nodes that form a drone sortie in the incumbent complete route $\overline{\mathcal{R}}^{\mathsf{inc}}$ (marked with dark gray) and a node (marked with light gray), where the index of each customer is in the brackets. In Figure 4 (a), the move results in node changes with two components because the node 1 and nodes $3, 5, 11$ are not consecutive in the new truck route $\mathcal{R}^{\mathsf{new}}$. But in Figure 4 (b), despite that the move changes two groups of nodes, the node 13 and nodes $3, 5, 11$ are consecutive, merging the two components into one.



$\mathcal{R}^{\mathsf{inc}}$
$v_0[0]$ $v_1[9]$ $v_2[3]$ $v_3[5]$ $v_4[11]$ $v_5[13]$ $v_6[1]$ $v_7[7]$ $v_8[15]$ $v_9[4]$ $v_{10}[2]$ $v_{11}[8]$ $v_{12}[n+1]$
$\mathcal{R}^{\mathsf{new}}$
$v_0[0]$ $v_1[9]$ $v_2[1]$ $v_3[13]$ $v_4[3]$ $v_5[5]$ $v_6[11]$ $v_7[7]$ $v_8[15]$ $v_9[4]$ $v_{10}[2]$ $v_{11}[8]$ $v_{12}[n+1]$
$\rho_1 = \rho_2$
ρ_3
ρ_4

(a) A move resulting in inconsecutive node changes



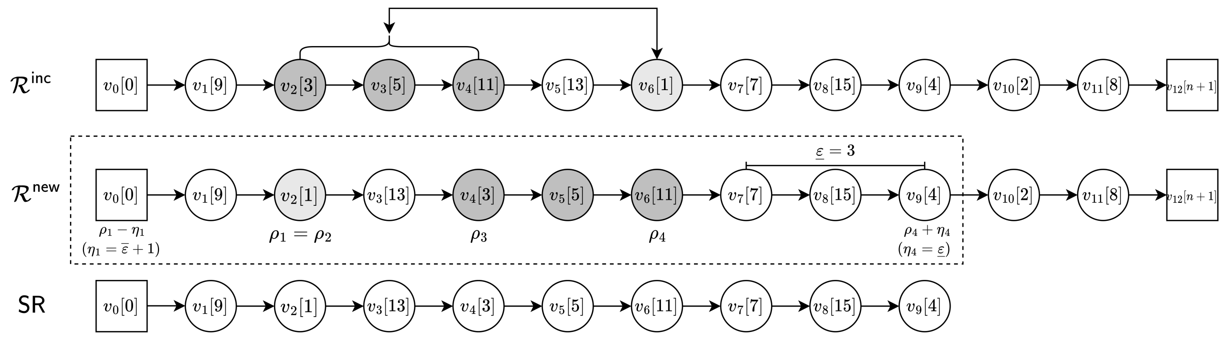
$\mathcal{R}^{\mathsf{inc}}$
$v_0[0]$ $v_1[9]$ $v_2[3]$ $v_3[5]$ $v_4[11]$ $v_5[13]$ $v_6[1]$ $v_7[7]$ $v_8[15]$ $v_9[4]$ $v_{10}[2]$ $v_{11}[8]$ $v_{12}[n+1]$
$\mathcal{R}^{\mathsf{new}}$
$v_0[0]$ $v_1[9]$ $v_2[13]$ $v_3[3]$ $v_4[5]$ $v_5[11]$ $v_6[1]$ $v_7[7]$ $v_8[15]$ $v_9[4]$ $v_{10}[2]$ $v_{11}[8]$ $v_{12}[n+1]$
ρ_1
ρ_4

(b) A move resulting in consecutive node changes

Figure 4: Move component illustration

To create subroutes from a neighboring truck route $\mathcal{R}^{\mathsf{new}}$, we define ρ_1 (resp. ρ_3) and ρ_2 (resp. ρ_4) as the starting position and the ending position of the first component (resp. second component) in $\mathcal{R}^{\mathsf{new}}$ when the move results in inconsecutive node changes. If there is only one node included in a component, then $\rho_1 = \rho_2, \rho_3 = \rho_4$. For example, in Figure 4 (a), $\rho_1 = \rho_2 = 2, \rho_3 = 4, \rho_4 = 6$. If the move results in consecutive node changes, we discard ρ_2 and ρ_3. In Figure 4 (b), $\rho_1 = 2, \rho_4 = 5$. In addition, we define $\underline{\varepsilon}$ as the number of nodes to be considered after ρ_2 (or ρ_4) in the subroute of $\mathcal{R}^{\mathsf{new}}$.

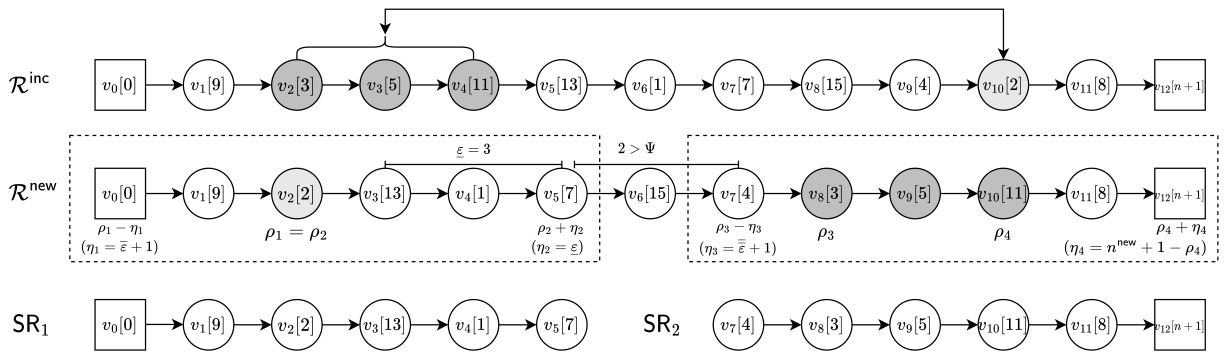
We first present the case where we generate a single subroute SR from a neighboring truck route $\mathcal{R}^{\mathsf{new}}$. Let n^{new} be the number of customers in $\mathcal{R}^{\mathsf{new}}$. We define the subroute $\mathsf{SR} = (v_{\rho_1-\eta_1}, ..., v_{\rho_1}, ..., v_{\rho_4}, ..., v_{\rho_4+\eta_4})$ from $\mathcal{R}^{\mathsf{new}}$, where $\eta_1 = \overline{\varepsilon} + 1$ with $\overline{\varepsilon}$ the number of consecutive customers that are not drone-eligible before v_{ρ_1}, and $\eta_4 = \min\{\underline{\varepsilon}, n^{\mathsf{new}} + 1 - \rho_4\}$. For the particular move in Figure 5, assume that $\overline{\varepsilon} = 1$, and $\underline{\varepsilon} = 3$, we have $\mathsf{SR} = (0, 9, 1, 13, 3, 5, 11, 7, 15, 4)$.



$\mathcal{R}^{\mathsf{inc}}$
$v_0[0]$ $v_1[9]$ $v_2[3]$ $v_3[5]$ $v_4[11]$ $v_5[13]$ $v_6[1]$ $v_7[7]$ $v_8[15]$ $v_9[4]$ $v_{10}[2]$ $v_{11}[8]$ $v_{12}[n+1]$
$\underline{\varepsilon} = 3$
$\mathcal{R}^{\mathsf{new}}$
$v_0[0]$ $v_1[9]$ $v_2[1]$ $v_3[13]$ $v_4[3]$ $v_5[5]$ $v_6[11]$ $v_7[7]$ $v_8[15]$ $v_9[4]$ $v_{10}[2]$ $v_{11}[8]$ $v_{12}[n+1]$
$\rho_1 - \eta_1$
$(\eta_1 = \overline{\varepsilon} + 1)$
$\rho_1 = \rho_2$
ρ_3
ρ_4
$\rho_4 + \eta_4$
$(\eta_4 = \underline{\varepsilon})$
SR
$v_0[0]$ $v_1[9]$ $v_2[1]$ $v_3[13]$ $v_4[3]$ $v_5[5]$ $v_6[11]$ $v_7[7]$ $v_8[15]$ $v_9[4]$

Figure 5: An example of creating a subroute

If there are more than Ψ customers between customer $v_{\rho_2+\eta_2}$ and customer $v_{\rho_3-\eta_3}$, i.e., $\rho_2 + \eta_2 + \Psi < \rho_3 - \eta_3$, we generate two subroutes $\mathsf{SR}_1 = (v_{\rho_1-\eta_1}, ..., v_{\rho_1}, ..., v_{\rho_2}, ..., v_{\rho_2+\eta_2})$ and $\mathsf{SR}_2 = (v_{\rho_3-\eta_3}, ..., v_{\rho_3}, ..., v_{\rho_4}, ..., v_{\rho_4+\eta_4})$ with $\eta_2 = \underline{\varepsilon}$, and $\eta_3 = \overline{\overline{\varepsilon}} + 1$ (where $\overline{\overline{\varepsilon}}$ is the number of consecutive customers that are not drone-eligible before v_{ρ_3}) for $\mathcal{R}^{\mathsf{new}}$. This condition ensures that there is a sufficient gap between the end of SR_1 and the beginning of SR_2. Figure 6 shows an example of creating two subroutes when $\Psi = 1$, $\overline{\overline{\varepsilon}} = 0$ (the values of other parameters are the same as in Figure 5.)



$\mathcal{R}^{\mathsf{inc}}$
$v_0[0]$ $v_1[9]$ $v_2[3]$ $v_3[5]$ $v_4[11]$ $v_5[13]$ $v_6[1]$ $v_7[7]$ $v_8[15]$ $v_9[4]$ $v_{10}[2]$ $v_{11}[8]$ $v_{12}[n+1]$
$\underline{\varepsilon} = 3$
$2 > \Psi$
$\mathcal{R}^{\mathsf{new}}$
$v_0[0]$ $v_1[9]$ $v_2[2]$ $v_3[13]$ $v_4[1]$ $v_5[7]$ $v_6[15]$ $v_7[4]$ $v_8[3]$ $v_9[5]$ $v_{10}[11]$ $v_{11}[8]$ $v_{12}[n+1]$
$\rho_1 - \eta_1$
$(\eta_1 = \overline{\varepsilon} + 1)$
$\rho_1 = \rho_2$
$\rho_2 + \eta_2$
$(\eta_2 = \underline{\varepsilon})$
$\rho_3 - \eta_3$
$(\eta_3 = \overline{\overline{\varepsilon}} + 1)$
ρ_3
ρ_4
$\rho_4 + \eta_4$
$(\eta_4 = n^{\mathsf{new}} + 1 - \rho_4)$
SR_1
$v_0[0]$ $v_1[9]$ $v_2[2]$ $v_3[13]$ $v_4[1]$ $v_5[7]$
SR_2
$v_7[4]$ $v_8[3]$ $v_9[5]$ $v_{10}[11]$ $v_{11}[8]$ $v_{12}[n+1]$

Figure 6: An example of creating two subroutes

To calculate the objective function value of a subroute $(v_{\rho-\eta}, ..., v_\rho, ..., v_{\rho'}, ..., v_{\rho'+\eta'})$ in $\mathcal{R}^{\mathsf{new}}$ with $(\rho, \eta, \rho', \eta') \in \{(\rho_1, \eta_1, \rho_4, \eta_4), (\rho_1, \eta_1, \rho_2, \eta_2), (\rho_3, \eta_3, \rho_4, \eta_4)\}$, we adapt the computation performed for the HDP, which forms the basis of the subHDP, as shown in equations (12)–(16). We denote by $\mathcal{R}^{\mathsf{new}'}$ the set of drone-eligible customers in $\mathcal{R}^{\mathsf{new}}$, and denote by $\mathcal{A}_{\mathcal{R}^{\mathsf{new}}}$ the arc set in $\mathcal{G}_{\mathcal{R}^{\mathsf{new}}}$. In (12)–(14) for the FRDDP-C, $\widetilde{\omega}_{\mathsf{C}}(v_g)$ is the approximate minimum cost up to the node v_g, $\widetilde{\omega}_{\mathsf{C}}(v'_f)$ is the approximate minimum cost up to the drone-eligible customer v'_f taking into account that the truck is traveling to v_{f-1}, and $\widetilde{p}_{\mathsf{C}}(v'_f)$ the index of the node that leads to the minimum cost when computing $\widetilde{\omega}_{\mathsf{C}}(v'_f)$. In equations (15) and (16) for the FRDDP-M, $\widetilde{\omega}_{\mathsf{M}}(v_g)$ is the approximate minimum makespan up to the node v_g, and $\widetilde{p}_{\mathsf{M}}(v'_f)$ is the index of the launch node that leads to the approximate minimum makespan at the drone-eligible customer v'_f. It is important to note that the values $\omega^{\mathsf{H}}_{\mathsf{C}}(v_g)$, $\omega^{\mathsf{H}}_{\mathsf{C}}(v'_f)$, $p_{\mathsf{C}}(v'_f)$, $\omega^{\mathsf{H}}_{\mathsf{M}}(v_g)$, and $p_{\mathsf{M}}(v'_f)$ come from the incumbent truck route $\mathcal{R}^{\mathsf{inc}}$. The computations for a subroute of $\mathcal{R}^{\mathsf{new}}$ are the following.

$$\widetilde{\omega}_{\mathsf{C}}(v_g) = \begin{cases} \omega^{\mathsf{H}}_{\mathsf{C}}(v_g) & g = \rho - \eta \\ \min\Big\{\widetilde{\omega}_{\mathsf{C}}(v_{g-1}) + c_{g-1,g}, \\ \qquad \min\limits_{v'_f : f \geqslant \rho-\eta \wedge \langle v_{\widetilde{p}_{\mathsf{C}}(v'_f)}, v_f, v_g\rangle \text{ subject to } (4b)-(4d)} \Big\{\widetilde{\omega}_{\mathsf{C}}(v'_f) \\ \qquad + c'_{fg} + c_{f-1,f+1} + C_{f+1,g}\Big\}\Big\} & g \in \{\rho-\eta+1, \ldots, \rho'+\eta'\} \end{cases} \tag{12}$$

$$\widetilde{\omega}_{\mathsf{C}}(v'_f) = \begin{cases} \omega^{\mathsf{H}}_{\mathsf{C}}(v'_f) & f = \rho - \eta \\ \min\limits_{v_e : e \geqslant \rho-\eta \wedge (v_e, v'_f) \in \mathcal{A}_{\mathcal{R}^{\mathsf{new}}}} \Big\{\widetilde{\omega}_{\mathsf{C}}(v_e) + c'_{ef} + C_{e,f-1}\Big\} & v'_f \in \mathcal{R}^{\mathsf{new}'} \wedge f \in \{\rho-\eta+1, \ldots, \rho'+\eta'\} \end{cases} \tag{13}$$

where
$$\widetilde{p}_{\mathsf{C}}(v'_f) = \begin{cases} p_{\mathsf{C}}(v'_f) & f = \rho - \eta \\ \arg\min\limits_{e : e \geqslant \rho-\eta \wedge (v_e, v'_f) \in \mathcal{A}_{\mathcal{R}^{\mathsf{new}}}} \Big\{\widetilde{\omega}_{\mathsf{C}}(v_e) + c'_{ef} + C_{e,f-1}\Big\} & v'_f \in \mathcal{R}^{\mathsf{new}'} \wedge f \in \{\rho-\eta+1, \ldots, \rho'+\eta'\} \end{cases} \tag{14}$$

$$\widetilde{\omega}_{\mathsf{M}}(v_g) = \begin{cases} \omega^{\mathsf{H}}_{\mathsf{M}}(v_g) & g = \rho - \eta \\ \min\Big\{\widetilde{\omega}_{\mathsf{M}}(v_{g-1}) + T_{g-1,g}, \\ \qquad \min\limits_{v'_f : f \geqslant \rho-\eta \wedge \langle v_{\widetilde{p}_{\mathsf{M}}(v'_f)}, v_f, v_g\rangle \text{ subject to } (4b)-(4d)} \Big\{\widetilde{\omega}_{\mathsf{M}}(v_{\widetilde{p}_{\mathsf{M}}(v'_f)}) + \widehat{\tau} + \max\{\tau'_{\widetilde{p}_{\mathsf{M}}(v'_f),f} \\ \qquad + s'_f + \tau'_{fg}, T_{\widetilde{p}_{\mathsf{M}}(v'_f),f-1} + s_{f-1} + \tau_{f-1,f+1} + T_{f+1,g}\} + \check{\tau}\Big\}\Big\} & g \in \{\rho-\eta+1, \ldots, \rho'+\eta'\} \end{cases} \tag{15}$$

where

$$\widetilde{p}_{\mathsf{M}}(v'_f) = \begin{cases} p_{\mathsf{M}}(v'_f) & f = \rho - \eta \\ \arg\min\limits_{e : e \geqslant \rho-\eta \wedge (v_e, v'_f) \in \mathcal{A}_{\mathcal{R}^{\mathsf{new}}}} \Big\{\widetilde{\omega}_{\mathsf{M}}(v_e) + \widehat{\tau} + \max\Big\{\tau'_{ef}, T_{e,f-1}\Big\}\Big\} & v'_f \in \mathcal{R}^{\mathsf{new}'} \wedge f \in \{\rho-\eta+1, \ldots, \rho'+\eta'\} \end{cases} \tag{16}$$

Finally, we compute the approximate saving induced by the neighboring truck route $\mathcal{R}^{\mathsf{new}}$ generated from $\mathcal{R}^{\mathsf{inc}}$. This saving, denoted by `subHDP`$(\mathcal{R}^{\mathsf{new}}, \mathcal{R}^{\mathsf{inc}})$ (Algorithm 3, line 6, and Algorithm 4, line 7), is calculated in equation (17) when we have one subroute, and in equation (18) when we have two subroutes.

$$\texttt{subHDP}(\mathcal{R}^{\mathsf{new}}, \mathcal{R}^{\mathsf{inc}}) = (\omega^{\mathsf{H}}_{\mathsf{C}}(v_{\rho_4+\eta_4}) - \omega^{\mathsf{H}}_{\mathsf{C}}(v_{\rho_1-\eta_1})) - (\widetilde{\omega}_{\mathsf{C}}(v_{\rho_4+\eta_4}) - \widetilde{\omega}_{\mathsf{C}}(v_{\rho_1-\eta_1})) \tag{17}$$

$$\begin{aligned} \texttt{subHDP}(\mathcal{R}^{\mathsf{new}}, \mathcal{R}^{\mathsf{inc}}) = {} & (\omega^{\mathsf{H}}_{\mathsf{C}}(v_{\rho_2+\eta_2}) - \omega^{\mathsf{H}}_{\mathsf{C}}(v_{\rho_1-\eta_1})) - (\widetilde{\omega}_{\mathsf{C}}(v_{\rho_2+\eta_2}) - \widetilde{\omega}_{\mathsf{C}}(v_{\rho_1-\eta_1})) \\ & + (\omega^{\mathsf{H}}_{\mathsf{C}}(v_{\rho_4+\eta_4}) - \omega^{\mathsf{H}}_{\mathsf{C}}(v_{\rho_3-\eta_3})) - (\widetilde{\omega}_{\mathsf{C}}(v_{\rho_4+\eta_4}) - \widetilde{\omega}_{\mathsf{C}}(v_{\rho_3-\eta_3})) \end{aligned} \tag{18}$$

6 Computational results

We present a series of computational experiments to test the performance of the HVNS algorithm introduced in Section 5. We first introduce all the data sets we use in Section 6.1. We then establish the values of certain key parameters, and evaluate different algorithmic components of the HVNS to assess their added value in Section 6.2. In Section 6.3, we compare the results obtained by our HVNS against nine benchmark sets from the literature. We coded our HVNS using Java version 18.0.2 and we conducted all experiments on a Linux machine equipped with an Intel Xeon(R) Gold 6226R CPU clocked at 2.90 GHz.

6.1 Data sets and benchmarks

In this section, we introduce all the data sets [1] that we use in our computational experiments. We also present the benchmarks from the literature against which we compare the results of our HVNS.

There are two data sets we use in Section 6.2 to evaluate different configurations of the HVNS, each corresponding to a different objective. For the VRPD-C, we use the data set introduced by Sacramento et al. (2019). In this data set, there are four sets of 12 instances, each with 6, 10, 12, and 20 customers, and four sets of 16 instances, each with 50, 100, 150, and 200 customers. The problem parameters are as follows: the launch time $\widehat{\tau}$ and the recovery time $\breve{\tau}$ are both 1 minute. The customer service time of the truck and the drone are 2 minutes and 1 minute. The speed of the trucks and the drones are 35 mph and 50 mph. The drone flight endurance is 30 minutes. The maximum duration of each route is 480 minutes. Additionally, the fuel price and consumption of the truck are set to 1.13 €$/l$ and 0.07 l/km. The transportation cost of the drone for every arc $(i, j) \in \mathcal{A}$ is set to $c'_{ij} = 0.1c_{ij}$ in Sacramento et al. (2019), considering the fact that the U.S. firm Workhorse estimated an approximate value of 2 cents per mile for the electricity costs (Kharpal, 2016), which corresponds to between 10–15% of the total truck's cost. For the VRPD-M, we use the data set introduced by Bouman et al. (2018b), which contains instances with 4 to 499 customers. In these instances, the drone is twice as fast as the truck on all arcs, i.e., $\tau_{ij} = 2\tau'_{ij}$. Similarly to El-Adle et al. (2021) who used this data set, we set the drone flight endurance to 30 minutes. The rest of the parameters are set as 0. We compare the performance of our HVNS against nine benchmark sets from the literature in Section 6.3 (see Table 2 for an overview). Note that our HVNS is capable of handling eight possible problem variants based on the aforementioned three axes. However, we found benchmarks related to only four variants (as indicated in Table 3), and thus we performed experiments only on those variants.

Name	Problem variant	Reference	Method	Time limit	Hardware	Data set
SPR	VRPD-C, $m \geqslant 1$, NoLW	Sacramento et al. (2019)	ALNS	5 minutes (10 runs)	Intel Xeon 2660v3 @ 2.60 GHz	Sacramento et al. (2019)
RFK	VRPD-C, $m \geqslant 1$, NoLW	Rave et al. (2023)	ALNS	5 minutes (10 runs)	AMD Ryzen 9 3950X with 32 GB RAM	Sacramento et al. (2019)
EGH	VRPD-M, $m = 1$, LW	El-Adle et al. (2021)	MIP	3.1 hours	Intel i7-7700K processor and 32 GB of RAM	Bouman et al. (2018b)
TB-1 TB-2	VRPD-M, $m = 1$, LW VRPD-M, $m = 2$, LW	Tamke and Buscher (2021)	B&C	12 hours	Intel Xeon(R) E5-4627 v2 @ 3.3 GHz with 8 cores and 768GB RAM	Reinelt (1997)
RR-RANGE30 RR-MHD	VRPD-M, $m = 1$, LW VRPD-M, $m = 1$, NoLW	Roberti and Ruthmair (2021)	B&P	1 hour	Intel Xeon E5-2670v2 @ 2.5 GHz, a memory limit of 8-GB RAM	Poikonen et al. (2019)
KVMLTB-1 KVMLTB-M	VRPD-M, $m = 1$, LW VRPD-M, $m > 1$, LW	Kitjacharoenchai et al. (2019)	Adapted FSTSP heuristic	No (20 runs)	Intel Core i5 @ 2.7 GHz with 8 GB RAM	Reinelt (1997)

Table 2: Summary of data sets and benchmarks

For the VRPD-C, we use two benchmark sets from Sacramento et al. (2019) and Rave et al. (2023), which we refer to as SPR and RFK. Both works consider multiple trucks and assume NoLW. Both papers presented an ALNS algorithm and performed computational experiments using the data set introduced by Sacramento et al. (2019). Each instance was run 10 times for 5 minutes.

[1] available from https://www.math.u-bordeaux.fr/~afroger001/documents/vrpd_instance_sets.zip

	Cost (C)		Makespan (M)	
	LW	NoLW	LW	NoLW
$m = 1$	-	-	✓	✓
$m > 1$	-	✓	✓	-

Table 3: Considered VRPD problem variants

For the VRPD-M, we compare our results with a total of seven benchmark sets, including five benchmarks solved by exact algorithms and two benchmarks solved by heuristics. The first benchmark we compare with is from El-Adle et al. (2021), which we refer to as `EGH`. This work considers a problem with a single truck and a single drone and assumes LW. The problem is formulated with a MIP model enhanced by a series of bound improvement strategies and solved by Gurobi with a time limit of 11,000 seconds (approximately 3.1 hours). The authors generate five groups of 30 instances based on the Bouman et al. (2018b) instances. Specifically, two groups of instances are built by selecting the first 15 and 19 customers of the 19-customer instances, and three groups are built by selecting the first 23, 27, and 31 customers of the 49-customer instances. The next two benchmarks come from Tamke and Buscher (2021), who considered 1 or 2 trucks ($m = 1, 2$), 1 or 2 drones attached to each truck, and the LW assumption. These authors solved the VRPD-M with a B&C algorithm imposing a time limit of 43,200 seconds (12 hours). For their computational experiments, they took three TSPLIB (Reinelt, 1997) instances: kroA100, kroA200 and kroA300 and selected the first 14, 19, 24, 29 customers to vary the instance size. Since our HVNS cannot handle multiple drones attached to one truck, we only compare the results they obtained with a single drone attached to each truck. We refer to their benchmarks with one truck and two trucks as `TB-1` and `TB-2`. Another two benchmarks, denoted by `RR-RANGE30` and `RR-MHD`, are two variants of the TSPD tackled by Roberti and Ruthmair (2021), assuming LW and NoLW. The authors proposed a B&P algorithm and imposed a time limit of 3,600 seconds (1 hour) to solve each instance. They used the data set introduced by Poikonen et al. (2019), with 9, 19, 29, and 39 customers (with 75 instances for each customer size). The next two benchmarks are from Kitjacharoenchai et al. (2019), who chose 22 instances with 14–98 customers from Reinelt (1997). We refer to their benchmarks with one truck and multiple trucks as `KVMLTB-1` and `KVMLTB-M`. Kitjacharoenchai et al. (2019) studied the multiple TSPD, where the drone may be recovered by different trucks. The authors conducted a series of experiments to compare their heuristic with an adaptation of a heuristic from the literature that assumes the truck and drone are paired. We compare our algorithm to this adapted heuristic. Since the flight endurance of drones is neglected in `KVMLTB-1` and `KVMLTB-M`, we categorize these two benchmarks into the LW assumption and also ignore the drone's flight endurance when comparing the results. We note that, when solving the VRPD with the LW assumption, condition (4d) is disregarded.

Given that the results we benchmark the performance of our algorithm against obtained using various machines, in Section D of the supplementary material, we conducted a comparison between the hardware reported in benchmarks `SPR`, `RFK`, `KVMLTB-1`, and `KVMLTB-M` and our hardware using the single thread rating reported on the website: `https://www.cpubenchmark.net/`. According to the available information, there is a variation in CPU speed ranging from -20% to +20% that must be borne in mind when looking at the results.

In the following sections, we present average results based on the number of customers (or trucks).

The detailed results for every single tested instance are available online[2].

6.2 HVNS configuration

We conduct a series of experiments to determine the configuration of our HVNS. We first determine suitable values of the key parameters that significantly affect the HVNS in Section 6.2.1. We then evaluate the added value of the 17 operators used in the shaking phase and the VND in Section 6.2.2. Finally, in Section 6.2.3 we investigate the impact of using the HDP and the subHDP.

In sections 6.2.1–6.2.3, we chose a subset of 20 instances: 10 instances with 20, 50, 100, 150, and 200 customers from Sacramento et al. (2019) for the VRPD-C ($m \geqslant 1$ and NoLW assumption), and 10 instances with 19, 49, 74, 99, and 174 customers from Bouman et al. (2018b) for the VRPD-M ($m = 2$ and LW assumption). For each instance, we run the algorithm five times and compare the average objective values of these five runs within a time limit of $\mathsf{T}^{\mathsf{max}} = 5$ min. We sample the value of β used in several destroy operators from a uniform distribution between $0.2n$ and $0.3n$. Furthermore, we set the temperature factor $t^{\text{init}} = 0.004$, and the number of non-improving iterations $\mathsf{N}^{\mathsf{max}} = 10$ after which the threshold λ is reset. In Tables 4–8, the configuration that gives the best overall average values is displayed in bold and corresponds to the one chosen in the final configuration of our HVNS.

6.2.1 Parameter settings

In this section, we investigate four parameters, specifically λ_0, μ, $\underline{\varepsilon}$ and Ψ, that highly affect the HVNS algorithm. The first two parameters are used to discard unpromising moves and operate at the truck route level. The last two parameters are used in the subHDP to approximate the objective function saving induced by a move.

In a first set of experiments, we compare the results obtained by our HVNS with different values for λ_0 and μ. The parameter λ_0 is used in Algorithm 1 to initialize the value of λ which is subsequently used in Algorithms 3 and 4 when comparing the duration variation induced by a move. Specifically, if the duration variation between a neighboring truck route and the original truck route exceeds λ, then the move is discarded. This threshold is updated throughout the search. We experiment with $\lambda_0 = 0.0$ and $\lambda_0 = 0.2$. The parameter μ is used in Algorithms 3 and 4 when examining the duration of a truck route modified by a move. Specifically, if the duration of a neighboring truck route is larger than μT, then the move is discarded. We experiment with $\mu = 1.3$ and $\mu = +\infty$, where the latter discards limitations on a neighboring truck route's duration with respect to T. Note that since there is no maximum duration in the data set of Bouman et al. (2018a) for the VRPD-M, the experiments of μ are only conducted for the VRPD-C.

For the first set of experiments, we set $\underline{\varepsilon} = 5$ and $\Psi = 5$. Let $\text{obj}_{u,\delta,r}$ be the objective function value computed by our algorithm for instance u, under configuration δ, and a run number r (with $r = 1, \ldots, 5$). Let $\overline{\text{obj}}_{u,\delta}$ be the average objective function values of five runs (i.e., $\overline{\text{obj}}_{u,\delta} = \frac{\sum_{r=1}^{5} \text{obj}_{u,\delta,r}}{5}$). For each instance u, we then compute the best average objective function value $\overline{\text{obj}}_u^{\mathsf{BEST}} = \min_{\delta}\{\overline{\text{obj}}_{u,\delta}\}$ over all configurations. Furthermore, we compute the gap as $\text{gap}_{u,\delta,r} = 100(\text{obj}_{u,\delta,r} - \overline{\text{obj}}_u^{\mathsf{BEST}})/\overline{\text{obj}}_u^{\mathsf{BEST}}$. Lastly, for each configuration δ, we compute the average and standard deviation of these gaps over all instances and all runs. Table 4 reports these values for the VRPD-C, the VRPD-M, and for both problems (under columns VRPD). Based on Table 4, we observe that $\lambda_0 = 0.2$ outperforms the configuration $\lambda_0 = 0.0$ as it

[2]https://www.math.u-bordeaux.fr/~afroger001/documents/vrpd_detailed_results.zip

gives the minimum average gaps and the minimum standard deviations of gaps for both problems. These results indicate that allowing a new neighboring truck route with a little more duration leads to better results. As for μ, we observe that setting $\mu = +\infty$ leads to significantly worse results, when compared with $\mu = 1.3$. Thus, we set $\lambda_0 = 0.2$ and $\mu = 1.3$ in the HVNS.

λ_0	μ	VRPD-C		VRPD-M		VRPD	
		Gap		Gap		Gap	
		Avg	Std Dev	Avg	Std Dev	Avg	Std Dev
0.0	1.3	1.04	2.38	0.65	1.62	0.84	2.04
	$+\infty$	8.85	9.14			4.75	7.74
0.2	**1.3**	**0.24**	**0.83**	**0.13**	**1.27**	**0.18**	**1.07**
	$+\infty$	13.01	18.04			6.57	14.32

Table 4: Evaluation of parameters λ_0 and μ

In a second set of experiments, we compare the results obtained by our HVNS with different values for $\underline{\varepsilon}$ and Ψ, which are associated with the subroute generation in the subHDP. Parameter $\underline{\varepsilon}$ is the number of nodes to be included after the ending position of a move to generate a subroute, while parameter Ψ is the minimum number of customers to be considered between the end of the first subroute and the beginning of the second subroute (when two subroutes are generated). We tune $\underline{\varepsilon}$ and Ψ with different values as follows: $\underline{\varepsilon} = \{1, 5, 10\}$ and $\Psi = \{1, 5, 10\}$. Table 5 reports the averages and standard deviations of the gaps between the results of each parameter combination and the best average results among all parameter combinations (all calculations are performed as in Table 4). It is evident that the combination of $\underline{\varepsilon} = 5$ and $\Psi = 5$ yields high-quality solutions as this combination obtains the minimum overall averages and relatively small standard deviations of the gaps. Therefore, we set $\underline{\varepsilon} = 5$ and $\Psi = 5$ in the HVNS.

$\underline{\varepsilon}$	Ψ	VRPD-C		VRPD-M		VRPD	
		Gap		Gap		Gap	
		Avg	Std Dev	Avg	Std Dev	Avg	Std Dev
1	1	0.70	1.02	0.71	1.18	0.70	1.10
1	5	0.66	0.88	0.39	1.03	0.52	0.97
1	10	0.54	0.89	0.50	0.83	0.52	0.86
5	1	0.09	0.71	0.44	1.27	0.27	1.05
5	**5**	**0.19**	**0.64**	**0.23**	**1.25**	**0.21**	**0.99**
5	10	0.21	0.58	0.82	1.49	0.51	1.17
10	1	1.35	2.92	1.16	1.87	1.25	2.45
10	5	1.28	2.94	1.09	1.72	1.19	2.41
10	10	1.21	2.61	1.19	1.66	1.20	2.19

Table 5: Evaluation of parameters $\underline{\varepsilon}$ and Ψ

6.2.2 Evaluation of operators

Given the parameter values established in Section 6.2.1, we evaluate all the operators in the HVNS, including 10 operators in the shaking phase and seven operators in the VND phase. We perform 17 sets of experiments to evaluate each operator by removing one (and maintaining the remaining 16) in each set. Table 6 shows the results of each configuration, where each line represents the configuration without the specific operator, and "All" denotes the configuration that includes all 17 operators.

Table 6 reports the averages and standard deviations of the gaps between the results of each parameter combination and the best average results among all parameter combinations (all calculations are

Operator	VRPD-C		VRPD-M		VRPD	
	Gap		Gap		Gap	
	Avg	Std Dev	Avg	Std Dev	Avg	Std Dev
Random destroy	0.28	0.72	1.24	1.76	0.76	1.43
Worst travel time destroy	0.46	0.82	0.84	1.60	0.65	1.29
Cluster destroy	0.60	0.97	0.89	1.64	0.75	1.35
Worst waiting-time destroy	0.55	0.97	0.28	1.22	0.41	1.11
One route destroy	0.42	0.90	0.96	1.60	0.69	1.32
Drone destroy	0.57	1.03	0.56	1.64	0.56	1.36
Random repair	0.55	1.19	0.38	1.34	0.46	1.27
Least travel time repair	0.31	0.71	1.68	2.06	0.99	1.69
Nearest polar angle repair	0.44	0.86	0.45	0.91	0.44	0.89
Split	0.51	0.92	0.48	1.48	0.49	1.24
Inter-route 1-1 swap	0.50	0.87	1.27	1.94	0.89	1.55
Inter-route 2-opt	2.33	3.84	1.11	2.27	1.72	3.21
Intra-route 1-1 swap	0.68	1.10	0.46	1.41	0.57	1.27
Intra-route 3-1 swap	0.34	0.90	0.64	1.54	0.49	1.27
Intra-route 1-0 relocate	1.32	1.50	1.25	1.58	1.29	1.54
Intra-route 3-0 relocate	0.39	0.75	0.74	1.78	0.57	1.38
Intra-route 2-opt	0.88	1.41	1.57	2.23	1.22	1.90
All	**0.41**	**0.92**	**0.27**	**1.26**	**0.34**	**1.11**

Table 6: Performance assessment of the 17 HVNS operators

performed as in Table 4). Despite that deactivating certain operators leads to slightly better objective values and smaller standard deviations for a particular problem variant, the "All" configuration yields the minimum average gap and relatively small standard deviations, showing that all proposed operators effectively contribute to the HVNS. Thus, we choose to retain all of them in the HVNS.

6.2.3 HDP and subHDP evaluation

In this section, we evaluate the added value of the HDP and the subHDP. We deactivate the subHDP and refer to this configuration as HVNS-NosubHDP. This is done by removing all calls to the `subHDP`$(*)$ in Algorithm 3, line 6 and Algorithm 4, line 7 in the HVNS. We then propose configuration HVNS-EDP, which similar to the HVNS-NosubHDP does not use the subHDP, and use the EDP (i.e., we replace the `HeuristicDP` by the `ExactDP` in Algorithms 1, 3 and 4). To have a comprehensive overview of the value of the EDP, the HDP and the subHDP, we compare the results of the HVNS with the HVNS-EDP and the HVNS-NosubHDP not only under a time limit of five minutes, but also under an alternative stopping criterion imposing a maximum number of iterations equal to 1000. In this case, we replace $\mathsf{T} \leqslant \mathsf{T}^{\mathsf{max}}$ by $\mathsf{I} \leqslant 1000$ in Algorithm 1. We refer to this version of the HVNS as HVNS-I1000.

Tables 7 and 8 report the comparison of the three configurations for the VRPD-C and the VRPD-M, respectively. For each stopping criterion, these two tables report the gaps between the results of each configuration δ and the best results among the three configurations, which is computed as $100(\overline{\text{obj}}_\delta - \overline{\text{obj}}^{\mathsf{BEST}})/\overline{\text{obj}}^{\mathsf{BEST}}$, where $\overline{\text{obj}}_\delta$ and $\overline{\text{obj}}^{\mathsf{BEST}}$ have the same meanings as in Section 6.2.1. These two tables also report the average number of iterations within five minutes, and the average computational time after 1000 iterations computed over five runs. Note that we stopped HVNS-I1000 at a time limit of 24 hours even if it is not completed.

In the HVNS (stopping criterion $\mathsf{T}^{\mathsf{max}} = 5$ min), the configuration that uses both HDP and subHDP significantly outperforms the other two configurations by obtaining all the minimum average objective

Instance	HVNS						HVNS-I1000					
	Gap^{E}	Gap^{H}	Gap^{SH}	Iter^{E}	Iter^{H}	Iter^{SH}	Gap^{E}	Gap^{H}	Gap^{SH}	Time^{E}	Time^{H}	Time^{SH}
20.10.1	0.00	0.00	**0.00**	35372	73439	96614	1.81	1.81	0.00	6	5	5
20.20.1	0.00	0.00	**0.00**	12809	32746	55860	0.00	0.00	0.00	24	11	10
50.10.1	0.00	0.00	**0.00**	129	1524	6259	0.00	0.00	0.00	2104	199	51
50.30.1	0.25	0.00	**0.00**	532	3471	11804	0.00	0.06	0.06	566	89	31
100.10.1	48.96	2.83	**0.00**	-	30	347	*0.90*	0.30	0.00	*86400* (*256*)	9878	898
100.30.1	8.92	0.82	**0.00**	5	227	1287	0.46	0.00	0.09	11830	1382	233
150.10.1	58.97	7.36	**0.00**	-	2	213	*4.53*	0.00	0.00	*86400* (*26*)	30695	1518
150.30.1	50.18	4.54	**0.00**	-	22	293	*0.16*	0.00	0.16	*86400* (*522*)	9678	1041
200.10.1	49.61	7.27	**0.00**	-	2	84	*10.14*	0.00	0.62	*86400* (*34*)	41579	2509
200.30.1	54.55	10.59	**0.00**	-	3	137	*0.81*	0.00	0.17	*86400* (*318*)	20166	2310

[1] The superscripts "E", "H", and "SH" in the column headings denote HVNS-EDP, HVNS-NosubHDP, and HVNS, respectively.

[2] The values underlined in italics are the results obtained after 24 hours for HVNS-I1000. The number given between parentheses is the number of iterations performed before reaching this time limit.

Table 7: Comparison of the HVNS-EDP, the HVNS-NosubHDP, and the HVNS for the VRPD-C

Instance	HVNS						HVNS-I1000					
	Gap^{E}	Gap^{H}	Gap^{SH}	Iter^{E}	Iter^{H}	Iter^{SH}	Gap^{E}	Gap^{H}	Gap^{SH}	Time^{E}	Time^{H}	Time^{SH}
s-61-n20	0.00	0.00	**0.00**	73548	120707	184478	0.00	0.00	0.00	7	5	4
u-61-n20	0.00	0.00	**0.00**	34178	66434	93559	0.00	0.00	0.00	12	8	7
s-71-n50	0.33	0.15	**0.00**	421	2006	6560	0.00	0.15	0.33	723	149	50
u-71-n50	0.39	0.52	**0.00**	521	2495	8727	0.00	0.69	0.95	585	119	40
s-81-n75	22.35	0.00	**0.18**	2	197	1384	0.00	0.41	1.20	16341	1474	219
u-81-n75	5.63	0.81	**0.00**	23	436	2886	0.00	0.74	1.82	5358	667	109
s-91-n100	31.65	3.36	**0.00**	2	29	398	0.00	0.15	0.62	94197	9059	775
u-91-n100	26.81	3.15	**0.00**	2	66	749	0.00	1.15	1.42	42585	3723	398
s-101-n175	55.46	29.00	**0.00**	-	2	66	*28.09*	0.00	0.20	*86400* (*9*)	76599	4040
u-101-n175	72.63	34.97	**0.00**	-	2	91	*0.77*	0.00	0.76	*86400* (*83*)	43789	3414

[1] The superscripts "E", "H", and "SH" in the column headings denote HVNS-EDP, HVNS-NosubHDP, and HVNS, respectively.

[2] The values underlined in italics are the results obtained after 24 hours for HVNS-I1000. The number given between parentheses is the number of iterations performed before reaching this time limit.

Table 8: Comparison of the HVNS-EDP, the HVNS-NosubHDP, and the HVNS for the VRPD-M

values for the VRPD-C and nine out of 10 minimum average objective values for the VRPD-M. In contrast, the HVNS-EDP and HVNS-NosubHDP only achieve the minimum average objective values for less than half of the total instances, mainly on small or medium-sized instances. As expected, the number of iterations decreases as the number of customers increases. Notably, the HVNS executes the highest number of iterations within five minutes. Due to the computational complexity of the EDP, the HVNS-EDP fails to conclude a single iteration on instances with more than 100 customers within the five-minute time limit. The HVNS-NosubHDP uses the HDP, which has a lower complexity than the EDP. However, the HVNS-NosubHDP performs 5 to 100 times fewer iterations than the HVNS on instances with more than 74 customers.

As for the HVNS-I1000, we observe that the HVNS-EDP and the HVNS-NosubHDP generally perform better than the HVNS. Considering the VRPD-C, the HVNS-NosubHDP obtains seven out of 10 minimum average objective values. Considering the VRPD-M, the HVNS-EDP obtains eight out of 10 minimum average objective values. These results are expected as the HVNS-EDP and the HVNS-NosubHDP use less approximations, when evaluating moves, compared to the HVNS. However, such approximations have a significant impact on run times. Specifically, the HVNS is 10 times faster than the HVNS-NosubHDP and up to 100 times faster than the HVNS-EDP. The speed of the HVNS coupled with the fact that it achieves relatively low gaps, implies that this algorithm lends itself to be used on large-sized instances. However, the HVNS-EDP and the HVNS-NosubHDP are not adequate for large-sized instances. In particular, the HVNS-EDP performs a limited number of iterations in 24 hours for instances with more than 100 customers. As our aim is to propose an effective heuristic for the VRPD, which is capable of handling large-sized instances, we opted to use the HVNS in the subsequent sections.

6.3 Comparison with state-of-the-art results

In this section, we compare the solutions computed by our HVNS with the solutions reported in the literature for the VRPD-C (see Section 6.3.1) and the VRPD-M (see Section 6.3.2).

6.3.1 Results for the VRPD-C

In Table 9, we compare our objective function values with the optimal solutions, obtained by CPLEX, reported by Sacramento et al. (2019) for all small-sized instances containing 6, 10, and 12 customers (see column "Opt"). We mention that we have not obtained better results by running CPLEX on the same model. Furthermore, in Table 9, we also compare the solutions obtained by our HVNS with the benchmarks SPR and RFK (as reported in Sacramento et al. (2019) and Rave et al. (2023)). Given that both benchmarks were executed 10 times with a time limit of five minutes, we also conduct 10 runs of five minutes. For each instance size, Table 9 reports the number of instances (#Ins), the average objective value of optimal solutions reported by Sacramento et al. (2019) (Opt), the average objective values of 10 runs ($\overline{\text{Obj}}$) and the best average objective values among 10 runs (Obj) obtained by SPR, RFK, and our HVNS. The BKS reports the average values of best-known solutions (BKSs) found from SPR and RFK over each instance size. $\overline{\text{Gap}}^{\text{SPR}}$ and $\overline{\text{Gap}}^{\text{RFK}}$ report the gaps between the average objective values of our HVNS and two benchmarks, while Gap^{BKS}, Gap^{SPR} and Gap^{RFK} report the gaps between the best average objective values of our HVNS and BKSs, as well as the two benchmarks. These gaps are calculated in a similar way. Given an instance, let $\overline{\text{obj}}^{\text{SPR}}$ and $\overline{\text{obj}}$ be the average objective value obtained by SPR and by our HVNS, we calculate the gap as $100(\overline{\text{obj}} - \overline{\text{obj}}^{\text{SPR}})/\overline{\text{obj}}^{\text{SPR}}$. We report the average of the gap over all instances with the same number of customers in column $\overline{\text{Gap}}^{\text{SPR}}$. In addition, for each instance size, Table 9 reports the number of matched optimal solutions (M^{OPT}), the number of matched and improved solutions compared with the BKSs (M^{BKS} and I^{BKS}) for our HVNS. Finally, the line Avg / Sum shows the total number of instances, overall average gaps, the total number of matched solutions, and the new best solutions we found compared with the BKSs from the two benchmarks.

We observe from Table 9 that our HVNS matches all the optimal solutions on instances with 6, 10, and 12 customers, and matches the BKSs on instances with 20 customers. Furthermore, our HVNS outperforms the two ALNS algorithms proposed by Sacramento et al. (2019) and Rave et al. (2023), reducing the average cost by 1.63% and 1.16%, respectively. Moreover, there are significant improvements for larger instances starting from 50 customers, where the cost has been reduced from 1.07% to 3.35% compared with the two ALNSs. Lastly, considering the 64 instances with 50 customers and more, we obtained 48 new BKSs. This again demonstrates the superiority of our HVNS on large-sized instances.

n	#Ins	SPR			RFK		BKS	HVNS							
		Opt	$\overline{\text{Obj}}$	Obj	$\overline{\text{Obj}}$	Obj		$\overline{\text{Obj}}$	Obj	$\overline{\text{Gap}}^{\text{SPR}}$	$\overline{\text{Gap}}^{\text{RFK}}$	Gap^{SPR}	Gap^{RFK}	Gap^{BKS}	$(\text{M}^{\text{OPT}}, \text{M}^{\text{BKS}}, \text{I}^{\text{BKS}})$
6	12	2.12	2.12	2.12	2.12	2.12	2.12	2.12	2.12	0.00	0.00	0.00	0.00	0.00	(12, 12, 0)
10	12	3.15	3.15	3.15	3.15	3.15	3.15	3.15	3.15	0.00	-0.02	0.00	0.00	0.00	(12, 12, 0)
12	12	3.36	3.36	3.36	3.36	3.36	3.36	3.36	3.36	0.00	0.00	0.00	0.00	0.00	(12, 12, 0)
20	12	-	4.12	4.11	4.09	4.09	4.09	4.09	4.09	-0.96	-0.02	-0.87	0.00	0.00	(-, 12, 0)
50	16	-	13.63	13.48	13.67	13.48	13.45	13.41	13.40	-1.55	-2.17	-0.56	-0.68	-0.38	(-, 8, 6)
100	16	-	18.66	18.34	18.65	18.41	18.27	18.09	18.03	-2.72	-2.79	-1.51	-1.89	-1.12	(-, 0, 15)
150	16	-	22.57	21.78	22.26	21.84	21.69	21.79	21.56	-3.35	-2.07	-1.05	-1.31	-0.70	(-, 0, 16)
200	16	-	26.93	26.32	26.47	25.96	25.90	26.09	25.69	-3.10	-1.07	-2.40	-1.07	-0.91	(-, 0, 11)
Avg / Sum	112									-1.63	-1.16	-0.88	-0.71	-0.44	(36, 56, 48)

Table 9: Comparison of results of HVNS and results reported by Sacramento et al. (2019) and Rave et al. (2023) for the VRPD-C

6.3.2 Results for the VRPD-M

We first evaluate the quality of the solutions computed by our HVNS for the VRPD-M by performing a comparison of the benchmarks `EGH`, `TB-1`, `TB-2`, `RR-RANGE30`, `RR-MHD` which are solved by exact algorithms (see Table 2). We ran five replications of our HVNS on each instance. The results are shown in Table 10. The columns $\overline{\text{Gap}}$ and Gap are computed in a similar way as in Table 9. We present the number of optimal solutions (#Opt) from each benchmark and the number of optimal solutions computed by our HVNS (M^{OPT}). We also report the number of matched and improved BKSs (M^{BKS} and I^{BKS}) compared with the BKSs from each benchmark.

Despite that these benchmarks only have small and medium-sized instances, we still observe several advantages of our HVNS. Among the 798 instances considered solved by the exact algorithms, our HVNS obtains 615 out of 644 optimal solutions, matches 655 BKSs and identifies 119 new BKSs. As the instance size becomes larger, our HVNS finds more new BKSs. Specifically, it identifies 50 new BKSs for the 75 instances with 39 customers compared to `RR-RANGE30`, achieving a significant improvement with a gap of -31.91%. Besides, it identifies 22 new BKSs for the 30 instances with 31 customers compared to `EGH` and 11 new BKSs for the 75 instances with 39 customers compared to `RR-MHD`, with a gap of -3.65% and -5.31%. In addition, the differences between the values of the columns $\overline{\text{Gap}}$ and Gap are very small, which means our HVNS is relatively stable.

We now compare our algorithm to a heuristic introduced by Kitjacharoenchai et al. (2019) for the VRPD-M on benchmarks `KVMLTB-1` and `KVMLTB-M`. This heuristic, referred to as the Adapted FSTSP heuristic, is the adaptation of the heuristic introduced by Murray and Chu (2015). Truck tours are first generated using heuristic procedures, and then the FSTSP heuristic of Murray and Chu (2015) is used to insert drone sorties within each tour. Similar to Kitjacharoenchai et al. (2019), we run our algorithm 20 times. Table 11 reports the performance of our HVNS compared with `KVMLTB-1` and `KVMLTB-M`, where the results are computed in a similar way as in Table 10 but summarized based on the same number of trucks rather than the same number of customers. We observe that our HVNS outperforms the heuristic algorithm presented by Kitjacharoenchai et al. (2019) by yielding better solutions for all 22 instances, and achieving an average gap of around -20%.

Benchmark	n	m	#Ins	#Opt	BKS	Time (s)	HVNS				
							Obj	$\overline{\text{Obj}}$	Gap	$\overline{\text{Gap}}$	$(\text{M}^{\text{OPT}}, \text{M}^{\text{BKS}}, \text{I}^{\text{BKS}})$
`EGH`	15	1	30	30	515.92	64.85	515.92	515.92	0.00	0.00	(27, 27, 3)
	19		30	28	549.59	1979.28	549.59	549.59	0.00	0.00	(26, 28, 2)
	23		30	17	550.32	8241.03	549.55	549.55	-0.21	-0.21	(17, 24, 6)
	27		30	5	581.54	10071.10	578.61	578.61	-0.65	-0.65	(5, 14, 16)
	31		30	2	632.08	10794.95	611.73	611.74	-3.65	-3.65	(1, 7, 22)
Avg / Sum			150	82	565.89	6,230.24	561.08	561.08	-0.90	-0.90	(76, 100, 49)
`TB-1`	14	1	6	6	6785.28	137.53	6785.28	6,785.28	0.00	0.00	(6, 6, 0)
	19		6	6	7680.69	15252.30	7680.69	7680.69	0.00	0.00	(6, 6, 0)
	24		6	3	8764.17	23632.88	8764.17	8764.17	0.00	0.00	(3, 6, 0)
	29		6	2	9532.61	33966.54	9532.62	9532.62	0.00	0.00	(1, 5, 0)
Avg / Sum			24	17	8190.69	18247.31	8190.69	8,190.69	0.00	0.00	(16, 23, 0)
`TB-2`	14	2	6	6	4380.72	24.68	4452.38	4452.38	1.55	1.55	(5, 5, 0)
	19		6	5	4840.82	779.07	4840.82	4840.82	0.00	0.00	(5, 6, 0)
	24		6	4	6066.45	5972.25	6046.80	6046.80	-0.31	-0.31	(3, 4, 1)
	29		6	1	6444.68	12668.54	6423.84	6423.84	-0.31	-0.31	(1, 3, 3)
Avg / Sum			24	16	5433.17	4,861.13	5440.96	5440.96	0.23	0.23	(14, 18, 4)
`RR-RANGE30`	9	1	75	75	121.09	0.25	121.09	121.09	0.00	0.00	(75, 75, 0)
	19		75	75	151.49	20.24	151.49	151.49	0.00	0.00	(75, 75, 0)
	29		75	68	175.65	755.20	168.65	168.69	-2.16	-2.15	(64, 66, 5)
	39		75	25	307.27	2896.68	187.15	187.42	-31.91	-31.81	(24, 24, 50)
Avg / Sum			300	243	188.88	918.10	157.10	157.17	-8.52	-8.49	(238, 240, 55)
`RR-MHD`	9	1	75	75	152.37	0.13	152.68	152.68	0.21	0.21	(73, 73, 0)
	19		75	75	188.15	3.18	188.31	188.31	0.09	0.10	(73, 73, 0)
	29		75	75	210.24	92.29	210.31	210.38	0.03	0.07	(71, 71, 0)
	39		75	61	256.72	1475.88	236.12	236.38	-5.31	-5.21	(54, 57, 11)
Avg / Sum			300	286	201.87	392.87	196.85	196.94	-1.24	-1.21	(271, 274, 11)
Sum			798	644							(615, 655, 119)

Table 10: Comparison of the HVNS results against five benchmarks solved by exact algorithms for the VRPD-M

Benchmark	m	#Ins	BKS	Time (s)	HVNS				
					Obj	$\overline{\text{Obj}}$	Gap	$\overline{\text{Gap}}$	$(\text{M}^{\text{BKS}}, \text{I}^{\text{BKS}})$
`KVMLTB-1`	1	10	4349.61	27	3252.36	3253.44	-23.93	-23.73	(0, 10)
`KVMLTB-M`	2	4	1206.46	61	922.03	929.33	-26.16	-25.28	(0, 4)
	3	4	905.90	48	719.20	725.42	-22.51	-21.88	(0, 4)
	5	4	679.42	53	571.97	573.84	-18.32	-17.86	(0, 4)
Avg / Sum		22	2484.69	42	1880.75	1884.03	-23.06	-22.61	(0, 22)

Table 11: Comparison of the HVNS results against two benchmarks from Kitjacharoenchai et al. (2019) for the VRPD-M

7 Conclusions

In this work, we consider several variants of the VRPD depending on whether the objective function is related to minimizing the total transportation cost or the makespan, on whether the drone is either allowed or not allowed to land while awaiting recovery, and on the number of trucks. We present a hybrid variable neighborhood search (HVNS) to solve the VRPD, where we represent a VRPD solution as a set of customer sequences and evaluate it using local search procedure solving for each sequence a problem that we refer to as the fixed route drone dispatch problem (FRDDP). Given a sequence of customers to be served by a single truck and its drone, the FRDDP selects a subset of customers for drone service and determines drone launch and recovery nodes for each selected customer, while ensuring that the served customer is positioned between these two nodes in the initial sequence. For the FRDDP with different objective functions, we develop a heuristic dynamic program (i.e., HDP) with a computational complexity of $\mathcal{O}(n^2)$, where n is the number of customers contained in the route, achieving a lower computational complexity than exact dynamic programs ($\mathcal{O}(n^3)$) proposed in the literature. In our HVNS, we solve the FRDDP using the HDP, and introduce mechanisms to filter out unpromising moves by approximating their objective function value savings. Numerous experiments conducted on two data sets for both objective functions demonstrate the important role of these components to reduce the computational time and to perform a larger exploration of the search space within the time limit. The comparison of the solutions computed by the HVNS on several variants of the problem against nine benchmark sets from the literature shows its high-quality results which are reflected in the identification of a large number of new BKSs, as shown in Table 12. The results demonstrate the relevance of designing a local search algorithm that solves FRDDPs to evaluate solutions.

Problem		Name	Reference	Method	Time limit	Data set	$\mathrm{M}^{\mathsf{OPT}}$	$\mathrm{M}^{\mathsf{BKS}}$	$\mathrm{I}^{\mathsf{BKS}}$
VRPD-C	NoLW	`SPR` ($m \geqslant 1$)	Sacramento et al. (2019)	ALNS	5 minutes (10 runs)	Sacramento et al. (2019)	36/36	53/112	57/112
		`RFK` ($m \geqslant 1$)	Rave et al. (2023)					53/112	51/112
VRPD-M	LW	`EGH` ($m = 1$)	El-Adle et al. (2021)	MIP model	3.1 hours	Bouman et al. (2018b)	76/82	100/150	49/150
		`TB-1` ($m = 1$)	Tamke and Buscher (2021)	Branch-and-cut	12 hours	Reinelt (1997)	16/17	23/24	0/24
		`TB-2` ($m = 2$)					14/16	18/24	4/24
		`RR-RANGE30` ($m = 1$)	Roberti and Ruthmair (2021)	Branch-and-price	1 hour	Poikonen et al. (2019)	238/243	240/300	55/300
		`KVMLTB-1` ($m = 1$)	Kitjacharoenchai et al. (2019)	Adapted FSTSP heuristic	No (20 runs)	Reinelt (1997)		0/10	10/10
		`KVMLTB-M` ($m > 1$)						0/12	12/12
	NoLW	`RR-MHD` ($m = 1$)	Roberti and Ruthmair (2021)	Branch-and-price	1 hour	Poikonen et al. (2019)	271/286	274/300	11/300

Table 12: Summary of computational results compared with all benchmarks

We believe that the concepts of heuristic dynamic programs put forward in this paper may be instrumental for a variety of synchronization problems involving multiple modes. It may also be worth investigating whether, and if so how, the algorithmic components, especially the HDP and the move evaluation mechanism subHDP proposed in this work, can be adapted to tackle more complex variants of the VRPD (e.g., time windows for the customer visits, or cyclic operations where the truck can wait for the drone at the same location).

Acknowledgements

This work was partially supported by the Innovation Foundation for Doctor Dissertation of Northwestern Polytechnical University (CX2021090). The first author thanks the support of the China Scholarship Council. The second author thanks the financial support of the Department of Elettronica, Informazione e Bioingeneria of Politecnico di Milano that made this collaboration possible.

References

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

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

Bouman, P., Agatz, N., and Schmidt, M. (2018b). Instances for the TSP with drone (and some solutions). `https://doi.org/10.5281/zenodo.1204676`. Accessed: 2023-1-9.

Campbell, J. F., Sweeney, D., and Zhang, J. (2017). Strategic design for delivery with trucks and drones. *Supply Chain Analytics Report SCMA (04 2017)*, pages 47–55.

Chen, C., Demir, E., and Huang, Y. (2021). An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots. *European Journal of Operational Research*, 294(3):1164–1180.

Chung, S. H., Sah, B., and Lee, J. (2020). Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research*, 123:105004.

Dell'Amico, M., Montemanni, R., and Novellani, S. (2021). Algorithms based on branch and bound for the flying sidekick traveling salesman problem. *Omega*, 104:102493.

Demir, E., Bektaş, T., and Laporte, G. (2012). An adaptive large neighborhood search heuristic for the pollution-routing problem. *European journal of operational research*, 223(2):346–359.

Drexl, M. (2012). Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. *Transportation Science*, 46(3):297–316.

El-Adle, A. M., Ghoniem, A., and Haouari, M. (2021). Parcel delivery by vehicle and drone. *Journal of the Operational Research Society*, 72(2):398–416.

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

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

Ha, Q. M., Deville, Y., Pham, Q. D., and Hà, M. H. (2020). A hybrid genetic algorithm for the traveling salesman problem with drone. *Journal of Heuristics*, 26(2):219–247.

Kharpal, A. (2016). This firm beat amazon to drone deliveries by launching it from the roof of a truck. https://www.cnbc.com/2016/08/18/this-firm-beat-amazon-to-drone-deliveries-by-launching-it-from-the-roof-of-a-truck.html. Accessed: 2023-06-22.

Kirschstein, T. (2020). Comparison of energy demands of drone-based and ground-based parcel delivery services. *Transportation Research Part D: Transport and Environment*, 78:102209.

Kitjacharoenchai, P., Min, B.-C., and Lee, S. (2020). Two echelon vehicle routing problem with drones in last mile delivery. *International Journal of Production Economics*, 225:107598.

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

Li, H., Chen, J., Wang, F., and Bai, M. (2021). Ground-vehicle and unmanned-aerial-vehicle routing problems from two-echelon scheme perspective: A review. *European Journal of Operational Research*, 294(3):1078–1095.

Macrina, G., Laporte, G., Guerriero, F., and Pugliese, L. D. P. (2019). An energy-efficient green-vehicle routing problem with mixed vehicle fleet, partial battery recharging and time windows. *European Journal of Operational Research*, 276(3):971–982.

Macrina, G., Pugliese, L. D. P., Guerriero, F., and Laporte, G. (2020). Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies*, 120:102762.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100.

Moshref-Javadi, M., Hemmati, A., and Winkenbach, M. (2020). A truck and drones model for last-mile delivery: A mathematical model and heuristic approach. *Applied Mathematical Modelling*, 80:290–318.

Moshref-Javadi, M. and Winkenbach, M. (2021). Applications and research avenues for drone-based models in logistics: A classification and review. *Expert Systems with Applications*, 177:114854.

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

Najy, W., Archetti, C., and Diabat, A. (2022). Collaborative truck-and-drone delivery for inventory-routing problems. *Transportation Research Part C: Emerging Technologies*, page 103791.

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

Poikonen, S. and Campbell, J. F. (2021). Future directions in drone routing research. *Networks*, 77(1):116–126.

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

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

Rave, A., Fontaine, P., and Kuhn, H. (2023). Drone location and vehicle fleet planning with trucks and aerial drones. *European Journal of Operational Research*, 308(1):113–130.

Reinelt, G. (1997). Tsplib. https://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html. Accessed: 2022-10-11.

Roberti, R. and Ruthmair, M. (2021). Exact methods for the traveling salesman problem with drone. *Transportation Science*, 55(2):315–335.

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

Sah, B., Gupta, R., and Bani-Hani, D. (2021). Analysis of barriers to implement drone logistics. *International Journal of Logistics Research and Applications*, 24(6):531–550.

Salama, M. and Srinivas, S. (2020). Joint optimization of customer location clustering and drone-based routing for last-mile deliveries. *Transportation Research Part C: Emerging Technologies*, 114:620–642.

Salama, M. R. and Srinivas, S. (2022). Collaborative truck multi-drone routing and scheduling problem: Package delivery with flexible launch and recovery sites. *Transportation Research Part E: Logistics and Transportation Review*, 164:102788.

Schermer, D., Moeini, M., and Wendt, O. (2019a). A hybrid VNS/tabu search algorithm for solving the vehicle routing problem with drones and en route operations. *Computers & Operations Research*, 109:134–158.

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

Singh, A. and Mutreja, S. (2022). Last mile delivery market. https://www.alliedmarketresearch.com/last-mile-delivery-market. Accessed: 2022-09-27.

Tamke, F. and Buscher, U. (2021). A branch-and-cut algorithm for the vehicle routing problem with drones. *Transportation Research Part B: Methodological*, 144:174–203.

Vidal, T. (2022). Hybrid genetic search for the CVRP: Open-source implementation and swap* neighborhood. *Computers & Operations Research*, 140:105643.

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

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

Zhen, L., Gao, J., Tan, Z., Wang, S., and Baldacci, R. (2023). Branch-price-and-cut for trucks and drones cooperative delivery. *IISE Transactions*, 55(3):271–287.

Zhou, H., Qin, H., Cheng, C., and Rousseau, L.-M. (2023). An exact algorithm for the two-echelon vehicle routing problem with drones. *Transportation Research Part B: Methodological*, 168:124–150.

Supplementary material

A MILP models for the VRPD-C and the VRPD-M

We first present a MILP model for the VRPD-C assuming the NoLW, and the number of trucks $m = |\mathcal{H}|$, where $|\mathcal{H}| > 1$. The decision variables are listed in Table 1. The continuous variable t_i^h indicates the earliest time at which the truck h can launch its drone at i or serve the customer if no drone launch is performed. If the drone is recovered by the truck at i, then t_i^h is the time when the recovery is complete. Otherwise, it is the arrival time of the truck at i. The drone's corresponding ready time $t_i'^h$ is equal to t_i^h if the drone is either launched or recovered by the truck h at customer i. Otherwise, $t_i'^h$ is the earliest time at which the drone can serve customer i.

$$\text{(VRPD-C)} \quad \min f^{\text{cost}} = \sum_{h\in\mathcal{H}} \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij}^h + \sum_{h\in\mathcal{H}} \sum_{\langle i,j,k\rangle\in\Delta} (c'_{ij} + c'_{jk}) y_{ijk}^h \tag{19}$$

subject to

$$\sum_{h\in\mathcal{H}} \Big(\sum_{i:(i,j)\in\mathcal{A}} x_{ij}^h + \sum_{i,k:\langle i,j,k\rangle\in\Delta} y_{ijk}^h \Big) = 1 \qquad \forall j \in \mathcal{C} \tag{20}$$

$$\sum_{j\in\mathcal{N}\setminus\{0\}} x_{0j}^h \leqslant 1 \qquad \forall h \in \mathcal{H} \tag{21}$$

$$\sum_{i\in\mathcal{N}\setminus\{n+1\}} x_{i,n+1}^h \leqslant 1 \qquad \forall h \in \mathcal{H} \tag{22}$$

$$x_{0,n+1}^h = 0 \qquad \forall h \in \mathcal{H} \tag{23}$$

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij}^h = \sum_{k:(j,k)\in\mathcal{A}} x_{jk}^h \qquad \forall h \in \mathcal{H}, j \in \mathcal{C} \tag{24}$$

$$p_i^h + 1 \leqslant p_j^h + |\mathcal{N}|(1 - x_{ij}^h) \qquad \forall h \in \mathcal{H}, (i,j) \in \mathcal{A} \tag{25}$$

$$p_j^h \leqslant |\mathcal{N}| \sum_{i:(i,j)\in\mathcal{A}} x_{ij}^h \qquad \forall h \in \mathcal{H}, j \in \mathcal{N}\setminus\{0\} \tag{26}$$

$$\sum_{j\in\mathcal{C}} q_j \Big(\sum_{i:(j,i)\in\mathcal{A}} x_{ji}^h + \sum_{i,k:\langle i,j,k\rangle\in\Delta} y_{ijk}^h \Big) \leqslant Q \qquad \forall h \in \mathcal{H} \tag{27}$$

$$\sum_{j,k:\langle i,j,k\rangle\in\Delta} y_{ijk}^h \leqslant 1 \qquad \forall h \in \mathcal{H}, i \in \mathcal{N}\setminus\{n+1\} \tag{28}$$

$$\sum_{i,j:\langle i,j,k\rangle\in\Delta} y_{ijk}^h \leqslant 1 \qquad \forall h \in \mathcal{H}, k \in \mathcal{N}\setminus\{0\} \tag{29}$$

$$2y_{ijk}^h \leqslant \sum_{l:(i,l)\in\mathcal{A}} x_{il}^h + \sum_{l:(l,k)\in\mathcal{A}} x_{lk}^h \qquad \forall h \in \mathcal{H}, \langle i,j,k\rangle \in \Delta \tag{30}$$

$$t_0^h = 0 \qquad \forall h \in \mathcal{H} \tag{31}$$

$$t_0'^h = 0 \qquad \forall h \in \mathcal{H} \tag{32}$$

$$t_{n+1}^h \leqslant T \sum_{i\in\mathcal{N}\setminus\{n+1\}} x_{i,n+1}^h \qquad \forall h \in \mathcal{H} \tag{33}$$

$$t_{n+1}'^h \leqslant T \sum_{i,j:\langle i,j,n+1\rangle\in\Delta} y_{i,j,n+1}^h \qquad \forall h \in \mathcal{H} \tag{34}$$

$$t_i^h + \hat{\tau} \sum_{j,l:\langle i,j,l\rangle\in\Delta} y_{ijl}^h + s_i + \tau_{ik} + \check{\tau} \sum_{l,j:\langle l,j,k\rangle\in\Delta} y_{ljk}^h \leqslant t_k^h + T(1 - x_{ik}^h) \qquad \forall h \in \mathcal{H}, (i,k) \in \mathcal{A} \tag{35}$$

$$t_i'^h + \hat{\tau} + \tau'_{ij} \leqslant t_j'^h + T(1 - \sum_{k:\langle i,j,k\rangle\in\Delta} y_{ijk}^h) \qquad \forall h \in \mathcal{H}, i \in \mathcal{N}\setminus\{n+1\}, j \in \mathcal{C}'\setminus\{i\} \tag{36}$$

$$t_j'^h + s_j' + \tau_{jk}' + \breve{\tau} \leqslant t_k'^h + T(1 - \sum_{i:\langle i,j,k\rangle \in \Delta} y_{ijk}^h) \qquad \forall h \in \mathcal{H}, j \in \mathcal{C}', k \in \mathcal{N}\backslash\{0\} \cup \{j\} \tag{37}$$

$$t_i^h - T(1 - \sum_{j,k:\langle i,j,k\rangle \in \Delta} y_{ijk}^h) \leqslant t_i'^h \qquad \forall h \in \mathcal{H}, i \in \mathcal{N}\backslash\{n+1\} \tag{38}$$

$$t_i^h + T(1 - \sum_{j,k:\langle i,j,k\rangle \in \Delta} y_{ijk}^h) \geqslant t_i'^h \qquad \forall h \in \mathcal{H}, i \in \mathcal{N}\backslash\{n+1\} \tag{39}$$

$$t_k^h - T(1 - \sum_{i,j:\langle i,j,k\rangle \in \Delta} y_{ijk}^h) \leqslant t_k'^h \qquad \forall h \in \mathcal{H}, k \in \mathcal{C} \tag{40}$$

$$t_k^h + T(1 - \sum_{i,j:\langle i,j,k\rangle \in \Delta} y_{ijk}^h) \geqslant t_k'^h \qquad \forall h \in \mathcal{H}, k \in \mathcal{C} \tag{41}$$

$$T' + T(1 - \sum_{j:\langle i,j,k\rangle \in \Delta} y_{ijk}^h) \geqslant t_k'^h - t_i'^h \qquad \forall h \in \mathcal{H}, i \in \mathcal{N}\backslash\{n+1\}, k \in \mathcal{N}\backslash\{0\} \cup \{i\} \tag{42}$$

$$p_j^h - p_i^h \leqslant |\mathcal{N}| w_{ij}^h \qquad \forall h \in \mathcal{H}, i \in \mathcal{N}\backslash\{n+1\}, j \in \mathcal{C}\backslash\{i\} \tag{43}$$

$$p_j^h - p_i^h \geqslant |\mathcal{N}|(w_{ij}^h - 1) + 1 \qquad \forall h \in \mathcal{H}, i \in \mathcal{N}\backslash\{n+1\}, j \in \mathcal{C}\backslash\{i\} \tag{44}$$

$$t_k^h - T\big(3 - \sum_{j:\langle i,j,k\rangle \in \Delta} y_{ijk}^h - \sum_{l,g:\langle f,l,g\rangle \in \Delta} y_{flg}^h - w_{if}^h\big) \leqslant t_f'^h \qquad \forall h \in \mathcal{H}, i \in \mathcal{N}\backslash\{n+1\}, k \in \mathcal{N}\backslash\{0\}, f \in \mathcal{C}\backslash\{i\} \tag{45}$$

$$x_{ij}^h \in \{0,1\} \qquad \forall h \in \mathcal{H}, (i,j) \in \mathcal{A} \tag{46}$$

$$y_{ijk}^h \in \{0,1\} \qquad \forall h \in \mathcal{H}, \langle i,j,k\rangle \in \Delta \tag{47}$$

$$w_{ij}^h \in \{0,1\} \qquad \forall h \in \mathcal{H}, i \in \mathcal{N}\backslash\{n+1\}, j \in \mathcal{N}\backslash\{0\} \tag{48}$$

$$t_i^h, t_i'^h \geqslant 0 \qquad \forall h \in \mathcal{H}, i \in \mathcal{N} \tag{49}$$

$$0 \leqslant p_i^h \leqslant |\mathcal{N}|, p_i^h \in \mathbb{Z}^+ \qquad \forall h \in \mathcal{H}, i \in \mathcal{N} \tag{50}$$

The objective function (19) minimizes the total transportation cost of both trucks and drones. Constraints (20) ensure that each customer is served exactly once by a truck or a drone. Constraints (21) and (22) guarantee that all trucks must depart from and return to the depot at most once. Constraints (23) prohibit traveling between depots. Constraints (24) enforce the flow conservation for a truck tour. Constraints (25) and (26) eliminate subtours for the truck. Constraints (27) state that the total demand served by the truck and its carried drone cannot exceed the capacity of the truck. Constraints (28) and (29) make sure that the drone can be launched and recovered at most once at a node. Constraints (30) ensure that if a drone is launched at node $i \in \mathcal{N}\backslash\{n+1\}$ and recovered $k \in \mathcal{N}\backslash\{0\}$, then its truck must visits nodes i and k. Note that a truck h may visit several nodes between the launch of the drone at a node i and its recovery at a node k. Constraints (31) and (32) initialize the time for trucks and drones at the beginning of each route. Constraints (33) and (34) enforce the maximum route duration for trucks and drones. Constraints (35) describe the truck movement together with a drone sortie. Correspondingly, constraints (36) and (37) enforce the ready time of the drones. Constraints (38) – (41) ensure the synchronization between the truck and its carried drone. Constraints (42) ensure the NoLW assumption. Specifically, they limit the travel time of both the truck and the drone between the launch of the drone and its recovery according to the flight endurance T'. Constraints (43) and (44) ensure the customer order of each truck. Constraints (45) enforce that the drone can only serve another customer after being recovered. Finally, constraints (46)–(50) define the domain of decision variables.

We now derive a MILP model for the VRPD-M (assuming the NoLW, and $m > 1$) from the one introduced for the VRPD-C. For this purpose, we introduce an additional variable t equal to the makespan, i.e., the maximum duration of all routes. We obtain a formulation for the VRPD-M by replacing the objective function (19) by (51) and adding constraints (52) and (53).

$$\text{(VRPD-M)} \quad \min f^{\text{makespan}} = t \tag{51}$$

$$\text{s.t.} \quad (20) - (50)$$

$$t \geqslant t^{h}_{n+1} \qquad \forall h \in \mathcal{H} \tag{52}$$

$$t \geqslant t'^{h}_{n+1} \qquad \forall h \in \mathcal{H} \tag{53}$$

We now discuss how to adapt the above formulations to other variants. The above two formulations consider a fleet of multiple trucks. Single truck cases can be handled by setting m to one. Moreover, to obtain the model considering the LW assumption, we remove (42). The MILP models for every problem variant are summarized in Table 13.

	NoLW	LW
VRPD-C	min (19) s.t.(20)–(50)	min (19) s.t.(20)–(41), (43)–(50)
VRPD-M	min (51) s.t.(20)–(50), (52)–(53)	min (51) s.t.(20)–(41), (43)–(50) (52)–(53)

Table 13: The MILP models for different problem variants

B Neighborhood exploration for inter-route moves

Algorithm 4: Search($\mathcal{S}^{\mathsf{inc}}, \overline{\mathcal{S}}^{\mathsf{inc}}, \lambda, \lambda^{\mathsf{avg}}, \lambda^{\mathsf{max}}, \mathsf{I}, \mathcal{K}_\vartheta$)

```
Data: Incumbent truck solution $\mathcal{S}^{\mathsf{inc}}$, its complete solution $\overline{\mathcal{S}}^{\mathsf{inc}}$, duration threshold $\mu$, duration variation
      threshold $\lambda$, the average duration variation threshold $\lambda^{\mathsf{avg}}$, the maximum duration variation
      threshold $\lambda^{\mathsf{max}}$, the current iteration number $\mathsf{I}$, an operator $\mathcal{K}_\vartheta$
1  *improved* $\leftarrow \mathsf{false}, \varsigma \leftarrow 0, \mathcal{S}^{\mathsf{new}} \leftarrow \mathcal{S}^{\mathsf{inc}}, \overline{\mathcal{S}}^{\mathsf{new}} \leftarrow \overline{\mathcal{S}}^{\mathsf{inc}}$ ;
2  **if** $\mathcal{K}_\vartheta$ *defines inter-route moves* **then**
3      **for** *each pair of neighboring truck route* $(\mathcal{R}_1^{\mathsf{new}}, \mathcal{R}_2^{\mathsf{new}})$ *generated from* $(\mathcal{R}_1^{\mathsf{inc}}, \mathcal{R}_2^{\mathsf{inc}}) \in \mathcal{S}^{\mathsf{inc}}$ **do**
4          **if** $\sum_{i \in \mathcal{R}_1^{\mathsf{new}}} q_i \leqslant Q$ *and* $\sum_{i \in \mathcal{R}_2^{\mathsf{new}}} q_i \leqslant Q$ **then**
5              **if** $T_{\mathcal{R}_1^{\mathsf{new}}} \leqslant \mu T$ *and* $T_{\mathcal{R}_2^{\mathsf{new}}} \leqslant \mu T$ **then**
6                  **if** $T_{\mathcal{R}_1^{\mathsf{new}}} - T_{\mathcal{R}_1^{\mathsf{inc}}} \leqslant \lambda$ *or* $T_{\mathcal{R}_2^{\mathsf{new}}} - T_{\mathcal{R}_2^{\mathsf{inc}}} \leqslant \lambda$ **then**
7                      **if** subHDP $(\mathcal{R}_1^{\mathsf{new}}, \mathcal{R}_1^{\mathsf{inc}})$+subHDP $(\mathcal{R}_2^{\mathsf{new}}, \mathcal{R}_2^{\mathsf{inc}}) > 0$ **then**
8                          $\overline{\mathcal{R}}_1^{\mathsf{new}} \leftarrow$ HeuristicDP($\mathcal{R}_1^{\mathsf{new}}$), $\overline{\mathcal{R}}_2^{\mathsf{new}} \leftarrow$ HeuristicDP($\mathcal{R}_2^{\mathsf{new}}$) ;
9                          **if** $F(\overline{\mathcal{R}}_1^{\mathsf{new}}) + F(\overline{\mathcal{R}}_2^{\mathsf{new}}) < F(\overline{\mathcal{R}}_1^{\mathsf{inc}}) + F(\overline{\mathcal{R}}_2^{\mathsf{inc}})$ **then**
10                             $\overline{\mathcal{S}}^{\mathsf{new}} \leftarrow \overline{\mathcal{R}}_1^{\mathsf{new}} \cup \overline{\mathcal{R}}_2^{\mathsf{new}} \cup \overline{\mathcal{S}}^{\mathsf{inc}} \backslash \{\overline{\mathcal{R}}_1^{\mathsf{inc}}, \overline{\mathcal{R}}_2^{\mathsf{inc}}\}, \mathcal{S}^{\mathsf{new}} \leftarrow \mathcal{R}_1^{\mathsf{new}} \cup \mathcal{R}_2^{\mathsf{new}} \cup \mathcal{S}^{\mathsf{inc}} \backslash \{\mathcal{R}_1^{\mathsf{inc}}, \mathcal{R}_2^{\mathsf{inc}}\}$;
11                             **if** $\mathsf{I} = 0$ **then** update $\lambda^{\mathsf{max}}$,
                                 $\lambda^{\mathsf{avg}} \leftarrow \big(\lambda^{\mathsf{avg}}\varsigma + (T_{\mathcal{R}_1^{\mathsf{new}}} - T_{\mathcal{R}_1^{\mathsf{inc}}}) + (T_{\mathcal{R}_2^{\mathsf{new}}} - T_{\mathcal{R}_2^{\mathsf{inc}}})\big)/(\varsigma + 2)$, $\varsigma \leftarrow \varsigma + 2$;
12                             **else** $\lambda^{\mathsf{avg}} \leftarrow \lambda$;
13                             *improved* $\leftarrow \mathsf{true}$;   **break**;
Result: $\mathcal{S}^{\mathsf{new}}, \overline{\mathcal{S}}^{\mathsf{new}}, \lambda^{\mathsf{avg}}, \lambda^{\mathsf{max}}$, *improved*
```

C Acceptance criterion of the HVNS

To avoid being trapped in a local optimum, in addition to the shaking phase, we use an SA-based solution acceptance criterion (see Algorithm 5). If the new solution $\overline{\mathcal{S}}^{\mathsf{new}}$ has an improved objective function value than the incumbent solution $\overline{\mathcal{S}}^{\mathsf{inc}}$, we always accept it. Otherwise, it is accepted with a probability $e^{\frac{F(\overline{\mathcal{S}}^{\mathsf{inc}})-F(\overline{\mathcal{S}}^{\mathsf{new}})}{t^c}}$, where $F(\overline{\mathcal{S}}^{\mathsf{inc}})$ and $F(\overline{\mathcal{S}}^{\mathsf{new}})$ denote the objective function value of $\overline{\mathcal{S}}^{\mathsf{inc}}$ and $\overline{\mathcal{S}}^{\mathsf{new}}$, and t^c controls the acceptance probability. The value of t^c is computed as $t^c = t^{\text{init}}F(\overline{\mathcal{S}}^{*})$, where t^{init} is the temperature factor and $F(\overline{\mathcal{S}}^{*})$ is the objective function value of the current best solution.

The best solution $\overline{\mathcal{S}}^{*}$ computed so far by the HVNS is restored if $\mathsf{N}^{\mathsf{max}}$ iterations have passed without any improvement (line 5). If an improved solution is found (line 2) or no improvement happens in a certain number of iterations (line 5), the threshold λ used in the procedure `Search` is set to λ^{avg}. Otherwise, λ is increased based on the number of iterations without improvement κ (line 8).

Algorithm 5: SAAcceptanceCriterion($\overline{\mathcal{S}}^{\mathsf{inc}}, \overline{\mathcal{S}}^{*}, \overline{\mathcal{S}}^{\mathsf{new}}, \lambda, \lambda^{\mathsf{avg}}, \lambda^{\mathsf{max}}, \kappa$)

Data: The incumbent complete solution $\overline{\mathcal{S}}^{\mathsf{inc}}$, the current best complete solution $\overline{\mathcal{S}}^{*}$ computed so far by the HVNS, a new complete solution $\overline{\mathcal{S}}^{\mathsf{new}}$, a duration threshold λ, the average duration variation threshold λ^{avg}, the maximum duration variation threshold λ^{max}, the current number of iterations without improvement κ

```
1 if Random(0,1) < $exp((F(\overline{\mathcal{S}}^{inc}) - F(\overline{\mathcal{S}}^{new}))/t^c)$ then $\overline{\mathcal{S}}^{inc} \leftarrow \overline{\mathcal{S}}^{new}$;
2 if $\overline{\mathcal{S}}^{inc}$ is feasible and $F(\overline{\mathcal{S}}^{inc}) < F(\overline{\mathcal{S}}^{*})$ then $\overline{\mathcal{S}}^{*} \leftarrow \overline{\mathcal{S}}^{inc}$, $\kappa \leftarrow 0$, $\lambda \leftarrow \lambda^{avg}$ ;
3 else
4     $\kappa \leftarrow \kappa + 1$;
5     if $\kappa > N^{max}$ then $\overline{\mathcal{S}}^{inc} \leftarrow \overline{\mathcal{S}}^{*}$, $\kappa \leftarrow 0$, $\lambda \leftarrow \lambda^{avg}$ ;
6     else
7         if $\lambda < \lambda^{max}$ then
8             $\lambda \leftarrow \lambda + \kappa$Random(0,1);
```

Result: $\overline{\mathcal{S}}^{\mathsf{inc}}, \overline{\mathcal{S}}^{*}, \lambda, \kappa$

D Hardware comparison

We conducted a comparison between the hardware reported in benchmarks `SPR`, `RFK`, `KVMLTB-1`, `KVMLTB-M` and our hardware using the single thread rating (STR) reported on the website: `https://www.cpubenchmark.net/` in Table 14 below. Column "Difference (%)" in Table 14 is computed as 100(STR - STR*)/STR*, where STR represents the single thread rating of a benchmark's hardware, and STR* represents ours.

	Reference	Difference (%)
Heuristic	This Paper (HVNS)	0.0
	Sacramento et al. (2019)	-20.9
	Rave et al. (2023)	18.1
	Kitjacharoenchai et al. (2019)	?

?: the information provided by the authors does not allow for the identification of the CPU model

Table 14: Single thread rating comparison