Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrie en Operationele Research

A comparative analysis of drone usage in parcel delivery

K.P.H. Hofstra (620775)

Supervisor:	P.J. Correia Duarte
Second assessor:	P. Bouman
Date final version:	1st July 2024

Abstract

In recent years, drones have greatly risen in popularity due to their ability to fly at high speeds with low cost. One application for drones that has recently garnered significant attention is the delivery of goods.

This paper studies a Multi-visit Vehicle Routing Problem with Drones (Mv-VRP-D), where multiple trucks assisted by multiple drones capable of multiple visits in a single flight are used to deliver packages to a set of customers. Because the problem is NP-Hard in nature, we propose an Adaptive Large Neighbourhood Search (ALNS) metaheuristic based on Sacramento et al. (2019) to find solutions to this problem. We assesses cost savings from problems incorporating multi-visit drones and multiple drones, compared to the Vehicle Routing Problem with Drones (VRP-D).

Using realistic parameters, we find considerable savings for multi-visit problems, multiple drone problems, and problems containing both. Sensitivity analysis highlights the significant role of drone capacity in cost savings, while drone endurance is less critical. Further analysis shows minimal additional savings from using a large number of drones with major savings already achievable by allowing two deliveries per drone flight.

1 Introduction

It has been over a decade since Amazon first proposed the use of Unmanned Aerial Vehicles (UAVs), more commonly known as drones, for package delivery. Since this moment, interest in these vehicles has only increased. Now, several organisations have launched delivery services that use drones, with market-leader Zipline having already done close to a million deliveries (Papandreou, 2024). Drones are particularly appealing for their ability to reach high speeds using relatively little energy. This makes them faster and more cost-effective, and they emit fewer greenhouse gases compared to traditional last-mile delivery vehicles like trucks. This combination of speed, efficiency, and environmental friendliness positions drones as a promising addition to the delivery sector, with even more advantages for delivery to difficult-to-reach locations.

Designing the hardware and software for delivery drones presents numerous challenges. In addition to these technical difficulties, tight regulations for the commercial use of drones pose significant obstacles. The Federal Aviation Administration (FAA), which oversees all civilian aviation in the United States, enforces rules such as a limit on the weight of drones and their maximum flight altitude. One of the most restrictive regulations is the requirement for drones to remain within the operator's visual line of sight. However, several companies have received FAA approval to operate small drones beyond the visual line of sight (Palmer, 2020), paving the way for more advanced delivery operations.

Although drones have several advantages over traditional last-mile delivery vehicles, there are two significant disadvantages for parcel delivery applications: their range and their weight capacity. Small drones can only operate for a short amount of time before needing to recharge, limiting their range around a launch location. Additionally, drones cannot carry much weight, making them unsuitable for delivering heavy parcels. Because of these limitations, this paper studies a problem where drones do not operate independently but cooperate with trucks, using the strengths of both vehicles to their advantage.

The first paper studying a problem involving drone-assisted delivery trucks was by Murray & Chu (2015). The paper extends the *Traveling Salesman Problem* (TSP) to include drones, resulting in the *Flying Sidekick Traveling Salesman Problem* (FSTSP). The paper presents a model for a single truck that cooperates with a drone. Both vehicles can deliver packages simultaneously, and the drone can use the truck as a base where it can drive along, get recharged, or get a new package to deliver.

Current literature often models the delivery of packages using delivery trucks as a Vehicle Routing Problem (VRP), an extension of the Traveling Salesman Problem (TSP) that involves routing multiple trucks from and to a depot to serve a set of customers with the aim of minimising costs. This paper builds on a broader version of the VRP, known as the *Vehicle Routing Problem with Drones* (VRP-D), where this fleet of trucks is assisted by drones in a similar way as in the FSTSP. Given the various implementations of drone usage in delivery scenarios, the primary contribution of this paper is to determine the benefit of having drones perform multiple deliveries without returning to a truck or depot in between and the benefit of having trucks carry multiple drone. This analysis will help clarify how changes in drone deployment impact overall operational efficiency.

Table 1: An overview of the problems discussed in the paper.

	Single-visit Drone	Multi-visit Drone
Single Drone	VRP-D	Mv-VRP-D
Multiple Drones	VRP-MD	Mv-VRP-D

The results of this analysis are crucial for businesses currently exploring drone delivery options. Understanding the benefits of deploying multiple drones can guide organisations in determining the optimal number of drones to use for efficient operations. Furthermore, on the design side of delivery drones, assessing the effectiveness of multi-visit drones plays a critical role in shaping design choices. Insights into the effect of multi-visits can influence critical design parameters such as battery life, payload capacity, and durability, ultimately affecting the overall utility and operational design of the drones. These factors combined can aid organisations to optimise their investment in drone technology, ensuring it aligns with their operational needs and maximises efficiency.

To address these problems, an Adaptive Large Neighbourhood Search (ALNS) metaheuristic is implemented. The algorithm will be used to find solutions for four problems: The Vehicle Routing Problem with Drones (VRP-D), the Multi-visit Vehicle Routing Problem with Drones (Mv-VRP-D), the Vehicle Routing Problem with Multiple Drones (VRP-MD) and the Multi-visit Vehicle Routing Problem with Multiple Drones (Mv-VRP-MD). How these problems relate to each other can be seen in Table 1.

Results show that having drones perform multiple visits, on average, decreases costs by 13% for large instances, with a majority of those savings being found by allowing two or three deliveries per flight. Having trucks carry multiple drones decreases costs by about 5%. Combining the two features together, saves 15% on average. Additionally, the drone capacity plays a significant role in cost savings, particularly for multi-visit problems, while drone endurance needs to be adjusted significantly to have an effect.

The remainder of this paper is structured as follows. Section 2 discusses related works in current literature and their relevance to this study. In Section 3, a detailed description of the problem is provided. The methodology is explained in Section 4. Numerical results are presented in Section 5. Finally, conclusions are presented in Section 6, and a discussion is provided in Section 7.

2 Literature Review

The use of UAV-assisted delivery trucks in parcel delivery was first proposed by Murray & Chu (2015), who present the *Flying Sidekick Traveling Salesman Problem* (FSTSP). This formulation adapts the traditional TSP by incorporating a drone to cooperate with a truck to perform deliveries. The primary objective of the FSTSP is to minimise the total time required to service all customers and ensure the return of both vehicles to the depot. Given that the FSTSP is a NP-Hard problem, the paper addresses the challenge of solving non-trivially-sized instances by employing a route and re-assign heuristic. This approach demonstrates significant time savings over the conventional truck-only TSP model, showing the potential of UAVs to enhance delivery systems.

A problem similar to the FSTSP, is the *Traveling Salesman Problem with Drones* (TSP-D) discussed by Agatz et al. (2018). In this problem, the assumption is made that the travel time of a drone between two locations is some factor times the travel time of the truck. This restriction makes it possible to proof the maximum possible time savings from using a drone alongside a delivery truck compared to the truck-only case. Several fast heuristics are presented that achieve close to optimal solutions. Experiments show that substantial time savings are possible compared to truck-only deliveries.

The multiple FSTSP (mFSTSP) presented by Murray & Raj (2020) expands the FSTSP by allowing a single truck to coordinate with a fleet of drones. The objective of this problem is to minimise the time required to deliver all packages and return all vehicles to the depot. Initially, the problem is formulated as a MILP; however, due to computational complexity, a three-phased heuristic approach is employed to solve non-trivially-sized instances. Their main findings include that adding drones to an existing fleet tends to have diminishing marginal returns and that time savings are greatest in instances with densely packed customers.

Building upon these concepts, the combination of multiple drones and multi-visit capabilities into a single model is discussed in the literature as well ((Poikonen & Golden, 2020), (Luo et al., 2021)). For their *Multi-visit Traveling Salesman Problem with Multi-Drones* (MTSP-MD), Luo et al. (2021) design a multi-start tabu search (MSTS) algorithm to tackle medium- to large-sized instances. Their computational results show significant cost reductions with the implementation of multi-visits, multiple drones, and increased drone capacity.

Sacramento et al. (2019) present a problem with a fleet of trucks, each assisted by a single drone. In their paper, they develop an Adaptive Large Neighbourhood Search (ALNS) meta-heuristic to find solutions for their Vehicle Routing Problem with Drones (VRP-D) for instances with up to 250 customers. This heuristic produces solutions with significant cost savings compared to the truck-only case. Their sensitivity analysis reveals that drone endurance can be reduced from the initial 30 minutes without significantly affecting the savings. Moreover, drone speed does not substantially impact the objective costs, and increasing the payload capacity beyond 5 kilograms does not result in considerable additional savings.

The paper presented by Wang et al. (2017) discusses a problem involving a fleet of trucks, each assisted by a fleet of drones. The paper performs theoretical analysis to evaluate the maximum savings that can be obtained as well as worst-case scenarios. The authors construct theorems to set bounds on these savings. If we consider a situation with m trucks, each assisted by up to two drones that are 50% faster than the trucks, they show that, in the best case, the delivery completion time for this problem can be reduced by 75% with respect to the truck-only case.

Another problem in the drone delivery literature is the *Multiple Traveling Salesman Problem with Drones* (mTSPD), presented by Kitjacharoenchai et al. (2019). This problem involves a fleet of trucks collaborating with a fleet of drones. different from the problem discussed in Wang et al. (2017), each drone can cooperate with multiple trucks and operate independently. The main objective is to minimise operational time, and solutions are generated using an Adaptive Insertion algorithm (ADI). Compared to solutions for a problem similar to the one discussed in

Sacramento et al. (2019), on average, the mTSPD produces better results.

Finally, Masmoudi et al. (2022) discuss the Vehicle Routing Problem with Drones equipped with multi-package payload compartments (VRP-D-MC). This problem involves a fleet of trucks assisted by drones capable of performing multiple deliveries and traveling between trucks. Additionally, packages have both a weight and a type, which restricts which vehicle can deliver them. The objective of the problem is to maximise profit. The paper proposes an adaptive multi-start simulated annealing (AMS-SA) metaheuristic to solve both the VRP-D-MC and the VRP-D on which the problem is based. In their experiments, the AMS-SA outperforms current state-of-the-art algorithms for the VRP-D. Furthermore, the VRP-D-MC shows an increase in profit compared to the VRP-D.

In current literature, not many papers study a problem involving a fleet of trucks assisted by multiple multi-visit drones. Among the papers discussed in this section, only Masmoudi et al. (2022) address a problem with a similar structure. However, only a small portion of their experiments focus on the comparison of problem with and wihout multi-visit drones. Additionally, no comparison is made between problems with and without multiple drones per truck. Our paper differs from Masmoudi et al. (2022) by specifically focusing on these features and modifying them to observe the effects of different restrictions and assumptions.

3 Problem Description

The problem addressed in this paper is a complex optimisation challenge that combines traditional vehicle routing with the capabilities of drones. It involves finding the most efficient routes for a fleet of capacitated drone-assisted trucks to deliver goods. The main objective is to minimise total operational costs, which in our case consist only of the fuel and energy costs of the trucks and drones, respectively.

In this problem, each delivery truck drives a route that starts and ends at the depot. These trucks have a weight limit, as well as a maximum route duration. Along its route, a truck visits customers to deliver packages. At each customer visit, a truck may choose to launch the drones it is equipped with to make deliveries. During a drone operation, which will be called a *sortie*, a drone is launched, visits customers to deliver packages, and returns to the truck at a recovery location. A recovery will always be performed at a customer that is visited later than the customer it was launched from, or at the depot. If the recovery location is the depot, the drone does not need to return to the truck and will remain at the depot. While driving from the launch location to the recovery location, the truck is allowed to make deliveries of its own as well as launch and recover other drones. A solution to the problem can be seen as a set of routes, each consisting of a truck route and a list of drones and their routes. In a solution, every customer has to be serviced exactly once, by either a truck or a drone.

A drone operation is subject to several restrictions. Firstly, a drone can perform multiple deliveries in a sortie, but the total weight of all parcels the drone carries can not exceed a drone's capacity, which is much less than the truck's capacity. Additionally, there is a limit on the total flight duration, which includes the launch, service, recovery and travel phases of the sortie. Upon launch, the drone travels directly to the customers' locations, delivers the parcels, and then returns to the recovery location without pause. When a recovery is performed, the recovered drone is able to be launched from the recovery location immediately. Each drone operates independently of all others, meaning that the launch or recovery of a drone does not impose restrictions on the launch or recovery of any other drone.

For this problem, some assumptions are made. Firstly, we assume the drones operate on the same network as the trucks. Additionally, the cost of a drone flying from A to B is a fraction α of the cost of a truck driving the same route, where α is less than 1. Another assumption is that multiple drones can be launched and recovered in parallel. The duration of multiple recoveries and launches will take the same amount of time as one recovery and one launch. The final assumption is that once the drone is recovered, a fully charged battery is available and can be replaced immediately, allowing for instant redeployment.

4 Methodology

In this section, a framework for an Adaptive Large Neighbourhood Search (ALNS) algorithm for the Multi-visit Vehicle Routing Problem with Multi-Drones (Mv-VRP-MD) is presented. The algorithm presented is based on the ALNS metaheuristic described by Sacramento et al. (2019). They formulate the VRP-D, a sub-problem of the Mv-VRP-MD, where trucks are assisted by a single drone capable of performing only one delivery per sortie. Besides the ALNS metaheuristic, Sacramento et al. (2019) provide a mathematical formulation for the VRP-D, classifying it as NP-Hard. Their attempts to solve this problem using a MIP formulation yielded results in a reasonable time frame only for instances with up to 12 customers, for which the ALNS also found optimal solutions. Since the Mv-VRP-MD is a generalisation of the VRP-D, similar results are anticipated. Consequently, a mathematical formulation will not be considered here, and the focus will be solely on the ALNS algorithm.

Since the ALNS will also be used to solve sub-problems of the Mv-VRP-D, some features of the algorithm will be removed for problems that do not require them. These parts of algorithms will be in red. As will be mentioned in the description of these algorithms, the red sections are only included in a problem with multi-visit drones.

This section is structured as follows: First, the general framework of the ALNS algorithm is defined in Section 4.1. Secondly, Section 4.2 presents the method used to find an initial solution. Finally, 4.3 and 4.4 formulate the destroy and repair methods used in the ALNS, respectively.

4.1 Adaptive Large Neighbourhood Search

An ALNS algorithm is a type of heuristic that iteratively uses destroy and repair methods to deconstruct and reconstruct a solution. A destroy method removes a number of customers from the solution, which a repair method inserts back. Using a variety of destroy and repair methods, each using different criteria for the removal and insertion of customers, provides diverse solutions, which avoids getting trapped in local minima.

The selection of a destroy and repair method each iteration is done using the *roulette wheel selection principle*. Each method has a weight that determines the probability that method is selected. Let Ω^- and Ω^+ be the set of destroy and repair method respectively, and let $\omega_{r,i}$ be the weight of method r for iteration i. The probability to select method r for iteration i is equal to $\frac{\omega_{r,i}}{\sum_{j\in\Omega^-}\omega_{j,i}}$, if r is a destroy method and equal to $\frac{\omega_{r,i}}{\sum_{j\in\Omega^+}\omega_{j,i}}$ if r is a repair method.

The weights are initialised equally, and the destroy methods remain that way. The weights of the repair methods, however, are tuned depending on their performance during the ALNS. When repair method r is selected, its weight is updated based on the performance parameter Ψ. The value Ψ takes depends on the quality of the new found solution and is given below.

$$\Psi = \begin{cases} \sigma_1 & \text{if the new solution is a global minimum,} \\ \sigma_2 & \text{if the new solution is not a global minimum, but better than the current solution,} \\ \sigma_3 & \text{if the new solution is worse than the current solution, but is accepted,} \\ \sigma_4 & \text{if the new solution is rejected.} \end{cases} \tag{1}$$

With the use of Ψ and the reaction factor $\rho \in (0,1)$, the weights are updated as follows:

$$\omega_{r,i+1} = \begin{cases} \rho\omega_{r,i} + (1-\rho)\Psi & \text{if repair method } r \text{ is selected for iteration } i, \\ \omega_{r,i} & \text{otherwise.} \end{cases} \tag{2}$$

The probability to accept a solution is dependant on its cost. Let s represent the current solution, s^t the proposed new solution, and $c(\cdot)$ the cost function. A new solution s^t is always accepted if its costs are less than or equal to the costs of s. If s^t has higher costs than s, the acceptance of s^t is determined probabilistically. The formula for the probability of acceptance is as follows:

$$e^{\frac{c(s)-c(s^t)}{T}}, \tag{3}$$

where T is the current temperature. The temperature starts at the value T_{init} and linearly decreases to zero over time. Let t^{elap} be the elapsed time, and t^{max} denote the maximum running time of the algorithm. The temperature at time t^{elap} is calculated as follows:

$$T = T_{init}(1 - \frac{t^{elap}}{t^{max}}). \tag{4}$$

The algorithm starts with a high temperature to allow acceptance of solutions with lower objective values, making it easier to explore different solutions. During the ALNS, the temperature linearly decreases towards zero, which limits acceptance to solutions that are not significantly worse, helping to refine the final solution. Based on these definitions, the pseudo-code for the ALNS algorithm is given in Algorithm 1. Additionally, The algorithms incorporates a maximum number of iterations without improvement. When there has not been a new global minimum for this number of iterations, the current solution is replaced by the best found solution and the algorithm continues.

4.2 Initial Solution

The construction of the initial solution involves three phases. In the first phase, customers are inserted into the solution using a Nearest Neighbour Algorithm. Each route begins at the depot, and as long as the route's capacity and maximum duration are not exceeded, the customer closest to the last added customer is included in the route as a truck visit. If a route reaches one of its

Algorithm 1: ALNS Algorithm

```
Input:
Initial Temperature: $T_{init}$,
Max iterations without improvement: $noImprMax$,
Time limit: $t^{max}$.
1  $s \leftarrow ConstructInitialSolution()$;
2  $s^* \leftarrow s$;
3  $noImpr \leftarrow 0$;
4  while $t^{elap} < t^{max}$ do
5      $d(\cdot) \leftarrow chooseDestroy()$;
6      $r(\cdot) \leftarrow chooseRepair()$;
7      $s^t \leftarrow r(d(s))$ ;
8      $T \leftarrow T_{init}(1 - \frac{t^{elap}}{t^{max}})$;
9      $p \leftarrow e^{\frac{c(s)-c(s^t)}{T}}$;
10     if $Unif(0,1) < p$ then
11         $s \leftarrow s^t$;
12         $noImpr \leftarrow 0$;
13     end
14     if $c(s) < c(s^*)$ then
15         $s^* \leftarrow s$;
16     else
17         $noImpr \leftarrow noImpr + 1$;
18         if $noImpr \geq noImprMax$ then
19             $s \leftarrow s^*$;
20             $noImpr \leftarrow 0$;
21         end
22     end
23     $updateWeights()$;
24 end
25 return $s^*$;
```

limits and there are still customers left unassigned, a new route is initiated.

In the second phase, each route is enhanced through an iterated local search using three methods: relocate moves, exchange moves, and 2-opt moves. For each method, any move that reduces the objective value is executed until no further cost-reducing move can be found.

In the third and final phase, customers are inserted into the solution as drone customers. Define D as the set of all customers with a demand smaller than the capacity of the drone, Q^D. For each customer $n \in D$, the customer is removed from the current solution and is attempted to be reinserted using the $FindSortie(n, s, \eta)$ algorithm, described in Algorithm 2. The algorithm returns the sortie containing n that decreases the costs the most, and $\emptyset$ if no sortie can be found, that results in a solution with a cost lower than η. Both the construction of a new sortie and the insertion into existing sorties are considered. Combining all three phases, the pseudo-code of the construction of the initial solution is shown in Algorithm 3. In the algorithm, $AttemptDroneInsertions(s, N')$ uses Algorithm 2 to attempt the reinsertion of a subset of customers N' into solution s as drone customers if the reinsertion results in a cost decrease.

Algorithm 2: $FindSortie(n, s, \eta)$ function for finding the best insertion for a customer, using a drone, with respect to some threshold cost η. The red section (Lines 11-19) is only included in a problem with multi-visit drones.

```
    Input:
    Customer to insert as drone-visit: $n$,
    Solution without customer: $s$,
    threshold: $\eta$,
    maximum number of deliveries in a sortie: $maxDeliveries$
1   $BestSortie \leftarrow \emptyset$;
2   for each Route in s do
3       if $Capacity(Route) + q_n < Q^T$ then
4           for available Launch l and compatible Recovery r do
5               Construct sortie $p \leftarrow (l, n, r)$;
6               if $Duration(p) < e$ AND $c(s) + Cost(p) < \eta$ then
7                   $BestSortie \leftarrow p$ ;
8                   $\eta \leftarrow c(s) + Cost(p)$;
9               end
10          end
11          for each Sortie in Route do
12              if $Capacity(Sortie) + q_n < Q^D$ AND $Deliveries(Sortie) < maxDeliveries$
                  then
13                  Construct sortie $p$ as the best insertion of customer $n$ in $Sortie$;
14                  if $Duration(p) < e$ AND $c(s) + Cost(p) - Cost(Sortie) < \eta$ then
15                      $BestSortie \leftarrow p$;
16                      $\eta \leftarrow c(s) + sortieCost(p)$;
17                  end
18              end
19          end
20      end
21  end
22  return $BestSortie$;
```

Algorithm 3: $ConstructInitialSolution()$ algorithm, used to find a starting point for the ALNS.

```
1   $s \leftarrow NearestNeighbour()$;
2   for each Route in $s_{init}$ do
3       while cost-decreasing move is found do
4           $RelocateMoves(s)$;
5           $ExchangeMoves(s)$;
6           $2OptMoves(s)$;
7       end
8   end
9   Construct D as the set of all customers with $q_n < Q^D$;
10  while cost-decreasing move is found do
11      $AttempBestDroneInsertions(s, D)$;
12  end
13  return $s$;
```

4.3 Destroy methods

In every iteration of the ALNS, the current solution is first deconstructed until at least β customers are removed from the solution. For a solution with $|N|$ customers, removal factor δ, lower bound n_{min} and upper bound n_{max}, the number of removed customers $\beta = min(max(n_{min}, \delta|N|), n_{max})$. The parameter n_{min} is chosen to be a random number for each iterations to find varying solutions for small instances. If β is too low, it might not be possible to escape local minima, if β is too high, delicate improvements might not be found. There are two destroy methods, which are defined in the subsections below.

4.3.1 Random destroy method

The first destroy method randomly selects customers from the solution until β customers are removed. When a customer is selected, we first check if that customer is being serviced by a truck or a drone. If it is a truck and the location does not serve as a launch or recovery, only that customer is removed. If the location does serve as a launch and/or recovery for a drone sortie, the sortie(s) are removed from the solution along with the selected customer. If it is a drone, we remove only the customer from the solution. If this customer was the only delivery within its sortie, the entire sortie is removed. It is possible that more than β customers are removed if a removal corresponds to a truck visit that is a launch and/or recovery point for one or more sorties.

4.3.2 Cluster destroy method

The second destroy method concentrates removal around a randomly selected focal customer, n^F. The customer is removed from the solution in the same way as described in Section 4.3.1 and further removals are selected from customers around n^F. A new removal is randomly selected from the two customers closest to n^F, until at least β customers are removed.

4.4 Repair methods

After a destroy method is applied to a solution, a repair method reinserts the set of removed customers, D, back into the solution. Some methods aim to find the best solution, while others focus on diversification. Each repair method ensures that the resulting solution is feasible, making it a candidate for the best solution. As previously mentioned, the repair methods have weights that are adjusted based on their performance during the algorithm, which constitutes the adaptive part of the ALNS. This dynamic adjustment of weights ensures that more effective repair methods are selected more frequently than those with lower performance. In the following subsections, the four repair methods used in the ALNS are described.

4.4.1 Greedy repair method

The first repair method is the greedy repair method, which has three stages. In the initial stage, all customers in D are reintroduced into the partial solution through the $TruckBestInsertion(n, s)$ algorithm, applied in a random order. This algorithm places customer n into the partial solution

s as a truck visit in a way that minimally increases costs, considering both existing routes and the possibility of opening a new route to ensure a feasible insertion can be found.

The second phase involves drone delivery insertions. Initially, attempts are made to convert current truck deliveries into drone deliveries. Subsequently, the same is done for existing drone deliveries, to increase their efficiency.

Finally, a local search algorithm is executed with a probability of p^{LS}. This probability is calculated using a temperature T^{LS} in a manner similar to the acceptance probability in the ALNS algorithm. However, unlike the temperature there, which decreases over time, T^{LS} increases. This dynamic temperature ensures that the local search is primarily employed when the repaired solution has a cost very close to or better than the cost of the current solution at the start of the ALNS and is used more frequently as the time limit approaches. The local search algorithm swaps drone deliveries between sorties if the exchange results in a cost reduction and continues to do so until no further cost-decreasing exchanges are identified. Moreover, for every launch and recovery location, there is an attempt to relocate it to an alternative location that reduces the total costs. The exact formula that is used to calculate p^{LS}, along with the pseudo-code of the repair method, can be found in Algorithm 4.

Algorithm 4: Pseudo-code for the repair method $GreedyRepair(s)$, where s is a partial solution. The red section (Lines 11-15) is only included in a problem with multi-visit drones

```
   Input:
   Partial Solution: $s$,
   time limit: $t^{max}$
   cost of last accepted solution: $c_{curr}$
1  Construct $D$ as that set of all customers not yet in $s$;
2  while $D \neq \emptyset$ do
3      $n \leftarrow RandomCustomer(D)$;
4      $D \leftarrow D \setminus \{n\}$;
5      $TruckBestInsertion(n, s)$;
6  end
7  Construct $N^T$ as the set of all truck deliveries with $q_n < Q^D$;
8  $AttemptBestDroneInsertions(s, N^T)$;
9  Construct $N^D$ as the set of all drone deliveries;
10 $AttemptBestDroneInsertions(s, N^D)$;
11 $T^{LS} \leftarrow \gamma c_{curr}(\frac{t^{elap}}{t^{max}})$;
12 $p^{LS} \leftarrow e^{\frac{c_{curr}-c(s)}{T^{LS}}}$;
13 if $Unif(0,1) < p^{LS}$ then
14     $LocalSearchSorties(s)$;
15 end
16 return $s$;
```

4.4.2 Nearby-area repair method

The second repair method has a two-phased structure similar to the first two phases of the repair method in Section 4.4.1, but uses different insertion criteria. In the first phase, the removed customers D get randomly inserted as a truck visit in a feasible position in a ζ-mile range. If no feasible position is found, a new route is opened for the inserted customer. Furthermore, in the second phase, the drone insertion of truck customers is done by randomly selecting from the set of feasible sorties that do not increase the cost of the solution by more than 10%. Because the criteria for (re)insertion are relatively flexible, this repair method is unlikely to be extensively used in instances containing a large number of customers, since the insertions will most likely lead to high costs. However, its primary purpose is to generate variable solutions for smaller instances.

4.4.3 Closest insertion repair method

The closest insertion repair method attempts to insert the removed customers D into one specific route, namely, the route the customer closest to $n \in D$ is in. This is done using the $AttemptBestInsertion(n, r)$ function, which attempts to find the best insertion for the customer n into route r considering both the use of trucks and drones. If no feasible insertion is found, the customer is added to the set N_G. Any customers in N_G are then inserted into the solution using the greedy repair method, as described in Algorithm 4.

4.4.4 Heavy insertion repair method

The final repair method, which is the heavy insertion repair method, adopts a heavy-first policy. The method removes all customers from D which are too heavy to be used in a drone sortie and adds them to a new set D^H. The customers from D^H are then inserted into the solution using the $TruckBestInsertion(n, s)$ function in a random order. The customers remaining in D are inserted into the solution using the closest insertion repair method detailed in Section 4.4.3. This approach ensures that customers who cannot be serviced by drones are efficiently inserted into the truck route.

5 Results

This section presents the results for the ALNS metaheuristic discussed in Section 4 applied to four problems: The Vehicle Routing Problem with Drones (VRP-D), the Multi-visit Vehicle Routing Problem with Drones (Mv-VRP-D), the Vehicle Routing Problem with Multiple Drones (VRP-MD) and the Multi-visit Vehicle Routing Problem with Multiple Drones (Mv-VRP-MD). First, the parameter values used throughout this section are provided in Section 5.1. The instances used for the experiments are defined in Section 5.2. General experiments are conducted in Section 5.3, and sensitivity analyses for drone features are performed in Section 5.4. Finally, Section 5.5 examines and evaluates the use of two drones for multiple drone problems and the allowance of an unlimited number of visits for multi-visit problems.

All code was written in Java using IntelliJ IDEA 2021.3.4 (Edu). The code was executed using an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz, 2803 MHz, 4 core's, 8 logical processors.

Table 2: A summary of the parameters used for the implementation of the ALNS metaheuristic.

Parameter	**Notation**	**Value**
Truck capacity	Q^T	$[1400 - 100 * drones]$ kg
Drone Capacity	Q^D	5 kg
Maximum Route Duration	h	8 h
Drone Endurance	e	30 min
Truck Service Time	ST^T	2 min
Drone Service Time	ST^D	1 min
Drone Launch Time	DL	1 min
Drone Recovery Time	DR	1 min
ALNS Initial Temperature factor	T^*_{init}	0.004
Reaction Factor	ρ	0.9
Removal Factor	δ	0.15
Local Search Sortie Scale Factor	γ	0.03
Nearby Range	ζ	5
Minimum Customers Removed	n_{min}	$Random\{1,2,3\}$
Maximum Customers Removed	n_{max}	40
Fuel Price	fp	1.13 €/L
Fuel Consumption	fc	0.07 L/km
Miles Converter	mc	1.61 km/mi
Drone Cost Factor	α	0.10
Maximum running time	t^{max}	5 min

5.1 Parameters

In this subsection, the values of the parameters defined in section 4 and some additional parameters are given. Table 2 summarises most parameter values. Unless mentioned otherwise, all parameters are set to the same values used by Sacramento et al. (2019).

Due to the algorithm's time-consuming nature, we set the local search sortie scale factor, so that roughly 5-10% of local searches are accepted for larger instances. This adjustment helps prevent the ALNS from spending most of its running time on this algorithm.

As described in Section 3, the cost of a solution consists solely by the fuel and energy costs for trucks and drones, respectively. Let d_{ij} represent the Euclidean distance between two locations i and j, measured in miles. The cost of driving from i to j with a truck is given by $c^T_{ij} = fp \cdot fc \cdot mc \cdot d_{ij}$. Conversely, the cost of covering the same route with a drone, c^D_{ij}, is defined as follows: $c^D_{ij} = \alpha \cdot c^T_{ij}$.

Next, we present and justify the values of additional parameters used in the ALNS. According to Sacramento et al. (2019), the initial temperature for the ALNS, T_{init}, is determined by multiplying T^*_{init} by the cost of the initial solution and adding 10% to this temperature for small instances. However, our experiments with this temperature showed acceptance rates significantly lower than those reported by Sacramento et al. (2019). Consequently, for an instance with a set of customers N and an initial solution s_{init}, we define T_{init} as follows: $T_{init} = T^*_{init} \cdot c(s_{init}) \cdot 2 \ln\left(e^{4\frac{200}{|N|}}\right)$

The results of using both initial temperatures are shown in Figure 1. While our temperature still results in the acceptance of solutions with lower costs, it produces acceptance rates closer

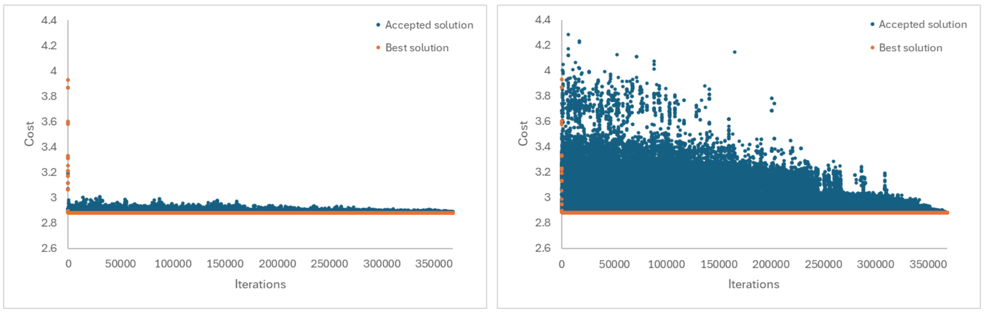

Figure 1: Cost of the accepted and best solution in each iteration for the temperature suggested by Sacramento et al. (2019) (left) and the temperature set by this paper (right). These results are obtained by applying ALNS for the VRP-D for 10 seconds to instance 12.10.3.

to the desired outcomes than the temperature proposed by Sacramento et al. (2019).

Furthermore, all repair method weights are initialised to 1, and the performance parameters σ_i are defined as follows: $\sigma_1 = 33$, $\sigma_2 = 13$, $\sigma_3 = 9$, $\sigma_4 = 0$. This setup ensures that repair methods producing desirable outcomes will have their weights adjusted upwards. The maximum number of iterations without improvement $noImprMax = 1000$. Given the greedy nature of the methods used in the ALNS, this value is kept relatively low to avoid spending much time exploring worse solutions.

5.2 Instances

The heuristics described in Section 4 are tested on instances of varying sizes. The instances used are the same as discussed in Sacramento et al. (2019). They contain between 6 and 200 customers, on a grid of size $2d \times 2d$ where d varies between 2.5 and 20 miles. The name of the instances will be given as $n.m.i$, where n is the number of customers, m is the dimension of the grid and i is the generic name of the instance.

The depot is always generated in the middle of the grid, at $(0, 0)$. The customers' x- and y-coordinates are both drawn from a uniform distribution $U(-d, d)$. The parcel weight of the customer's packages, further referred to as their demands, is drawn from the same distribution as in Sacramento et al. (2019). They state that 86% of the packages they deliver are less than 2.27 kilograms. Furthermore, the maximum accepted package weight is assumed to be 68 kilograms. Therefore, let q_n be the demand of customer n, and let $p \sim U(0, 1)$, then,

$$q_n(p) = \begin{cases} q_n \in U(0, 2.27) & \text{if } p < 0.86, \\ q_n \in U(2.27, 68) & \text{otherwise.} \end{cases} \tag{5}$$

5.3 Computational results

In the following section, the results obtained for the VRP-D, the Mv-VRP-D, the VRP-MD, and the Mv-VRP-MD using the ALNS metaheuristic formulated in Section 4 and the parameters described in Section 5.1 will be studied. For each instance, the ALNS metaheuristic is applied three times for each problem, and the best objective values are kept. The performance of the ALNS algorithm is evaluated based on the SMV, the SMD and the SMV-MD which are the the savings achieved over the VRP-D, for the Mv-VRP-D, the VRP-MD and Mv-VRP-MD

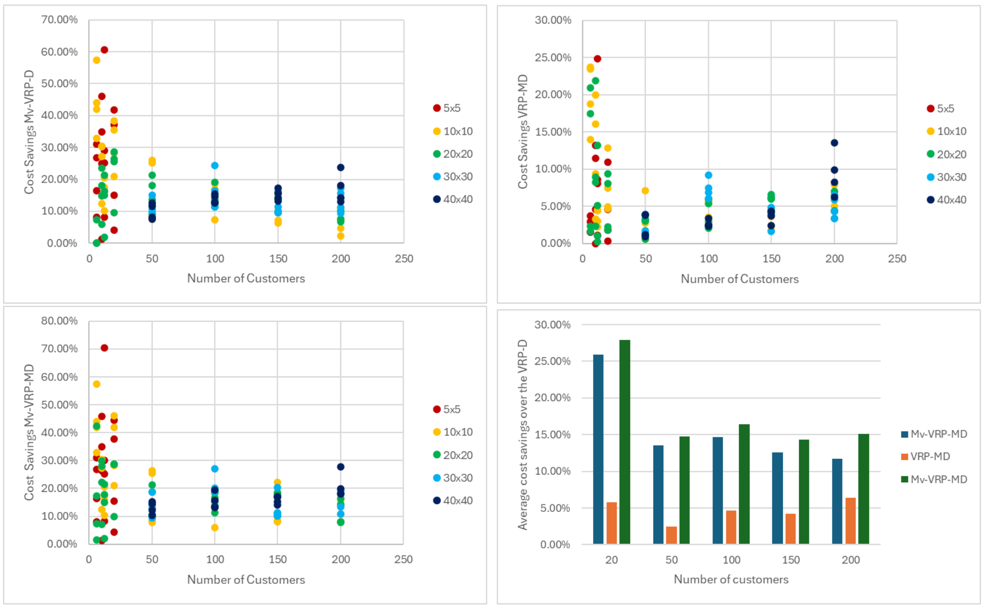

Figure 2: Savings obtained over the VRP-D for the Mv-VRP-D (top left), VRP-MD (top right) and Mv-VRP-MD (bottom left). The graph in the bottom right shows the average savings for each problem, as a function of the number of customers. All values are obtained from table 4 in Appendix A.

respectively. These savings are defined in Equation 6.

$$SMV = 1 - \frac{z^{Mv-VRP-D}}{z^{VRP-D}},\ SMD = 1 - \frac{z^{VRP-MD}}{z^{VRP-D}},SMV - MD = 1 - \frac{z^{Mv-VRP-MD}}{z^{VRP-D}}, \quad (6)$$

where z^i is the cost of the best solution found by the ALNS algorithm for problem i. The best-found objective values are given in Table 4 Appendix A. Figure 2 gives an overview of the obtained savings and Figure 3 gives a visualisation of the obtained solutions, for instance 100.20.2, for each of the four problems.

Figure 2 (top left) shows the obtained savings for the Mv-VRP-D. We observe the largest variation in savings in the smallest instances. This is expected because the placement of a few customers can significantly influence the cost of a solution, while larger instances consistently have both easy-to-service and hard-to-service customers. Most instances have savings above 10%, although some instances with a large number of customers in a small grid show smaller improvements.

We anticipated that the multi-visit problem would find considerably more savings for instances with tightly packed customers due to the opportunity to visit many customers in a single sortie, which is not the case. This can be attributed to the fact that the drone's endurance is not the limiting factor for multi-visit sorties; rather, the drone's capacity is. The analysis to support this claim will be presented in Section 5.4.

Figure 2 (top right) shows the savings obtained for the VRP-MD. Similar to the Mv-VRP-D, smaller instances exhibit a larger variation in improvements. However, the savings for the VRP-MD are considerably lower compared to the Mv-VRP-D. Additionally, savings on average seem to increase for instances with 200 customers.

Figure 2 (bottom left) presents the savings for the Mv-VRP-MD. The most notable observation is that the savings do not significantly exceed those obtained for the Mv-VRP-D.

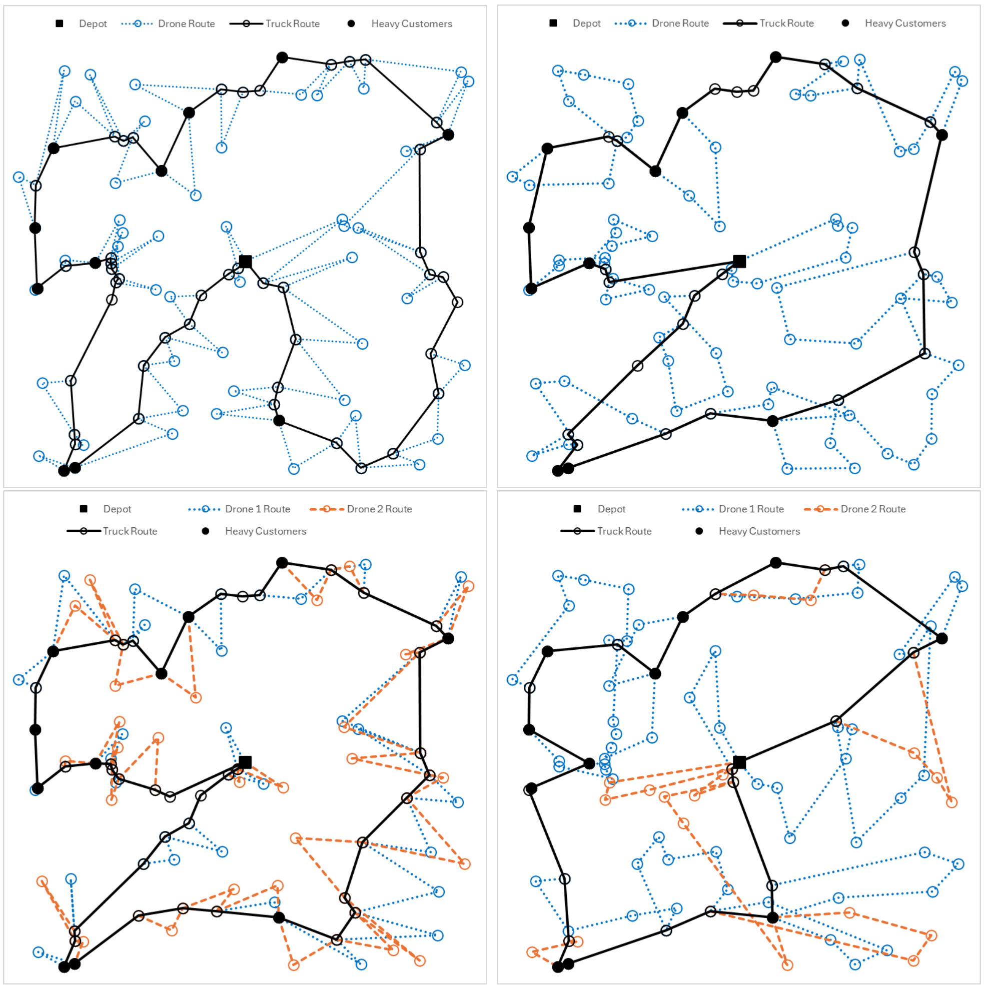

Figure 3: Best known solution for an instance with 100 customers for the VRP-D (top left), Mv-VRP-D (top right), VRP-MD (bottom left), Mv-VRP-MD (bottom right)

Additionally, the combination of multiple drones and multi-visit drones does not outperform the sum of the savings of the two individual strategies.

Figure 2 (bottom right) provides a side-by-side comparison of the average savings for all problems, as a function of the number of customers in an instance.

5.4 Sensitivity analysis

For the main analysis of this paper, we used drone parameters considered to be realistic. However, since there is currently no real-world experimentation of parcel delivery by drone-assisted trucks, the values of these parameters remain uncertain. Therefore, it is important to study their effects. Section 5.4.1 examines the influence of changing the drone endurance parameter, while Section 5.4.2 investigates the impact of varying the capacity of drones.

The analysis in this section was performed by changing only the value of the parameter stated, while keeping all other parameters set to the values described in Section 5.1. Results are obtained by testing on all instances 100.30.X, 150.30.X, and 200.30.X. The runtime and number of iterations are the same as those in Section 5.3.

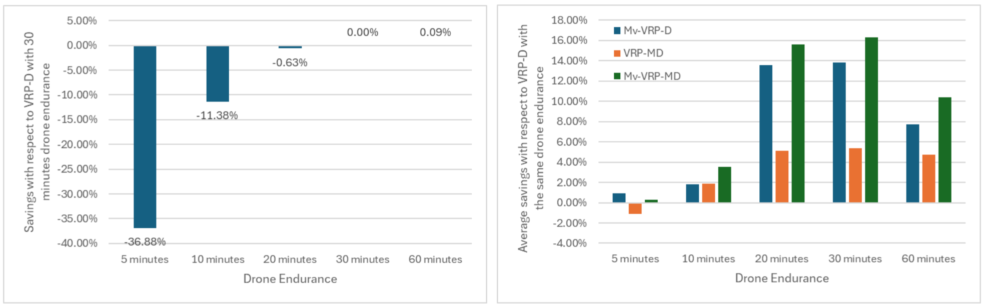

Figure 4: On the left, the cost savings for various drone endurances for the VRP-D with respect to the same problem with an endurance of 30 minutes are given. The right graph shows the savings over the VRP-D for the three other problems as a function of drone endurance.

5.4.1 Drone endurance

Figure 4 (left), shows the average savings for the VRP-D using various drone endurances, with respect to the VRP-D with an endurance of 30 minutes. Greater endurance does not significantly impact the cost of the solution, nor does an adjustment to 20 minutes. However, further reduction leads to a notable increase in costs.

The right graph in Figure 4 presents the average savings for the three other methods as a function of drone endurance. For all three methods, no significant savings are obtained with a drone endurance of 5 minutes. This outcome is expected, as only 2 minutes remain for flight if we account for the launch, service, and recovery times. Constructing a sortie with more than 2 visits becomes impossible, and a sortie with exactly 2 visits would only leave room for 1 minute of flight. Additionally, since so few sorties can be constructed, adding an extra drone does not improve the cost of a solution either.

Average savings increase for 10 and 20 minutes of endurance and stagnate at 30 minutes. This is in line with expectations, as additional drone endurance allows for more feasible sorties. However, both for multi-visit drones and multiple drones, increasing the endurance from 20 to 30 minutes does not lead to more efficient routing by the ALNS.

Increasing the endurance further to 60 minutes greatly decreases the savings obtained for multi-visit problems, while keeping the costs for the VRP-MD relatively stable. Since any solution for a problem with an endurance of 30 minutes is also feasible for an endurance of 60 minutes, the results for multi-visit problems are unexpected. The most reasonable explanation is that the ALNS does not perform as well in finding suitable multi-visit sorties when the drone endurance is greater than necessary. These findings could also explain why savings are not greater for instances with densely packed customers. For these instances, an endurance of 30 minutes could have the same effect as an endurance of 60 minutes has on the instances used in this subsection.

5.4.2 Drone capacity

Figure 5 shows the average savings obtained for the VRP-D, using different drone capacities, relative to the same problem with a drone capacity of 5 kilograms on the left. The average savings obtained for the other problems over the VRP-D are shown in the graph on the right. We observe that lowering the capacity to 2.5 kilograms and raising the capacity to 10 kilograms

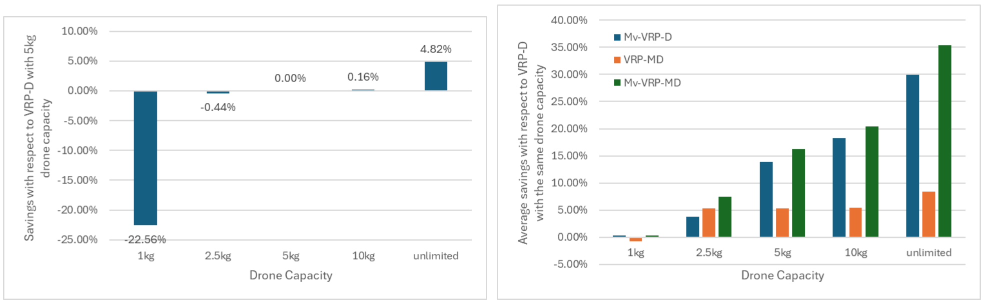

Figure 5: On the left, the cost savings for various drone capacities for the VRP-D with respect to the same problem with a capacity of 5 kilograms are given. The right graph shows the savings over the VRP-D for the three other problems as a function of drone capacity.

does not significantly change the costs for the VRP-D. A drone capacity of 1 kilogram increases costs by over 20%, while an unlimited capacity reduces costs by about 5%. This can be primarily attributed to the way customer demand is constructed. On average, 86.0% of customers have a demand of less than 2.5 kilograms, and 87.6% have a demand of less than 10 kilograms. For an instance with 100 customers, this would mean only 1 or 2 additional customers that can be serviced by a drone.

The pattern observed in the right graph of Figure 5 aligns with our expectations as well. Since relatively few customers have a demand less than 1 kilogram, multiple drones can not be effectively utilised using this drone capacity. Additionally, an even smaller number of customers have summed demands less than 1 kilogram, making multi-visit sorties not advantageous either.

For greater capacities, savings for multi-visit problems steadily increase, as expected, due to the ability to combine more customers in a single sortie. The average cost for VRP-MDs remains roughly the same for capacities of 2.5, 5, and 10 kilograms, for the same reasons given for the VRP-D. Similarly, further cost savings are observed when the drone capacity is set to unlimited.

5.5 Analysis of multiple drones and multi-visits

During the experiments in Section 5.3, the number of drones in multiple drone problems and the limit on the number of deliveries for a multi-visit drone were fixed. In this subsection, these choices are motivated and adjusted to observe their influence. The runtime and number of iterations are the same as those in Section 5.3. Results were obtained from experiments on the instances in sec:appendixA named XX.XX.1 with 20 customers or above.

5.5.1 Number of drones per truck

Throughout the experiments, in problems involving trucks supported by multiple drones, only two drones were utilised. This decision was based on preliminary findings indicating that deploying more than two drones seldom results in significant improvements. To demonstrate this effect, the results for configurations involving between 2 and 5 drones are displayed in Figure 6. As can be seen in the figure, beyond instances with more than 50 customers, there is no clear increase in savings when we allow more than two drones per truck. This paper considers two possible explanations for this result.

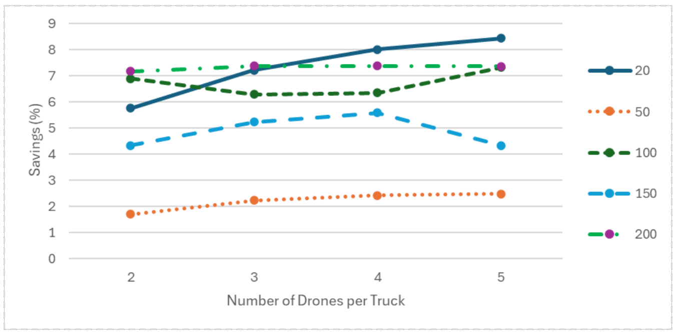

Figure 6: Average cost savings of the VRP-MD with respect to the VRP-D for different number of drone per truck, grouped by number of customers.

Firstly, because the drones add weight to the truck, the capacity constraint can be more restrictive for multiple drone problems. For the size of the simulations, in cases with only a single drone per truck, the demand constraint rarely makes an impact on the solution. Using Monte Carlo simulation[1] we can approximate the probability that the total weight of all packages exceeds the capacity of the truck. For a set of customers N, that is, the probability $P[\sum_{i\in N} q_i > Q^T]$. For some values of $|N|$ and Q^T, the results of the Monte Carlo simulations are given in Table 3

Table 3: The estimated probability the total demand of all customers exceeds Q^T for various number of customers (x-axis) and drones (y-axis) (%).

	50	100	150	200
1	$\ll$0.001	<0.001	1.080	26.085
2	$\ll$0.001	0.007	3.721	44.343
3	$\ll$0.001	0.055	10.47347	64.615
4	$\ll$0.001	0.352	24.029	81.897
5	<0.001	1.823	44.699	92.969

If the total demand of all customers does not exceed the capacity of the truck, the constraint would put no restriction on the solution, since all customer could theoretically be serviced by the same truck. Table 3 shows that indeed the constraint rarely comes into play for scenarios with less than 200 customers in single drone problems. As truck carry more drones it happens more frequently.

The second explanation for this result, has to do with the computational complexity of the problem. During experiments, the number of iterations in the allowed 5 minutes decreases as more drones are added. This is mainly due to the fact, that additional drones significantly increase the number of feasible launch and compatible recovery locations for sorties. The decrease in iterations could lead to less optimised solutions.

5.5.2 Number of drone deliveries per sortie

In Section 5.3, cost savings were achieved by permitting drones to perform multiple deliveries within a single sortie, constrained only by their capacity Q^D and maximum flying time e. However, in practical applications, delivery drones would need to be specifically engineered

[1]Monte Carlo simulation is a method that approximates a statistical value by repeated random sampling. In our case, this means sampling the weight of $|N|$ customers and checking if the sum of their weights is more than Q^D, approximating the probability that Q^D is exceeded by the demand of $|N|$ customers.

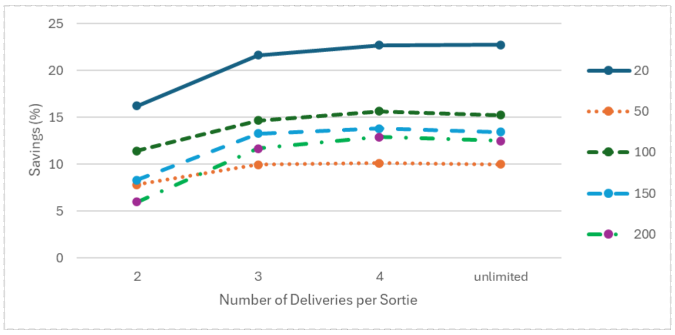

Figure 7: Average cost savings for various limits of deliveries in a sortie, grouped by the number of customers.

to handle such multi-visit deliveries. This section presents results for limiting the number of deliveries a drone can make per sortie. Cost comparisons between these restricted multi-visit deliveries and single-visit drone operations are presented in Figure 7.

The graph shows that major savings are obtained from allowing 2 visits per sortie for all instance sizes, and almost all savings are achieved with 3 visits per sortie. In some instances with a large number of customers, the ALNS finds lower costs with 3 visits per sortie compared to the costs using unlimited visits. This suggests that the ALNS might be utilising unnecessarily long sorties, and shorter sorties could potentially provide greater savings.

5.6 Performance on clustered instances

For all experiments in this section, instances with uniformly distributed customers on a grid were used. In reality, it is more realistic for customers to be clustered, similar to neighbourhoods in a city. Therefore, this subsection presents the results of the ALNS heuristic on the clustered instances from Sacramento et al. (2019). The clusters were constructed by generating θ focal points. Each customer is then assigned to one of these focal points, and their location relative to the focal point is determined using a normal distribution with a standard deviation of 2 miles.

The ALNS was applied 5 times for each problem and for each instance. The results are given in Table 5 in Appendix B.

The experiments show similar performance to instances with uniformly distributed customers. Average savings deviate at most 1.5 percentage points for all problems, compared to the instances 150.30.X. Additionally, there is no evidence that the number of clusters influences the obtained savings for any problem.

6 Conclusion

In this paper, we study the Multi-visit Vehicle Routing Problem with Drones (Mv-VRP-D) and several of its sub-problems. The Mv-VRP-D is a problem in operations research involving parcel delivery by multiple trucks, each assisted by multiple drones capable of performing multiple deliveries in a single flight, known as a *sortie*. Given the NP-hard nature of the problem, we propose an Adaptive Large Neighbourhood Search (ALNS) metaheuristic based on the algorithm presented by Sacramento et al. (2019), to find solutions. The main objective of this paper is to assess the cost savings achievable by allowing drones to perform multiple deliveries and having multiple drones cooperate with each truck.

Using parameters considered realistic, we find that for instances with between 100 and 200 customers, the average obtained savings are 13% for the multi-visit problems, 5% for those involving multiple drones, and 15% for the combination of both. Sensitivity analysis shows that drone capacity plays a major role in cost savings for multi-visit problems, while drone endurance is less influential. Throughout the experiments, two drones were used for problems including multiple drones, and analysis on the number of drones indicates that adding more drones does not significantly increase savings for instances containing a large number of customers. In the same experiments, there was no limit on the number of deliveries per sortie, but further analysis shows that the major savings can be achieved with two visits per sortie, and almost all savings are realized with three visits per sortie.

7 Discussion

In this section, we critically assess the results and conclusions from this paper. Additionally, some suggestions for future research are given.

Firstly, most results are obtained by running the ALNS metaheuristic for three iterations. While more iterations could provide more reliable results, the time constraints of this paper did not allow for it. To mitigate this, conclusions are based on averaging results for greater reliability. Additionally, the use of constant drone endurance and cost per kilometer is not realistic. This assumption may overestimate the savings for multi-visit problems because it does not account for the increased weight of drones carrying more parcels. Similarly, the sensitivity analysis on drone capacity might present a more optimistic view of savings when increasing capacity. Lastly, additional parameter tuning can be performed. Currently, the ALNS for all problems is executed using parameters similar to those in Sacramento et al. (2019). However, different parameters could be optimized for the other problems, particularly for multi-visit applications.

For future research, a more realistic energy consumption model could be used instead of constant endurance to find more accurate costs, particularly for multi-visit scenarios. Existing drone literature employs energy drain functions that base energy consumption on the speed and weight of a drone (Luo et al., 2021; Masmoudi et al., 2022; Murray & Raj, 2020). Another suggestion is to adjust the number of drones in a truck dynamically while solving the problem. Allowing the algorithm to determine this while creating solutions, would eliminate the need for an assumption on the number of drones. Finally, exploring an adaptive approach for the number of visits in a sortie could be beneficial. In some of our experiments, the ALNS found better solutions when limiting the number of visits in a sortie to three instead of having no limit. A promising implementation would be to start with a relatively low number of visits in a sortie and gradually increase this value over time.

References

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

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

Luo, Z., Poon, M., Zhang, Z., Liu, Z. & Lim, A. (2021). The multi-visit traveling salesman problem with multi-drones. *Transportation Research Part C: Emerging Technologies*, *128*, 103172.

Masmoudi, M. A., Mancini, S., Baldacci, R. & Kuo, Y.-H. (2022). Vehicle routing problems with drones equipped with multi-package payload compartments. *Transportation Research Part E: Logistics and Transportation Review*, *164*, 102757.

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

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

Palmer, A. (2020). *Amazon wins faa approval for prime air drone delivery fleet.* Retrieved 13-05-2024, from `https://www.cnbc.com/2020/08/31/amazon-prime-now-drone-delivery-fleet-gets-faa-approval.html`

Papandreou, T. (2024). *Beyond hype: Drone delivery takes flight in 2024.* Retrieved 13th of May 2024, from `https://www.forbes.com/sites/timothypapandreou/2024/03/13/beyond-hype-drone-delivery-takes-flight-in-2024/?sh=7ed069245bb9`

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

Sacramento, D., Pisinger, D. & 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.

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

A Table of main results

Table 4: Best-found objective values for all instances from Sacramento et al. (2019), using the ALNS metaheuristic for the VRP-D, Mv-VRP-D, VRP-MD and Mv-VRP-MD. Additionally, the obtained savings over the VRP-D are given for the three other problems.

Instance	z^{VRP-D}	$z^{Mv-VRP-D}$	z^{VRP-MD}	$z^{Mv-VRP-MD}$	**SMV**	**SMD**	**SMV-MD**
6.5.1	1.0982	1.0086	1.0813	1.0086	8.16%	1.54%	8.16%
6.5.2	0.8422	0.6165	0.8176	0.6165	26.80%	2.92%	26.80%
6.5.3	1.2114	0.8362	1.1931	0.8362	30.97%	1.51%	30.97%
6.5.4	0.9460	0.7904	0.9105	0.7904	16.45%	3.75%	16.45%
6.10.1	2.4061	1.6145	1.8362	1.6145	32.90%	23.69%	32.90%
6.10.2	1.6793	0.9416	1.2847	0.9416	43.93%	23.50%	43.93%
6.10.3	1.3255	0.5650	1.0771	0.5650	57.38%	18.74%	57.38%
6.10.4	1.4431	0.8369	1.2419	0.8369	42.01%	13.94%	42.01%
6.20.1	2.6776	2.6776	2.1180	1.5408	0.00%	20.90%	42.46%
6.20.2	4.3196	4.0064	4.2179	4.0064	7.25%	2.35%	7.25%
6.20.3	3.8247	3.8247	3.7630	3.7630	0.00%	1.62%	1.62%
6.20.4	3.6787	3.6787	3.0380	3.0380	0.00%	17.42%	17.42%
10.5.1	1.6556	1.6353	1.6556	1.6323	1.23%	0.00%	1.41%
10.5.2	1.4519	0.7848	1.2603	0.7848	45.94%	13.19%	45.94%
10.5.3	1.4736	1.1043	1.3052	1.0863	25.06%	11.43%	26.28%
10.5.4	1.2849	0.8365	1.2261	0.8365	34.90%	4.57%	34.90%
10.10.1	2.3265	1.6188	1.8613	1.6188	30.42%	20.00%	30.42%
10.10.2	3.1586	2.7636	3.0546	2.7636	12.50%	3.29%	12.50%
10.10.3	2.5527	1.8607	2.1436	1.8574	27.11%	16.03%	27.24%
10.10.4	2.5393	1.7702	2.3005	1.7702	30.29%	9.41%	30.29%
10.20.1	4.4524	4.1855	4.3472	4.1329	6.00%	2.36%	7.18%
10.20.2	6.1678	5.2499	5.6575	4.8019	14.88%	8.27%	22.15%
10.20.3	4.5463	3.7222	4.1397	3.1906	18.13%	8.94%	29.82%
10.20.4	6.1536	4.6995	4.8040	4.4238	23.63%	21.93%	28.11%
12.5.1	1.3738	1.0284	1.2634	1.0284	25.14%	8.04%	25.14%
12.5.2	1.0590	0.4178	0.7956	0.3136	60.55%	24.87%	70.39%
12.5.3	1.4477	1.3282	1.4316	1.3264	8.25%	1.11%	8.38%
12.5.4	1.5810	1.1226	1.4456	1.1057	28.99%	8.56%	30.06%
12.10.1	2.6810	2.1301	2.5608	2.1291	20.55%	4.48%	20.59%
12.10.2	2.6842	2.2483	2.5652	2.2483	16.24%	4.43%	16.24%
12.10.3	2.8805	2.5845	2.8166	2.5845	10.27%	2.22%	10.27%
12.10.4	2.3142	1.9080	2.2448	1.9053	17.55%	3.00%	17.67%
12.20.1	5.7776	4.5407	5.0175	4.5407	21.41%	13.16%	21.41%
12.20.2	8.2725	8.1112	8.2508	8.1112	1.95%	0.26%	1.95%
12.20.3	4.1669	3.5377	4.1238	3.5377	15.10%	1.03%	15.10%
12.20.4	6.0886	5.0927	5.7769	4.9998	16.36%	5.12%	17.88%
20.5.1	1.7935	1.5233	1.7610	1.5163	15.06%	1.81%	15.45%
20.5.2	1.8221	1.7457	1.8166	1.7450	4.19%	0.30%	4.23%
20.5.3	1.4866	0.8667	1.4185	0.8241	41.70%	4.58%	44.57%
20.5.4	1.3789	0.8669	1.2280	0.8593	37.13%	10.95%	37.69%
20.10.1	3.2525	2.3361	3.0109	2.3339	28.18%	7.43%	28.25%
20.10.2	3.0894	1.9902	2.6919	1.6639	35.58%	12.87%	46.14%
20.10.3	3.7023	2.2830	3.5294	2.1499	38.33%	4.67%	41.93%
20.10.4	3.1966	2.5259	3.0376	2.5259	20.98%	4.98%	20.98%
20.20.1	7.3230	5.4461	6.7346	5.2178	25.63%	8.03%	28.75%
20.20.2	7.5394	5.3847	7.3732	5.3847	28.58%	2.20%	28.58%
20.20.3	7.4610	5.4971	6.7610	5.3140	26.32%	9.38%	28.78%
20.20.4	7.0133	6.3392	6.8868	6.3156	9.61%	1.80%	9.95%
50.10.1	5.8613	5.4142	5.8247	5.4045	7.63%	0.63%	7.79%
50.10.2	5.5849	5.1206	5.4991	5.1206	8.31%	1.54%	8.31%
50.10.3	5.4161	4.0102	5.0329	4.0319	25.96%	7.08%	25.56%
50.10.4	5.1405	3.8477	4.9995	3.7895	25.15%	2.74%	26.28%

(continued on next page)

Instance	z^{VRP-D}	$z^{Mv-VRP-D}$	z^{VRP-MD}	$z^{Mv-VRP-MD}$	**SMV**	**SMD**	**SMV-MD**
50.20.1	10.2349	9.2878	10.1736	9.1944	9.25%	0.60%	10.17%
50.20.2	10.0561	8.6840	9.7439	8.6303	13.64%	3.10%	14.18%
50.20.3	10.5018	8.5990	10.1554	8.5338	18.12%	3.30%	18.74%
50.20.4	10.6641	8.3817	10.3458	8.3915	21.40%	2.99%	21.31%
50.30.1	15.7714	14.1152	15.5611	14.1152	10.50%	1.33%	10.50%
50.30.2	15.0148	13.2067	14.7591	13.0640	12.04%	1.70%	12.99%
50.30.3	16.3865	13.9102	15.7530	13.3142	15.11%	3.87%	18.75%
50.30.4	18.5973	16.7987	18.3909	16.8205	9.67%	1.11%	9.55%
50.40.1	20.0883	17.5430	19.3144	17.0101	12.67%	3.85%	15.32%
50.40.2	20.6253	18.9516	20.3786	18.4928	8.12%	1.20%	10.34%
50.40.3	22.6452	20.0285	21.7850	19.8612	11.56%	3.80%	12.29%
50.40.4	22.3371	20.6440	22.1249	19.1063	7.58%	0.95%	14.46%
100.10.1	6.9395	5.6401	6.5401	5.6666	18.73%	5.76%	18.34%
100.10.2	7.5477	6.5143	7.2819	6.3369	13.69%	3.52%	16.04%
100.10.3	7.2211	6.0090	6.8056	5.8422	16.79%	5.75%	19.09%
100.10.4	7.3656	6.8222	7.1845	6.9238	7.38%	2.46%	6.00%
100.20.1	14.0361	12.1159	13.2848	12.0960	13.68%	5.35%	13.82%
100.20.2	14.0245	12.2796	13.6091	11.6770	12.44%	2.96%	16.74%
100.20.3	13.5900	12.0394	13.3056	12.0686	11.41%	2.09%	11.20%
100.20.4	13.8588	11.1996	13.0352	11.2909	19.19%	5.94%	18.53%
100.30.1	22.4972	19.5488	20.8207	19.0543	13.11%	7.45%	15.30%
100.30.2	22.1884	18.5412	20.6709	18.0356	16.44%	6.84%	18.72%
100.30.3	23.5601	20.5243	21.7523	18.5209	12.89%	7.67%	21.39%
100.30.4	22.4013	16.9240	20.3362	16.3566	24.45%	9.22%	26.98%
100.40.1	28.8820	24.4356	27.9082	24.3691	15.40%	3.37%	15.63%
100.40.2	29.5612	25.8186	28.8585	25.6880	12.66%	2.38%	13.10%
100.40.3	28.3618	24.7343	27.6476	24.5880	12.79%	2.52%	13.31%
100.40.4	28.8516	24.5611	28.0988	23.2202	14.87%	2.61%	19.52%
150.10.1	8.6076	7.7248	8.2970	7.6808	10.26%	3.61%	10.77%
150.10.2	8.1718	7.5951	7.9727	7.5141	7.06%	2.44%	8.05%
150.10.3	8.4473	7.1005	8.0540	6.5721	15.94%	4.66%	22.20%
150.10.4	8.7842	8.2201	8.6225	8.0379	6.42%	1.84%	8.50%
150.20.1	17.2130	14.5145	16.1650	14.0462	15.68%	6.09%	18.40%
150.20.2	16.7368	15.1325	15.6400	15.0260	9.59%	6.55%	10.22%
150.20.3	17.3303	14.8286	16.2916	14.3625	14.44%	5.99%	17.12%
150.20.4	16.7161	14.3572	15.6679	13.7221	14.11%	6.27%	17.91%
150.30.1	25.5092	23.0703	24.5042	22.6989	9.56%	3.94%	11.02%
150.30.2	26.3238	23.3337	25.0368	23.3337	11.36%	4.89%	11.36%
150.30.3	25.0831	21.0142	24.0701	19.9792	16.22%	4.04%	20.35%
150.30.4	25.9407	23.3632	25.5130	23.3335	9.94%	1.65%	10.05%
150.40.1	33.2507	28.5811	32.0169	28.5811	14.04%	3.71%	14.04%
150.40.2	36.5743	31.7542	34.9963	31.0142	13.18%	4.31%	15.20%
150.40.3	35.7271	30.1642	34.8553	29.6337	15.57%	2.44%	17.06%
150.40.4	34.3110	28.3813	32.8230	28.4982	17.28%	4.34%	16.94%
200.10.1	9.8477	9.0982	9.0665	8.7889	7.61%	7.93%	10.75%
200.10.2	9.7517	9.5370	9.1685	8.1885	2.20%	5.98%	16.03%
200.10.3	9.8025	9.3394	9.3117	9.0230	4.72%	5.01%	7.95%
200.10.4	10.0382	8.9192	9.3611	8.6171	11.15%	6.75%	14.16%
200.20.1	20.8003	17.8441	19.3342	17.3703	14.21%	7.05%	16.49%
200.20.2	21.1237	19.7219	20.4086	19.4774	6.64%	3.39%	7.79%
200.20.3	20.1779	18.1415	19.2955	17.2857	10.09%	4.37%	14.33%
200.20.4	19.0809	17.6269	18.2530	17.5468	7.62%	4.34%	8.04%
200.30.1	29.9570	27.1262	28.9457	26.7190	9.45%	3.38%	10.81%
200.30.2	30.4966	25.2174	28.7089	24.5742	17.31%	5.86%	19.42%
200.30.3	30.9327	26.1044	28.9153	25.3244	15.61%	6.52%	18.13%
200.30.4	31.0803	27.6082	29.7456	26.9325	11.17%	4.29%	13.35%
200.40.1	41.5775	34.0992	37.4499	33.9551	17.99%	9.93%	18.33%
200.40.2	42.6046	37.0575	39.0837	34.1138	13.02%	8.26%	19.93%
200.40.3	42.2931	36.2851	39.6723	34.6319	14.21%	6.20%	18.11%
200.40.4	42.0798	32.0561	36.3795	30.4228	23.82%	13.55%	27.70%

B Table of results for clustered instances

Table 5: Best-found objective values for all clustered instances from Sacramento et al. (2019), using the ALNS metaheuristic for the VRP-D, Mv-VRP-D, VRP-MD and Mv-VRP-MD. Additionally, the obtained savings over the VRP-D are given for the three other problems.

Instance	θ	z^{VRP-D}	$z^{Mv-VRP-D}$	z^{VRP-MD}	$z^{Mv-VRP-MD}$	**SMV**	**SMD**	**SMV-MD**
150.30.c.01	1	7.2092	5.86335	6.7486	5.673760558	18.67%	6.39%	21.30%
150.30.c.02	4	17.576	15.5682	16.565	15.49727275	11.43%	5.75%	11.83%
150.30.c.03	5	19.132	17.469	19.006	17.45730844	8.69%	0.66%	8.75%
150.30.c.04	3	17.525	15.6075	16.793	15.30194166	10.94%	4.18%	12.69%
150.30.c.05	3	12.947	11.338	12.578	10.98491155	12.43%	2.85%	15.16%
150.30.c.06	2	11.79	10.6811	11.52	10.78469115	9.40%	2.29%	9.40%
150.30.c.07	2	10.205	9.07811	9.8146	8.867070408	11.04%	3.82%	13.11%
150.30.c.08	5	16.071	14.6082	14.339	13.42660808	9.10%	10.78%	16.46%
150.30.c.09	5	14.111	11.8816	12.089	10.85479773	15.80%	14.33%	23.08%
150.30.c.10	3	17.791	16.9897	16.902	16.66024849	4.50%	5.00%	6.36%

C Code

The code for this thesis was written in Java and is made up of 10 classes. There roll will be explained in the order that is I deemed was most logical.

Firstly, the *Node* class is what defines a customer. Every node object has coordinates and a demand. Methods in this class include, calculating the distance to other customers and checking if a node is the depot.

The *Instance* class is used to define an instance. It contains the grid size, the number of customers and the number of drones that can be utilised. Customers can be either randomly generated, or added to the instance.

Next up is the *Route* class. In this class, all features of a route are stored. It contains the route of the truck, route(s) of the drone(s) and the arrival times at all locations. Additionally, parameter values with respect to the trucks and drones are set here. All methods that only concern a single route are found here. Some methods include, the insertion of a customer, the removal of a customer, checking the feasibility of adding a customer to the route and updating all the arrival times at customers.

The *Solution* class holds all information to properly define a solution. It stores the Instance object the solution is for, and all Route objects in the solution. The class contains a method to calculate the cost of the solution and a method that returns the customers that are currently in the solution. Additionally, the insertion methods are defined in this class as well as the drone exchange algorithm, used in the local search for sorties.

The *ALNS* and *MultiVisitALNS* classes are separate classes that hold all functionality and parameters of the ALNS algorithm for the single-visit and multi-visit drones, respectively. The classes are split up, because the additional methods used in the Multi-visit problems were most practical to implement using a different class. The methods in both classes are the same, with the *MultiVisitALNS* including some additional methods. These methods include: The ALNS algorithm, the destroy and repair methods, the construction algorithm for the initial solution and the algorithm to find feasible sorties.

For three additional classes, we will mention their purpose shortly. The *FileReader* class is used to read out the instances from Sacramento et al. (2019) from text files. The *MonteCarlo* class is used to produce the Monte Carlo simulations of which the results are presented in Table 3. Lastly, the *Distributions* Class holds methods related to the sampling of values from distributions. Additionally, some methods to get statistical values are written here.

Finally, all runs are performed in the *Main* class. In this class, the methods to gather all results are written. All lines that were used to obtain the results are still in this class, as comments. The only results that can not be obtained from running any of these lines, are the results for the sensitivity analysis. To get those, the drone features (endurance and capacity) need to manually be adjusted in the *Route* class. Results can then be obtained by running the *getResults* method for any instance.