ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Bachelor Thesis FEB63007

Investigating the effect of allowing a drone to transport multiple packages simultaneously

Name student: Sandra Uljee

Student ID number: 540970

Supervisor: Y.N. Hoogendoorn

Second assessor: O. Vicil

Date final version: 03-07-2022

Abstract

Drones have become a very popular topic in logistics because of their potential to significantly reduce the cost and time of making last-mile deliveries. There are various studies which investigate the use of these flying vehicles in order to transport parcels, food or other goods from a warehouse to customers. One of these papers introduces the *Vehicle Routing Problem with Drones*, where each truck in a fleet of driver-operated vehicles is equipped with a drone. Since the drone is able to serve some of the customers on the route, the corresponding truck can save both travel time and travel costs. In this research, we investigate the effect on the total costs when drones are allowed to deliver multiple packages within one trip, resulting in the *Vehicle Routing Problem with Multi-Visit Drones*. An ALNS metaheuristic is carried out in order to explore how beneficial such an adaption to the drones could be. The perfomance of the ALNS metaheuristic is first tested by applying the algorithm to the *Vehicle Routing Problem with Drones* and comparing the results with existing literature. Our main findings indicate that letting drones carry several commodities at the same time will save the most costs in situations where not too many customers have to be served.

Contents

1 Introduction

Drones, also called Unmanned Aerial Vehicles (UAVs), are among the most strongly discussed emerging technologies nowadays. In the period 2009-2015 most drones were being operated by the government, while businesses came in second with using about 22% of all registered drones (Choi-Fitzpatrick et al., 2016). In the meantime, it is safe to say that the business utilization of drones have outpaced governmental use, where the logistics industry is currently the prime user of the flying vehicles ("The Future of Drones in Transportation and Logistics", 2022). Various companies are studying how much cost, energy and time can be saved by delivering parcels, food, medical supplies and other goods with the use of drones (French, 2015).

Drone delivery started to get a great deal of attention when Jeff Bezos, the CEO of Amazon, revealed that the company was exploring the idea of using autonomous drones to deliver small packages to their customers (Rose, 2013). A year later, DHL launched a similar project for a drone delivery service (Hern, 2014). This service exploits quadcopters to transport small commodities, which includes medications and other urgent goods, to a small island a few miles from the German coast. Last year, Walmart started delivering various products with the use of drones to customers that live within 1.5 miles from two of their stores (Lee, 2022). A few months later, Wing, Google's sister company, started offering deliveries by drones in different suburbs of Dallas (Lee, 2022). In this project, the drones start their routes in two Walgreens parking lots and transfer goods varying from health products to groceries like ice cream to customers close by. While the services of Wing and Walmart are still limited, drone delivery is not only in the research-and-development phase anymore.

Delivering commodities by means of drones has its limits due to the restricted distance, flight endurance and capacity of the flying vehicles. Nevertheless, in cooperation with driver-operated trucks, drones can reduce operational costs, delivery times and environmental impact. When drones transport small packages and therefore assist drivers in making deliveries, more customers could be served in a certain time frame without driving extra miles (Trop, 2016).

The problem of distributing commodities is in the literature usually formulated as the *Vehicle Routing Problem* (VRP). With this mathematical model, routes starting and ending at a common depot are constructed for a fleet of vehicles to traverse, in order to satisfy all customer demand while minimizing the travel costs. When considering drones as an additional delivery option, it has been proposed to supply driver-operated trucks with a drone. This drone is then able to service some of the customers on the route, using it corresponding truck to launch. As the

drone is being used, the truck continues with its tour and recovers the drone at a later point. This drone addition problem is modelled as the *Vehicle Routing Problem with Drones* (VRP-D). Applegate et al. (2011) are the first to study the collaboration of the truck and the drone and make use of the *Traveling Salesman Problem* (TSP), a variant of the VRP with only one truck.

Sacramento et al. (2019) investigated a variant of the VRP-D where each driver-operated truck is equipped with a single drone. This problem generalizes the classical VRP, which means that it is NP-hard to solve. They use an Adaptive Large Neighborhood Search (ALNS) metaheuristic to solve this multi-truck problem and investigate how much costs can be saved compared to the case where only trucks are used.

As mentioned before, a drone has several limitations, one of them being the fact that it usually can only carry a single package. The drone therefore has to return to their corresponding truck as soon as it has served one customer. However, it could be that the drone is not at its maximum capacity and that is still has battery power left, such that the flying vehicle can be used more efficiently. Since more distance could be covered within the same trip, we investigate the possibility that a drone can transport multiple packages simultaneously. In this way, several customers can be served by a drone in the same drone operation. We introduce this problem as the *Vehicle Routing Problem with Multi-Visit Drones* (VRP-MVD). We use a variant of the ALNS metaheuristic in Sacramento et al. (2019) to solve this problem. The performance of this metaheuristic is first tested by applying the algorithm to both the VRP-D and the VRP and comparing the results with the objectives found in Sacramento et al. (2019).

By allowing drones to deliver multiple packages in one run, more customers can be served by a drone instead of a truck, saving travel time and expenses. Nevertheless, the drone will need some adjustments before it is able to carry several commodities at the same time, which costs money. The central question of this paper is therefore: *When is making adjustments to a drone in order for it to carry multiple packages simultaneously beneficial?* Next to this, we answer the follow up question: *Under what circumstances are the adjustments to the drone useful?*, where the number of customers and the distance between them are considered.

The remainder of this thesis is structured as follows. In Section 2 an overview of literature is presented that is relevant to the VRP-D, the VRP-MVD and the Large Neighborhood Search. In Section 3, both the VRP-D and the VRP-MVD are discussed extensively. The ALNS metaheuristic for both models is described step for step in Section 4. In Section 5, we discuss the results of applying the metaheuristic and give an answer to our research question. Lastly, a conclusion is provided in Section 6, together with suggestions for further research.

2 Literature Review

In this section we present some relevant literature for our research. First, we discuss previously conducted studies that concern the VRP with the inclusion of drones that can deliver either a single or multiple packages in one trip. The VPP is a combinatorial optimization problem which asks "What is the optimal set of routes for a fleet of vehicles to traverse in order to deliver packages to a given set of customers?" This optimal set of routes can either be determined by the minimal travel time that the vehicles need to serve all customers or the lowest costs that are required to achieve this goal. Next to this, we show the usefulness of an ALNS metaheuristic by means of considering its success in earlier research.

2.1 Drone delivery

Murray & Chu (2015) investigated two new optimization problems related to delivering small packages by drones. In their paper, mixed integer programming formulations are given for these problems, and different heuristics are suggested. Special attention is paid to a unique variant of the classical TSP, the *Flying Sidekick Traveling Salesman Problem* (FSTSP), in which a drone works in collaboration with a delivery truck in order to transport packages. The objective of this problem is to minimize the time that it takes to serve all customers and to return both vehicles to the depot. The second problem that is considered is the *Parallel Drone Scheduling Traveling Salesman Problem* (PDSTSP), which is appropriate for scenarios in which a significant part of the customers is close to the distribution centre. In this problem, multiple drones are used to deliver goods to these customers nearby, while a single truck operates independently on the other customers. Murray & Chu (2015) show that the use of drones for last-mile delivery is expected to faster distribute orders at a lower cost and also reduce environmental impact.

Another last-mile delivery concept in which a truck works together with a drone is discussed in Agatz et al. (2018). In this paper, a new variant of the traveling salesman problem, the *Traveling Salesman Problem with Drone* (TSP-D), is modelled as an integer problem. This problem is very similar to the FSTSP, except that it assumes that the drone is faster than the truck by a factor α. Furthermore, in contrast with the FSTSP, the truck may now wait for the drone in the same position as it was launched. Several route-first, cluster-second heuristics based on local search and dynamic programming are proposed to solve the problem. After applying these heuristics to various instances with different characteristics and sizes, it is concluded that including drones can generate substantial savings compared to truck-only delivery.

A new variant of the TSP-D is considered in Ha et al. (2018). Whereas the objective of Agatz et al. (2018) was to minimize the total service time, the aim of this paper is to minimize operational costs including total transportation cost and a penalty for wasted time as a result of vehicles that have to wait for each other. The launch and recovery of a drone are restricted to different locations, similar as in Murray & Chu (2015). A mathematical model for this problem is formulated and two algorithms are proposed for obtaining the solution. The first algorithm, known as TSP-LS, converts an optimal TSP solution to a feasible TSP-D solution with the use of local searches. The second algorithm, a Greedy Randomized Adaptive Search Procedure (GRASP), is derived from a new split procedure that optimally splits a TSP tour into a TSP-D solution. After this solution is generated, it is improved by local search operators. The results show that GRASP performs better than TSP-LS in terms of the quality of the solution under an acceptable running time.

Although the existing literature related to drone delivery is mostly focused on using drones exclusively or together with a single truck, Wang et al. (2017) investigate the use of a fleet of trucks in cooperation with a series of drones. In their paper, they introduce the *Vehicle Routing Problem with Drones* (VRP-D). The objective is to minimize the completion time of the routes and the drones can be dispatched from and picked up at the distribution centre or at any of the customers. The maximum savings that can be obtained from the use of drones are studied. The results depend on the number of drones per truck and the speed of the drones in comparison to the speed of the truck.

While most of the literature concerns combined truck-drone delivery where drones are assumed to deliver a single package per trip, Luo et al. (2021) considers using trucks in combination with multiple drones that can serve several customers in one operation. The paper investigates the multi-visit traveling salesman problem with multiple drones (MTSP-MD), where the objective is to minimize the travel time that is needed to serve all customers. Luo et al. (2021) make the assumption that the energy consumption of the drone depends on the total weight of packages carried by the flying vehicle, which declines after every delivery during the flight. The problem is formulated as a mixed-integer linear program (MILP) model and is solved with the use of a multi-start tabu search algorithm. It is shown that using multiple drones with multiple visits leads to a significant cost reduction.

A similar problem was studied in Poikonen & Golden (2020). In this paper, the k-Multi-visit Drone Routing Problem (k-MVDRP) is considered, which concerns a collaboration between a single truck and k drones. Each drone is able to launch from the truck with one or more packages

to serve customers. Again, a specification of a drone energy drain function is accounted for, factoring in the weight of each package. A heuristic algorithm was used to obtain results, which showed that the objective, the minimum route completion time, is highly sensitive to drone speed and the number of drones.

2.2 Large neighborhood search

Ropke & Pisinger (2006) explore the pickup and delivery problem with time windows, which serves a variety of transportation requests with a restricted number of vehicles. The problem is to construct routes that visit all customers in a way that corresponding pickup and delivery locations are placed on the same route and that the pickup is completed before the delivery. The proposed heuristic for this problem is based on an extension of the Large Neighborhood Search heuristic presented in Shaw (1997). It consists of different competing sub heuristics which are used with a recurrence based on their previous performance. This general idea is called an Adaptive Large Neighborhood Search (ALNS). The heuristic is tested and provides an improvement to the best known solutions from the literature for more than half of the problems. Ropke & Pisinger (2006) show with computational experiments that it is favourable to use a number of competing sub heuristics instead of just one.

A different application of an ALNS heuristic can be found in Azi et al. (2014), in which a vehicle routing problem with multiple routes is investigated. In this problem, the routing of a fleet of vehicles is determined, where every vehicle is able to perform multiple routes in a day. This could be relevant when for example perishable goods are transferred. The objective is to first maximize the number of customers that are served and then to minimize the total distance traveled. The problem is solved by using an ALNS heuristic. The several destruction and reconstruction operators make use of the hierarchical nature of the problem by working either at the customer, route or workday level. Computational results show the advantages of this multi-level approach.

In their paper, Sacramento et al. (2019) use an ALNS metaheuristic in order to solve the VRP-D, in which each driver-operated truck is equipped with a drone that assists them in delivery packages. For small instances, they compare the results obtained by their metaheuristic with the optimal objectives found by the mathematical model in order to test the performance of the algorithm. Sacramento et al. (2019) ran their heuristic 10 times for each of the instances and in every run, the algorithm was able to reach the optimal solution, showing the effectiveness of the metaheuristic.

3 Problem Description

In this section, we give a detailed explanation of the VRP-MVD, where its objective and constraints are central. The VRP-MVD is a combinatorial optimization problem which is used to determine the optimal set of routes for an unlimited homogeneous fleet of vehicles, each of them equipped with a single drone, to deliver packages to a given set of customer C. The drones that are used to assist the trucks in serving customers are capable of carrying multiple packages at the same time. Each customer $i \in C$ has to be served exactly once, by either a driver-operated delivery truck or the drone that belongs to that truck. All the trucks must depart from and return to the depot, but the drones may depart or return independently from their corresponding truck to this distribution centre. During a truck route, the drone can be dispatched from the truck to deliver packages several times. The truck then continues with its route and is only allowed to pick up the drone at a location different from its launch position. Not every package can be delivered by a drone, because the drones have a limited capacity Q^D. The total weight of the packages that a drone is carrying at the beginning of its operation is therefore restricted to this capacity. Another restriction on the drone is the maximum flying endurance e, which means that launching, traveling to all customers within the operation, serving these customers and recovering to the truck must all be done before the battery runs out. At all time that a drone is not delivering a package, it is transported by the truck in order to save battery power.

The restricted capacity Q of the trucks also have to be taken into account, such that total weight of the packages and drone equipment does not exceed this capacity. Another requirement is the maximum duration of a route, T_{max}, which corresponds to the work hours of a truck driver. While respecting these capacity and time constraints and meeting customers' demand, the objective is to minimize the total cost of the operation. These costs only consist of an estimation of the fuel consumption incurred by the vehicles when traversing the routes, because the drivers' salary is considered as a fixed cost (Sacramento et al., 2019). Due to the lack of actual data, it is assumed that the travel costs of a drone are independent of the load.

Because of the restricted weight that a drone can transport, the given set of customers, C, is split into a subset of customers that may be served by a drone and a subset of customers whose demand, q_i, can only be satisfied by a truck. Next, a set P of possible three-node sorties (i, j, k) is defined from where a drone can deliver a package. The first node, i, represents the launch position, the second node, j, represents the customer that is served by the drone, and the third node, k, represents the recovery position. The customer that is visited by the drone has to be

in the subset of customers whose demand can be carried by a drone ($q_i \leq Q^D$). It must also be taken into account that the sum of the launch time, the recovery time, the total travel time and the service time does not exceed the flight endurance of the drone.

The distance d_{ij} between locations i and j is calculated as the Euclidean distance between their coordinates. With the use of the truck speed v^T and the drone speed v^D, we can derive the travel times by means of $\tau_{ij}^{T,D} = d_{ij}/v^{T,D}$. We assume that the speed of a drone is independent of the load it is carrying. Thus, the drone is not getting faster when some of the packages are already delivered. The costs for the trucks are calculated as $c_{ij}^T = fp \cdot fc \cdot d_{ij}$, where fp is the fuel price and fc the fuel consumption. Using a drone instead of a truck is considerably cheaper, so the drone cost is equal to a factor $0 < \alpha < 1$ of the trucks cost, $c_{ij}^D = \alpha \cdot c_{ij}^T$.

When a drone is only allowed to serve one customer each time it launches, such that it has to return to their truck directly after delivering a package, the problem results in the VRP-D.

4 Methodology

In this section, we first discuss the ALNS metaheuristic that Sacramento et al. (2019) uses to solve the VRP-D. In the second part of this section, we adjust this metaheuristic in order to solve the case where a drone is allowed to deliver multiple packages within one operation.

4.1 Vehicle Routing Problem with Drones

We now consider the ALNS metaheuristic of Sacramento et al. (2019) to solve the VRP-D. Large Neighborhood Search (LNS) constantly improves an initial solution by destroying and repairing the current solution over and over again. Ropke & Pisinger (2006) extended LNS by using an ALNS framework which introduces multiple destroy and repair methods that are chosen based on their performance throughout the search. In this framework, destroy methods are used to tear down part of the current solution, while repair methods restore this partial solution. Destroy methods usually contain some randomness in order to eliminate different parts of the solution and therefore vary the search for new solutions. Also the repair methods can be stochastic to refrain from constructing the same solution. The metaheuristic is highly dependent of the degree of destruction. If a too small part of the solution is eliminated, the method can experience difficulty escaping local minima. However, if the destroy method removes too much from the solution, it can be difficult for the repair method to find a good solution.

Ω^- and Ω^+ are defined as the sets of destroy and repair methods, respectively. In every iteration, a destroy method and a repair method are chosen from these sets in order to alter the current solution. This selection is done statistically, established on the assigned weights of the different methods and the use of the *roulette wheel selection principle* (Sacramento et al., 2019). These weights are thus initialized with equal probability and are changed repeatedly by using a reaction factor $\rho \in [0, 1]$ and the score Ψ of the corresponding method. The different scores can be found in Table 1. Let w_{ij} be defined as the weight of method i in iteration j, then the weights are updated as

$$w_{i,j+1} = \rho w_{ij} + \Psi(1 - \rho). \tag{1}$$

Table 1: Scores of a method.

Parameter Ψ	**Description**
σ_1	The new solution resulted in a new global best solution
σ_2	The new solution was accepted with a lower cost than the current solution
σ_3	The new solution was accepted with a higher cost than the current solution
σ_4	The new solution was rejected

As we do not want to move randomly through the solution space, it is required to oversee and accept the solutions that are constructed iteratively. For this reason, the ALNS metaheuristic is extended with an acceptance criteria taken from Simulated Annealing (Sacramento et al., 2019). A temperature parameter T is used to control the acceptance probability. If a new solution s^t has an improved objective value with respect to the current solution s, then s^t is always accepted. If a destroy/repair method results in a higher objective, then s^t is accepted with probability

$$\exp\left(\frac{f(s) - f(s^t)}{T}\right), \tag{2}$$

where $f(s)$ is defined as the objective value of s. Following Sacramento et al. (2019), we set the temperature T at a start value T_{st} and let it decrease linearly towards zero. As a stopping criterion for the algorithm we use time, which means that we want T to hit zero when a certain time has passed. For this reason, we control T using the elapsed time, which is measured with CPU time. We define t^{elap} as the elapsed time since the start of the algorithm and t^{max} as the time limit, such that the temperature is updated with the use of

$$T = T_{st}\left(1 - \frac{t^{elap}}{t^{max}}\right). \tag{3}$$

The algorithm stops as soon as $t^{elap} \geq t^{max}$. The pseudo-code for the algorithm is given in Algorithm 1. It is different from other ALNS metaheuristics in the sense that it restores the best solution so far if a defined number of iterations has passed without any improvement.

Algorithm 1: Pseudo-code for the ALNS algorithm

Input: Initial temperature: T_{st},
Max iterations without improvement: $noImpvMax$,
Time limit: t^{max}

$s \leftarrow$ `InitialSolution()` ;
$s^* \leftarrow s$;
$noImpv \leftarrow 0$;
while $t^{elap} < t^{max}$ **do**
  Choose a destroy method $d() \in \Omega^-$ and a repair method $r() \in \Omega^+$;
  $s^t \leftarrow r(d(s))$;
  $T = T_{st}(1 - t^{elap}/t^{max})$;
  if $Random(0,1) < \exp\left(\frac{f(s)-f(s^t)}{T}\right)$ **then**
    $s \leftarrow s^t$;
  if $f(s) < f(s^*)$ **then**
    $s^* \leftarrow s$;
    $noImpv \leftarrow 0$;
  else
    $noImpv \leftarrow noImpv + 1$;
    if $noImpv \geq noImpvMax$ **then**
      $s \leftarrow s^*$;
      $noImpv \leftarrow 0$;
  Update scores of Ω^- and Ω^+ based on acceptance criteria
return s^* ;

4.1.1 Initial solution

The creation of the initial solution is based on heuristics and consists of four steps (Sacramento et al., 2019). First, we use a construction algorithm that only regards service by trucks. Then, a local search algorithm is used, again only considering trucks for servicing customers. After this, we apply a drone addition algorithm. Finally, another local search heuristic is used, but this time considering both trucks and drones. The construction algorithm in the first step is chosen as the Nearest Neighbor Algorithm (Sacramento et al., 2019). The route is thus build by searching for the nearest neighbor to the last visited customer as long as the capacity and time constraints are still satisfied. As soon as one of these restrictions is exceeded, a new route is constructed and this process continues until all customers have been served. In the second step we improve the solution obtained from this Nearest Neighbor Algorithm with the use of an improvement heuristic through relocation moves (Fosin et al., 2014). In this heuristic, a customer is removed from the solution and is then inserted back in the best possible position.

The process is repeated until no more improvements can be made. In the last step we include drone delivery by letting some customers be served by a drone instead of a truck. First, we define a set C' as the subset of all customers that can be visited by a drone in the current solution. Next, for each customer $i \in C'$, the customer is removed from this route and all feasible sorties where this removed customer is served by a drone are considered. However, we only remove a possible drone customer from the routes when it is not a launch or recovery position for another drone operation. The best possible sortie is found by means of the function `FindSortie(`c, s, η`)`, which is shown in Algorithm 2. The method continues until all customers in C' are checked, after which the sortie incurring the biggest saving is added to the solution. This part of the method is repeated until no more savings can be obtained.

Algorithm 2: `FindSortie(c,s,`η`)` function for finding the best sortie for customer c in the partial solution s with respect to a threshold cost η.

Input: Partial solution: s,
Customer to insert as drone-customer: c,
Threshold cost: η

```
1  BestSortie = ∅ ;
2  for Each Route in s do
3      if Capacity(Route) + q_c < Q then
4          for Pair Positions (i,k) in Route where i < k do
5              Construct p = (i, c, k) with launch i, customer c and recovery k ;
6              Check feasibility for sortie p ;
7              if SL + SR + τ^D_ic + τ^D_ck + Se^D_c < e AND f(s) + CostSortie(p) < η then
8                  BestSortie ← p ;
9                  Update η ;
10 return BestSortie ;
```

Just like Sacramento et al. (2019), we attempt to improve the initial solution even further by using a local search algorithm with a string relocation neighborhood. The algorithm selects a string of customers and checks whether the costs can be decreased by inserting these customers in the same order somewhere else in the solution. The string can be relocated in the same route, in which case the operation is called a 2-opt move, but it can also be inserted into another route, defining a string relocation move. The string of customers to relocate can be of any length, but the start and end locations can not be customers that are visited while a drone is conducting a sortie. This heuristic is able to help decrease the objective, as it may eliminate crosses between visits, which cannot be achieved by the single relocation moves we have used before.

4.1.2 Destroy methods

The ALNS metaheuristic of Sacramento et al. (2019) removes a part of the current solution in every iteration. The number of customers β to eliminate is determined by the parameters δ, c_{low} and c_{lim} by means of the formula

$$\beta = \min(\max(c_{low}, \delta * |C|), c_{lim}). \tag{4}$$

Here, δ is defined as the ratio of customers to remove, and c_{low} and c_{lim} are the minimum and maximum number of customers to remove, respectively. In every iteration of the ALNS metaheuristic, the parameter c_{low} is selected as a random number between 1 and 3, and the parameter c_{lim} is equal to 40. We describe the two different destroy methods that are used in Sacramento et al. (2019), from which one is randomly chosen in each iteration with equal probability. These probabilities stay the same during the entire algorithm, which means that the adaptive part of ALNS is only applied to the repair methods.

The first destroy method, the **random destroy method**, eliminates β random customers from the solution. If a removed customer is a launch of recovery position for a drone operation, the corresponding drone customer is also removed. It can happen that the removal of one truck customer leads to the elimination of two drone customers, because a launch and recovery operation can take place at the same customer. Therefore, the method is able to remove more customers than specified by β.

In the second destroy method, the **cluster destroy method**, the elimination of customers is performed in an area around a random seed customer. First, a random customer c_1 is chosen and removed from the current solution. The next customer to be removed is determined randomly from a subset of the two customers that are closest to the focal customer c_1. This process is repeated until β customers have been eliminated. Similar to the previous destroy method, when a removed customer corresponds to a launch of recovery position, the associated drone customer(s) will be removed as well.

4.1.3 Repair methods

After a subset of customers, D, has been removed with the use of a destroy method, the ALNS algorithm reconstructs the partial solution by reinserting these deleted customers. The selected methods repair the solution by including the removed customers one-by-one is a way that seems most favorable. The repair methods make sure that no infeasible routes are created. If an

insertion of a customer is not feasible in the current partial solution, the customer is set aside for a later moment or a new route is constructed. The possibility to open such a new route guarantees that the repair methods always find a feasible solution, because it is always possible to serve just one customer on a truck route. With the selection of the repair methods, we arrive at the adaptive part of the metaheuristic. Sacramento et al. (2019) introduce four repair methods and the algorithm selects one of them according to their weights, which are updated iteratively, based on their historic performance.

The first repair method, the **greedy truck-first sortie-second repair method**, consists of two steps and is shown in Algorithm 3. In step one, all customers from D are inserted into the partial solution as truck visits. Step two swaps the service of some customers such that they are visited by a drone instead of a truck. First, the function `RandomCustomer(D)` chooses a random customer from D, after which the function `TruckBestInsertion(c,s)` inserts this customer in the least costly way in the partial solution s. To ensure that only feasible solutions are considered, the function checks if the endurance time of a sortie is not exceeded, as the truck-insertion can be carried out within a sortie. Lines 6-17 concern the second step of the repair method. Again, a random customer is chosen from all customers in the current solution, and we check whether this is a truck-only customer (no launch or recovery operations take place) and if the demand of this customer can be carried by a drone. If both conditions hold, we attempt to find an appropriate way of delivering their package with a drone by means of the function `FindSortie(c,s,`η`)`, as defined in Algorithm 2. Only in the case where this switch from truck-customer to drone-customer leads to a lower cost, the switch takes place.

The second repair method, the **nearby-area truck-first sortie-second repair method**, is very similar to the previous method. However, the search for new solutions in the neighborhood is no longer carried out by the Best Insertion algorithm. The removed customers are placed in the routes as truck visits in a feasible position that is randomly chosen from a set of close by positions to the customer. This nearby area consists of all positions where at least one neighbor is located within a 5-mile range. In the second step, the selection of a random customer and the identification of all possible sorties happens in the same way as before. Nonetheless, instead of choosing the sortie with the least cost increase, we now select a random sortie that does not expand the cost of the partial solution with more than 10% in comparison with the solution before the removal. Therefore, this method is seen as weaker than the previous repair method. However, this method can have advantages for small instances, because it gives a greater variability when seeking for sorties.

Algorithm 3: Greedy truck-first sortie-second repair method.

Input: Partial solution: s,
Set of free customers: D

while $D \neq \emptyset$ **do**
  $c \leftarrow$ `RandomCustomer(`D`)` ;
  $D = D \backslash \{c\}$;
  `TruckBestInsertion(`c, s`)` ;
$C =$ `AllCustomers(s)` ;
while $C \neq \emptyset$ **do**
  $c \leftarrow$ `RandomCustomer(`C`)` ;
  $C = C \backslash \{c\}$;
  if $q_c \leq Q^D$ *AND* $c \neq$ *launch and/or recovery position* **then**
    $s' \leftarrow s$;
    $\eta = f(s')$;
    $s \leftarrow s \backslash \{c\}$;
    $p \leftarrow$ `FindSortie(`c, s, η`)` ;
    if $p \neq \emptyset$ **then**
      $s \leftarrow s \cup \{p\}$;
    else
      $s \leftarrow s'$;
return s ;

The third repair method, the **closest insertion repair method**, tries to include the removed customers by using trucks as well as drones. The method attempts to insert a free customer c into one route only, r, namely the one on which the customer closest to c is visited. Every feasible truck and drone insertion of c into this route is considered, and the one that results in the least cost increase is performed. If there is no possibility to include customer c into route r, then c is added to a set of leftover customers D_N. When all removed customers are considered, the repair method calls the greedy truck-first sortie-second repair method to insert these leftover customers.

The fourth and final repair method is the **heavy insertion repair method**. This method first removes all customers from D that have a demand that can only be satisfied by a truck and adds these customers to a new set D_T. Then, a random truck-customer c from D_T is placed in the current partial solution s by means of the Best Insertion Algorithm, which we also use in Algorithm 3. This process is repeated until all customers in D_T are inserted. Next, the customers that are allowed to be visited by a drone are inserted with the use of the close insertion repair method described above. This means that this repair method may also use the greedy truck-first sortie-second repair method.

4.2 Vehicle Routing Problem with Multi-Visit Drones

We now consider the case in which a drone can serve multiple customers before returning to its corresponding truck. We solve this new problem with an adjusted version of the ALNS metaheuristic that we defined in Algorithm 1.

4.2.1 Initial solution

We create an initial solution based on the same heuristics that we used for the VRP-D with a single delivery per drone operation. Just as before, we use the Nearest Neighbor Algorithm to construct the first solution and improve these routes by means of single relocation moves. However, the inclusion of drone visits is executed differently, as a potential drone customers is now not only allowed to be inserted between two customers served by a truck, but also between a truck customer and a drone customer, or even between two drone customers. The feasibility check in Line 6 in Algorithm 2 is therefore more extensive. When the considered insertion is between two truck visits, we have to check whether the customer to be inserted has a capacity below Q^D and if the drone can travel from the truck to the customer and back before its battery runs out. Otherwise, we first have to track down how much weight is already carried by the drone and if the package of the customer to insert can be added to this without exceeding the drone capacity. Then, the travel time of the current route of the drone needs to be computed, after which we can check if adding another drone customer is reachable with respect to the battery endurance. With this adjustment in finding potential sorties, we repeatedly insert the best possible sortie until no improvements can be made. As a last step, we again try to improve the obtained solution with the use of 2-opt and string relocation moves.

4.2.2 Destroy methods

The design of the destroy methods is the same for the VRP-MVD as for the VRP-D. However, the removal of corresponding drone customers does need to change. Before, when removing a truck customer, we sometimes needed to remove one or two drone customers as well if that customer corresponded to a launch and/or recovery position. But now, there can be multiple drone visits in a row, which all have the same launch and recovery position. This means that if a customer in one of these two positions is removed, all these drone customers have to be eliminated too. In comparison to the previous destroy methods, these methods can be seen as stronger because more customers may be removed, such that the solution is destroyed in a greater way.

4.2.3 Repair methods

The truck insertions that are considered in the repair methods do not need to be changed. These insertions are allowed to be performed within a sortie, and although these sorties may now be constructed differently, the check whether the endurance time of such a sortie is still respected after the insertion stays the same. However, the insertion of possible drone customers have to be altered in a same way as we did for the initial solution. The greedy truck-first sortie-second repair method already takes this into account as it directly calls the `FindSortie(c,s,`η`)` in Line 13 in Algorithm 3.

5 Results

In this section, we discuss how we set up our experiments and show the results we have obtained by implementing them in Java. Next to this, we mention our main findings and give an answer to the central question of this paper.

5.1 Parameter values

Different parameters concerning the characteristics and operation times of both the drone and the truck are used in the VRP-D as well as in the VRP-MVD. We make use of the same values as Sacramento et al. (2019), which they have found in the literature or in prototype models of different companies. These parameter values are summarized in Table 2. As the trucks are equipped with drones material, such as the drone itself, batteries and tools, the maximum capacity for packages is reduced in comparison to when a truck operates on its own.

Table 2: Parameter values.

Parameter	Notation	Value
Launch time drone	SL	1 min
Recovery time drone	SR	1 min
Service time drone	SS	1 min
Truck speed	v^T	35 mph
Drone speed	v^D	50 mph
Endurance	e	30 min
Truck capacity with drones	Q	1300 kg
Truck capacity without drones	Q^*	1400 kg
Drone capacity	Q^D	5 kg
Fuel price	fp	1.13 €/l
Fuel consumption	fc	0.07 l/km
Drone factor cost	α	10%
Maximum route duration	T_{max}	8h

5.2 Test instances

We make use of the instances that are generated by Sacramento et al. (2019) in order to test the performance of the algorithm. They generated these data sets in the following way. The central depot from which all trucks must depart from and return to is always located at coordinate (0,0). The customers to be served are generated in a grid of dimensions $2d \times 2d$ around this depot such that their coordinates follow a uniform distribution $U(-d, d)$. As these coordinates are in miles, we first multiply every coordinate with the constant 1.61 in order to convert it to kilometers. The instances that they have created are named `n.m.t.`, where `n` is the number of customers in the instance, `m` the dimension of the grid and `t` the generic name of the scenario. The instances we use contain between 6 and 200 customers and have grid sizes varying from 5×5 to 40×40 (Sacramento et al., 2019). All instances can be found at Zenodo[1]. The demand of a customer is generated according to a uniform distribution, depending on whether its package can be delivered by a drone or not. Let c be a customer and $0 \leq p < 1$ a random number associated with this customer, then their demand in kilograms is given by

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

5.3 Experiments

Next to the values for the model-parameters, we need values for the algorithm-parameters in order to study the performance of the ALNS metaheuristic. These were found by Sacramento et al. (2019) and are set as follows: initial temperature factor $T^*_{ST} = 0.004$, degree of destruction $\delta = 0.15$, and non-improvement parameter $noImprovMax = 1000$. If the algorithm is not able to find better solutions in the given number of iterations, it continues its search with the best known solution. The remaining algorithm-parameters, concerning the adaptive part of the metaheuristic, are the reaction factor $\rho = 0.9$ and the scores of the methods $\sigma_1 = 33$, $\sigma_2 = 9$, $\sigma_3 = 13$, $\sigma_4 = 0$ (Sacramento et al., 2019). Just as in Sacramento et al. (2019), the initial temperature T_{ST} is determined by multiplying T^*_{ST} by the value of the initial solution, such that the algorithm is able to adjust the temperature according to the instance size. For instances with less then 20 customers, the initial temperature is increased by 10% in order to avoid too small temperatures.

[1] https://doi.org/10.5281/zenodo.2572764

5.4 Vehicle Routing Problem with Drones

In order to test the performance of the metaheuristic, we want to compare the results of our implementation with the objectives that Sacramento et al. (2019) found. For this purpose, we use all of the 112 instances generated by Sacramento et al. (2019), consisting of 6 to 200 customers located in areas from 5 × 5 miles to 40 × 40 miles. The solutions for the metaheuristic are obtained by applying the algorithm 10 times to each instance with an execution time of $t^{max} = 5$ minutes (Sacramento et al., 2019). Of these 10 runs, we keep the best objective found, z^{ALNS}, the average objective, μ^{ALNS}, and the corresponding standard deviation, σ^{ALNS}. For the truck-only case, we set the drone capacity equal $Q^D = 0$ and the truck capacity to $Q^* = 1400$, apply the ALNS metaheuristic 10 times and keep again the best objective, z^{VRP}. For comparison, we highlight the same two different KPIs from these experiments as Sacramento et al. (2019). The first KPI is the saving obtained by delivering packages using both trucks and drones compared to the case where only trucks are used (SVRP). Next to this, we express the saving acquired by the ALNS metaheuristic over the initial solution where both trucks and drones are considered (SI). For this, we define z^{in} as the objective found by the heuristic for the initial solution. The KPIs are thus calculated with the following formulas:

$$\text{SVRP} = 1 - \frac{z^{ALNS}}{z^{VRP}} \quad \text{and} \quad \text{SI} = 1 - \frac{z^{ALNS}}{z^{in}}. \tag{6}$$

Table 3 in Appendix A shows the aforementioned objectives and these two KPIs for all 112 instances, together with the number of customers with a capacity that can be carried by a drone, $|C'|$, the average number of drone customers in the solutions, $\#C'$, and the average number of routes, $\#V$. For ease of comparison, we calculated the difference between our best found objective and the lowest total cost that Sacramento et al. (2019) obtained as a percentage of their solution for every instance. These percentages are shown in Figure 1. It can be seen that for 18.8% of the instances we were able to find the same best objective by means of the ALNS metaheuristic as Sacramento et al. (2019) within 10 runs. For 42.8% of the instances, our best found objective was within a 5% difference and in 17.0% of the situations we found a best total cost that was between 5% and 10% higher. This last case occurred mostly for instances with a great number of customers (≥ 50) which are located relatively far from each other. However, for 12 instances (10.7%), we were able to find an even cheaper set of routes than discovered by Sacramento et al. (2019). Surprisingly, this also often seems to be the case for situations with a lot of customers. There are also a few instances for which we were not able to come close to the

best found solution of Sacramento et al. (2019). In most of these cases, the ALNS metaheuristic was not capable to improve the initial solution (much).

Our implementation of the metaheuristic shows not much difficulty in finding the same best solution as Sacramento et al. (2019) for instances with a small number of customers that are located close to each other. Figure 1 shows that, for the rest of the instances, the percentage difference between our best found objective and the one of Sacramento et al. (2019) is steady with respect to the number of customers. Whether there are 50 or 200 customers, the differences between objectives for the various instances are distributed quite similar over the range between 0% and 10%.

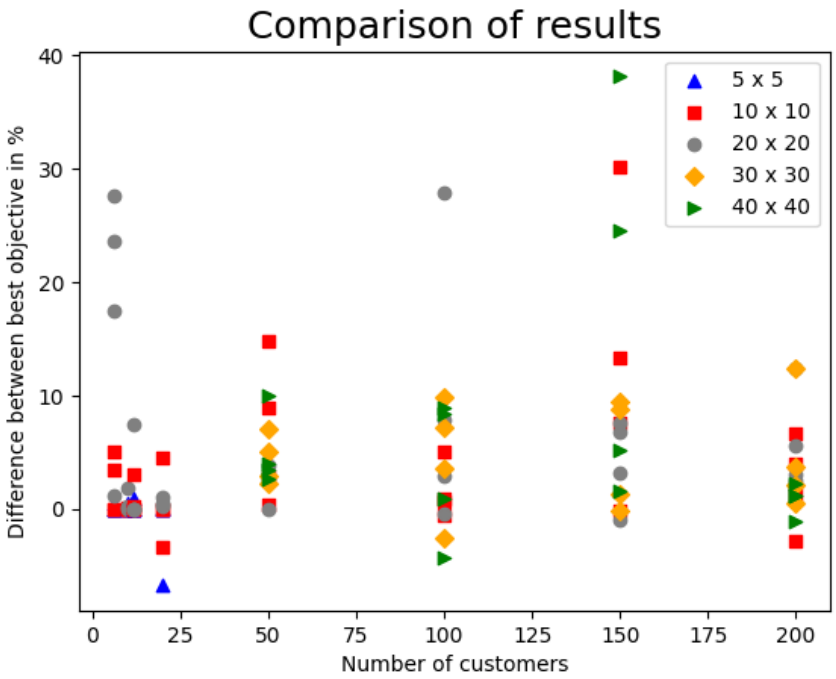
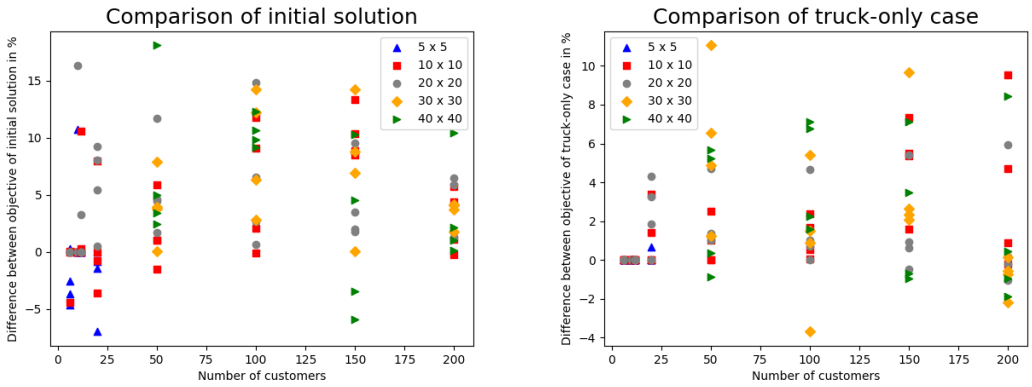


Figure 1: Comparison between the best objective value for every instance.

Additionally, we compare the objectives we have found for the initial solutions to the ones that Sacramento et al. (2019) discovered. We calculate again the difference in percentage for every instance, which can be found in Figure 2a. The total costs of the initial solution that we obtained by using our implementation of the metaheuristic was the same as in Sacramento et al. (2019) for 22.3% of the instances. All of these instances had less than 50 customers. We found an better initial solution in 11.6% of the cases and for the situations with 50 or more customers, we almost always obtained a total cost that was higher than that of Sacramento et al. (2019). These solutions were mostly between 0% and 10% more expensive. There seems to be

a correlation between the difference in initial solution and the difference in best objective value. The more the total costs of our initial solution differ from those of Sacramento et al. (2019), the more difficulty the algorithm experiences in coming close to their best objective value.

Lastly, we compare the objectives of the truck-only case, for which the percentage difference for all instances are shown in Figure 2b. For almost all instances with less than 50 customers, we were able to find the same best solution as Sacramento et al. (2019). However, for the greatest part of the instances, the objective was higher, with a maximum of 11.1%. There were also cases where we were able to find a better objective. From the figure, it can be seen that, in general, the difference between total costs becomes greater when the number of customers increases. So, for situations with a large number of customers, the algorithm struggles more to come close to the lowest total costs that Sacramento et al. (2019) found for the truck-only case.



(a) Comparison between the objective of the initial solution for every instance.

(b) Comparison between the objective of the truck-only case for every instance.

Figure 2: Comparison between our results and those of Sacramento et al. (2019)

5.5 Vehicle Routing Problem with Multi-Visit Drones

In order to investigate the difference between the total costs for the VRP-MVD and the VRP-D, we apply the adjusted metaheuristic for the VRP-MVD in the same way. For each instance, we run the algorithm 10 times and extract the best objective value, the average objective and the standard deviation. Next to this, we again notate the average number of customers and the average number of routes for that scenario. Just like before, we want to obtain the saving acquired by the ALNS metaheuristic over the initial solution (SI). So, for every instance, the objective found for generating initial solutions is documented. All these data can be found in Table 4 in Appendix B.

We want to know what happens with the objective if we allow a drone to deliver multiple

packages in one trip compared to when it has to return to their truck directly after serving one customer. For this purpose, we compute the percentage decrease in total costs for the VRP-MVD with respect to the VRP-D for all instances, which is displayed in Figure 3. From this figure, we can see that for almost all situations with less than 100 customers, the total costs have decreased. For instances where customers lie close to each other the savings are a lot higher than for instances where the customers are located in a greater range. The figure shows a declining trend, which means that allowing a drone to carry multiple packages becomes less attractive when the number of customers increases. For instances with 100 or more customers, there are still cases where savings up to 20% are achieved, but for more than half of the situations the algorithm was not able to find a better solution than for the VRP-D case. In theory, it is therefore better to take the solution for the VRP-D case in those cases, but because the algorithm also finds a different initial solution, it is not always able to obtain this solution within 5 minutes.

When there are only a few customers (≤ 20) that need to be served, which are located relatively close to each other ($\leq 10 \times 10$), the average saving that can be obtained is equal to 30%. Therefore, it is beneficial to adapt the drone in order for it to be capable of carrying multiple commodities, when the corresponding costs are not more than 30% of the total costs when a drone is only able to serve one customer each trip. For situations with a small number of customers (≤ 20) that lie relatively far from each other (20×20), the costs for adapting the drone should not be more than 9% for it to be advantageous. For 50 customers, these threshold savings are about 14% for small ranges (≤ 20) and 4% when customers are located further from each other. When situations with more than 50 customers are considered, there are only savings to be obtained in some of the cases, which means that it is not clear whether the adaption to the drone will be beneficial or a waste of money.

To conclude in which situations the adjustments to a drone are useful, we show the increase in the average number of drone customers in the solutions for each instance in Figure 4. It can be seen that the number of customers that is served with a drone instead of a truck is much higher for the VRP-MVD than for the VRP-D when we are considering 50 or less customers. In most of the cases, the number of drone customers increases between 20% and 60% when a drone is allowed to serve several customers in one trip. Especially for instances with just a few customers (≤ 12), the number of customers served by a drone increases much more when customers are located close to each other. Just like before with the objective, there seems to be a decreasing trend in the number of drone customers. Most of the time, there is still an increase in drone customers, but this increase becomes less when more customers are considered.

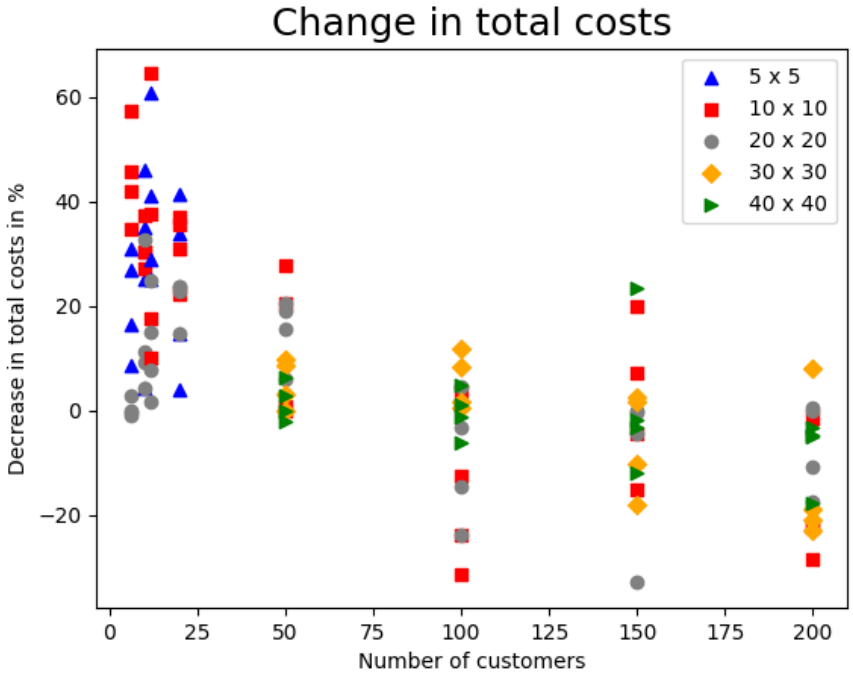


Figure 3: Comparison between the best objective value of the VRP-MVD and the VRP-D.

When we compare Figure 4 with Figure 3, it is noticeable that for the instances with 100 or 150 customers, the number of drone customers increases often when allowing a drone to deliver multiple packages in one operation, while the total costs do not decrease. For situations with 200 customers, the number of customers served by a drone instead of a truck is mostly less than for the VRP-D case.

Another interesting feature to look at is the average number of customers that is served by a drone within one drone trip. Therefore, we displayed this number for every instance in Figure 5. It is clear to see that the figure shows a decreasing trend, just like we saw in Figure 3 and Figure 4. For the situations with less than 100 customers, the number of drone customers per trip is considerably increased in comparison to the case where a drone is only allowed to serve one customer at a time. There is a noticeable difference when we consider the range in which the customers are located. For instances where customers lie close together, more customers are served within one operation compared to the instances where customers are more widely spread. For the cases with 100 customers or more, the number of customers visited per drone trip decreases fast. In these situations, a drone is only able to visit one or two customers in one time. Figure 5 shows that in almost all situations, the algorithm makes use of the fact that a drone is able to carry multiple packages at the same time. Taking this observation into account

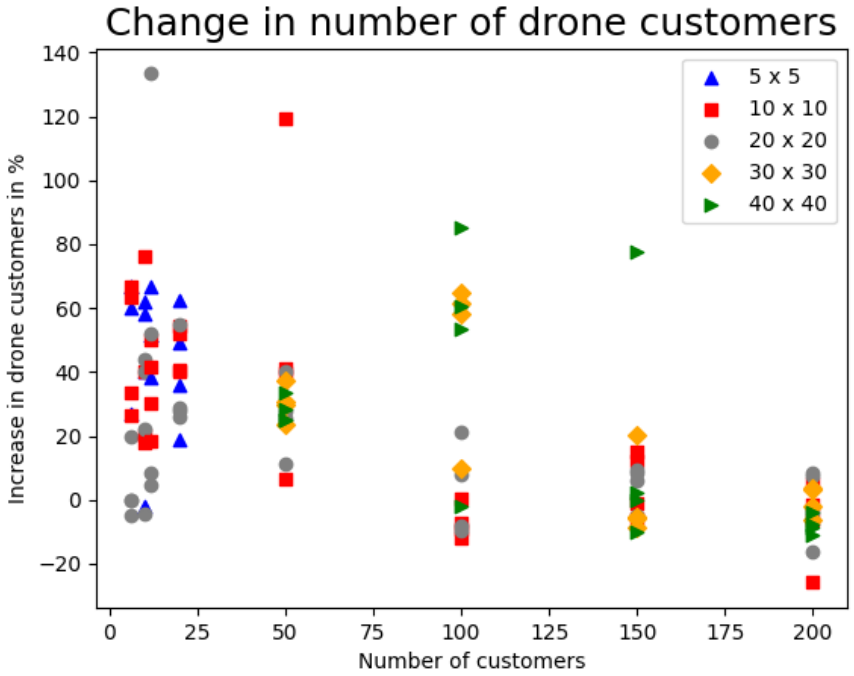


Figure 4: Comparison between number of drone customers for every instance.

when comparing to the other figures, we see that although the drone delivers multiple packages within one trip, the total number of drone customers does not always increase, meaning that the number of drone trips is decreased. This could be an explanation for the fact that the total costs are not decreased all the time when considering drones that can carry multiple packages simultaneously.

6 Conclusion

The focus of our research consisted of the following central question: *When is making adjustments to a drone in order for it to carry multiple packages simultaneous beneficial?* In order to answer this question, we applied an adapted version of the ALNS metaheuristic of Sacramento et al. (2019) to the VRP-MVD. However, to test the performance of our implementation of this metaheuristic, we first applied the algorithm to the VRP-D and compared our results to those of Sacramento et al. (2019). We documented the best objective we have found for 112 instances, and compared these to the lowest total costs that Sacramento et al. (2019) discovered. We obtained the same best solution in about 20% of the cases, and for most of the other situations, the routes we have found were less than 10% more expensive. In some cases, we were even able to find a lower cost. For the truck-only case, the difference between our objective and the one

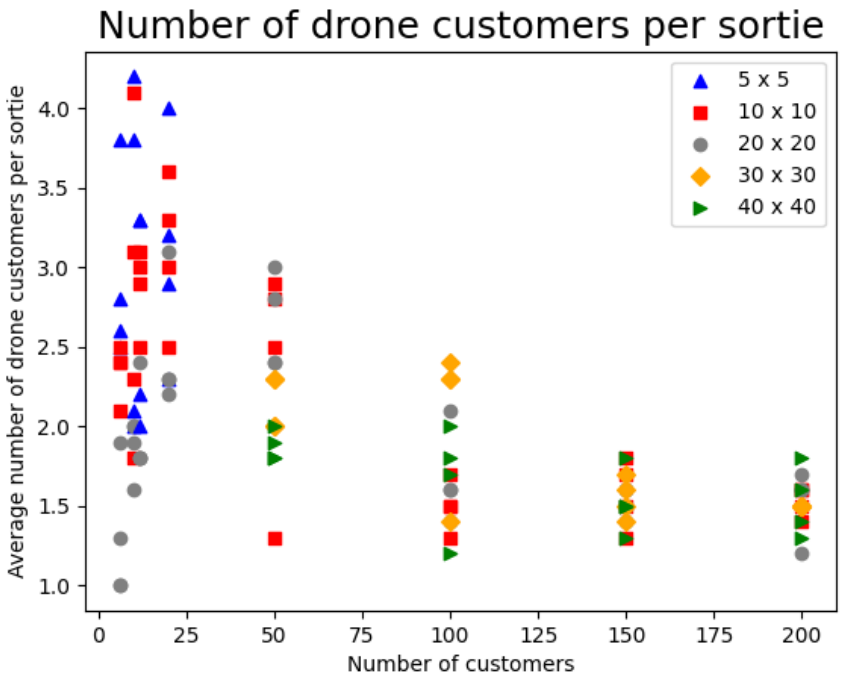


Figure 5: Average number of drone customers within one drone trip.

of Sacramento et al. (2019) was less than 5% in more than 85% of the cases.

After confirming the performance of the metaheuristic, we ran the algorithm on the VRP-MVD. The most important thing that we have discovered is that the total costs of the routes after making the adjustments to the drones definitely decreases when considering less than 100 customers. The percentage decrease differs for the several scenarios, which means that it depends on the situation for which the drones are used whether the adaption is advantageous. When 100 or more customers need to be served, allowing the drone to deliver more than one package within a trip is only cost-effective in some of the cases.

When looking at the number of drone customers in the solutions of the VRP-MVD compared to the solutions of the VRP-D, we conclude that allowing drones to carry multiple packages at the same time is only useful when not too many customers have to be served (< 200). The results show that the increase in number of packages delivered by a drone instead of a truck decreases with the number of customers. Therefore, the adaption to the drones makes more of a difference is scenarios where not a lot of customers are considered. The number of drone visits within one trip are however almost always greater than one when considering the VRP-MVD. This means that although the number of of drone customers per trip increases, the total number of customers served by a drone does not grow in all cases.

There are a lot of extensions and variants of the VRP-D and the VRP-MVD that can be studied in future work. One interesting adjustment would be the number of drones that a truck is equipped with. When increasing this number of drones, a lot more customers could be served by means of a drone instead of a truck, which can have a significantly effect on the total costs. Another intriguing aspect for further research is the consideration that the load that the drone is carrying affects its speed and battery duration. For the VRP-MVD, this would include a function that takes into account the different weight of the load during the drone trip, such that the drone can move faster and save some battery after one of the packages is delivered. Lastly, it would be interesting to investigate a different variant of the problem with respect to the launch and recovery positions of the drones. One could research what will happen if a truck is allowed to wait for a drone in the same place that is has launched. In this way, the drone can carry out several trips from this location, while the truck is waiting and thus saving travel costs. Another way to decrease the total cost is to consider the customer locations and the collection of launch and recovery positions as two separate sets. By doing so, the drone is able to launch and recover while the truck is carrying out its route, such that is can save battery by travelling with the truck for a longer period of time.

References

Agatz, N., Bouman, P., & Schmidt, M. (2018). Optimization Approaches for the Traveling Salesman Problem with Drone. *Transportation Science*.

Applegate, D., Bixby, R., Chvátal, V., & Cook, W. (2011). The Traveling Salesman Problem: A Computational Study. *Princeton University Press*.

Azi, N., Gendreau, M., & Potvin, J. (2014). An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Elsevier*.

Choi-Fitzpatrick, A., Chavarria, D., Cychosz, E., Dingens, J., Duffey, M., Koebel, K., ... Juskauskas, T. (2016). Up in the Air: A Global Estimate of Non-Violent Drone Use 2009-2015. *University of San Diego*.

Fosin, J., Carić, T., & Ivanjko, E. (2014). Vehicle Routing Optimization Using Multiple Local Search Improvements. *Automatika*.

French, S. (2015). Drone delivery is already here — and it works. *MarketWatch*.

The Future of Drones in Transportation and Logistics. (2022). *American Journal of Transportation*.

Ha, Q., Deville, Y., Pham, Q., & Hà, M. (2018). On the min-cost Traveling Salesman Problem with Drone. *Elsevier*.

Hern, A. (2014). DHL launches first commercial drone 'parcelcopter' delivery service. *https://www.theguardian.com/technology/2014/sep/25/german-dhl-launches-first-commercial-drone-delivery-service*.

Lee, T. (2022). How Walmart and Alphabet jumped ahead of Amazon in drone delivery. *https://arstechnica.com/tech-policy/2022/04/how-walmart-and-alphabet-jumped-ahead-of-amazon-in-drone-delivery/*.

Luo, Z., Poon, M., Zhang, Z., Liu, Z., & Lim, A. (2021). The Multi-visit Traveling Salesman Problem with Multi-Drones. *Elsevier*.

Murray, C., & Chu, A. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Elsevier*.

Poikonen, S., & Golden, B. (2020). Multi-visit drone routing problem. *Elsevier*.

Ropke, S., & Pisinger, D. (2006). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. *Transportation Science*.

Rose, C. (2013). Amazon's Jeff Bezos looks to the future. *https://www.cbsnews.com/news/amazons-jeff-bezos-looks-to-the-future/*.

Sacramento, D., Pisinger, D., & Ropke, S. (2019). An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Elsevier*.

Shaw, P. (1997). A new local search algorithm providing high qualitysolutions to vehicle routing problems. *Technical report, Department of Computer Science, University of Strathclyde, Scotland*.

Trop, J. (2016). Drone Delivery is About to Disrupt the Trucking Industry. *https://www.trucks.com*.

Wang, X., Poikonen, S., & Golden, B. (2017). The vehicle routing problem with drones: several worst-case results. *Springer Link*.

A Performance of the metaheuristic for the VRP-D

Table 3: Performance of the metaheuristic for the VRP-D model.

Scenario	$\|C'\|$	z^{ALNS}	μ^{ALNS}	σ^{ALNS}	$\#C'$	$\#V$	z^{in}	$SI(\%)$	z^{VRP}	$SVRP(\%)$
6.5.1	5	1.105	1.105	0.000	3	1	1.340	17.58	1.744	36.68
6.5.2	6	0.842	0.842	0.000	3	1	1.057	20.33	1.417	40.56
6.5.3	5	1.211	1.211	0.000	3	1	1.303	7.00	1.782	32.01
6.5.4	5	0.946	0.946	0.000	3	1	0.982	3.69	1.816	47.92
6.10.1	5	2.529	2.529	0.000	3	1	2.880	12.18	3.026	16.43
6.10.2	6	1.738	1.738	0.000	3	1	1.771	1.88	3.266	46.78
6.10.3	6	1.326	1.352	0.032	3.8	1.8	1.903	30.36	3.184	58.37
6.10.4	6	1.443	1.443	0.000	3	1	1.658	12.94	2.654	45.63
6.20.1	6	3.146	3.175	0.060	4	2	3.670	14.30	5.819	45.94
6.20.2	5	4.371	4.370	0.000	3	1	5.561	21.41	5.951	26.56
6.20.3	6	4.884	4.884	0.000	3	1	5.052	3.31	7.487	34.79
6.20.4	6	4.548	4.548	0.000	3	1	4.548	0.00	7.253	37.30
10.5.1	5	1.664	1.664	0.000	4	1	1.664	0.00	2.004	17.00
10.5.2	9	1.452	1.452	0.000	5	1	1.515	4.18	1.765	17.72
10.5.3	8	1.475	1.516	0.051	5	1	1.863	20.86	2.108	30.04
10.5.4	9	1.285	1.285	0.000	5	1	1.997	35.66	2.152	40.29
10.10.1	8	2.326	2.364	0.046	5	1	2.674	12.99	4.378	46.86
10.10.2	8	3.159	3.207	0.067	4.9	1	3.509	9.98	3.853	18.02
10.10.3	7	2.555	2.739	0.358	5.1	1.4	3.546	27.93	3.942	35.17
10.10.4	9	2.539	2.539	0.000	5	1	2.871	11.56	3.671	30.82
10.20.1	7	4.535	4.535	0.000	4	1	5.276	14.05	7.100	36.13
10.20.2	8	6.177	6.177	0.000	5	1	6.784	8.95	8.186	24.54
10.20.3	9	4.546	4.546	0.000	5	1	5.165	11.99	7.159	36.49
10.20.4	7	6.172	6.233	0.040	4.7	1	6.742	8.44	7.659	19.41
12.5.1	9	1.374	1.374	0.000	6	1	1.530	10.21	1.777	22.68
12.5.2	12	1.069	1.069	0.000	6	1	1.783	40.03	2.085	48.72
12.5.3	10	1.448	1.448	0.000	6	1	1.629	11.15	2.326	37.76
12.5.4	10	1.581	1.606	0.024	6	1	1.756	9.99	2.193	27.91
12.10.1	10	2.764	2.764	0.000	6	1	3.771	26.69	4.175	66.21
12.10.2	10	2.689	2.808	0.053	6	1	3.658	26.47	4.001	32.79
12.10.3	9	2.882	2.882	0.000	6	1	3.692	21.96	3.895	26.03
12.10.4	10	2.314	2.314	0.000	6	1	3.171	27.03	4.440	47.88
12.20.1	11	5.778	5.854	0.166	6.8	1.8	7.019	17.68	9.692	40.39
12.20.2	10	8.273	8.273	0.000	3	1	8.273	0.00	9.919	18.82
12.20.3	9	4.167	4.340	0.143	4.8	1.6	5.616	25.80	6.653	37.37
12.20.4	11	6.543	6.659	0.094	6	1	7.715	15.19	8.172	19.94
20.5.1	15	1.793	1.795	0.000	9.8	1	2.035	11.85	2.553	29.76
20.5.2	14	1.822	1.822	0.000	8	1	1.904	4.33	2.598	29.86
20.5.3	19	1.487	1.487	0.000	10	1	1.903	21.83	2.110	29.51
20.5.4	18	1.379	1.457	0.055	9.8	1	1.818	24.15	2.164	36.29
20.10.1	17	3.401	3.401	0.000	9	1	3.894	12.66	5.273	35.51
20.10.2	19	3.089	3.089	0.000	10	1	4.586	32.64	5.253	41.19
20.10.3	19	3.703	3.737	0.047	10	1	5.001	25.96	5.218	29.03

(continued on next page)

Table 3 – continued from previous page

Scenario	$\|C'\|$	z^{ALNS}	μ^{ALNS}	σ^{ALNS}	$\#C'$	$\#V$	z^{in}	$SI(\%)$	z^{VRP}	$SVRP(\%)$
20.10.4	15	3.197	3.344	0.153	9.9	1	4.365	26.76	5.699	43.91
20.20.1	19	7.381	7.789	0.392	9.7	1	8.791	16.04	9.918	25.58
20.20.2	16	7.571	8.301	0.558	8.2	1.1	9.055	16.39	9.959	23.98
20.20.3	18	7.541	7.729	0.175	10	1	9.213	18.15	10.846	30.47
20.20.4	17	7.045	7.283	0.174	9.4	1.1	8.715	19.17	10.719	34.28
50.10.1	37	5.880	6.054	0.121	16.5	1	6.162	4.57	6.961	15.52
50.10.2	41	5.745	6.132	0.142	22.3	1	6.473	11.24	7.747	25.84
50.10.3	44	6.227	7.321	0.421	15.6	1	7.527	17.27	8.093	23.05
50.10.4	44	5.673	5.946	0.235	24.9	1	6.871	17.44	7.791	27.18
50.20.1	41	10.855	11.255	0.362	23.1	1	13.818	21.45	14.467	24.97
50.20.2	44	10.056	10.531	0.300	24.4	1	13.251	24.11	14.595	31.10
50.20.3	44	10.845	11.423	0.371	24.4	1	16.022	32.31	16.156	32.87
50.20.4	46	10.976	11.291	0.171	24.5	1	12.999	15.57	14.768	25.68
50.30.1	40	16.617	16.704	0.056	23.8	1	19.874	16.39	23.299	28.68
50.30.2	39	16.084	18.317	0.955	21.8	1	21.633	25.65	21.318	24.55
50.30.3	43	17.142	17.471	0.248	23.8	1	21.936	21.85	25.290	32.22
50.30.4	40	18.819	20.430	1.298	20.4	1	22.929	17.92	24.806	24.13
50.40.1	46	22.504	24.488	1.276	22.4	1	25.956	13.30	29.772	24.41
50.40.2	41	21.438	22.745	0.516	22	1.1	27.282	21.42	28.404	24.52
50.40.3	42	23.250	24.468	0.945	23.5	1	28.428	18.22	30.147	22.88
50.40.4	41	23.103	26.194	1.930	19.8	1	28.760	19.67	29.173	20.81
100.10.1	89	6.902	8.083	0.945	40.2	1	9.104	24.19	10.236	32.57
100.10.2	89	7.539	9.206	1.033	39.1	1	10.322	26.96	10.388	27.42
100.10.3	91	7.545	8.261	0.485	34.8	1	8.799	14.26	10.125	25.49
100.10.4	82	7.389	9.137	0.859	27.2	1	9.797	24.57	9.811	24.68
100.20.1	90	14.680	17.707	1.222	33.5	1	18.760	21.75	18.685	21.43
100.20.2	89	14.554	16.404	1.130	33	1	17.263	15.69	20.113	27.64
100.20.3	87	13.655	17.798	1.438	33.3	1	18.604	26.60	19.767	30.92
100.20.4	89	17.715	18.605	0.415	32.3	1	19.001	6.77	20.744	14.60
100.30.1	84	24.204	29.190	2.615	30.7	1	31.597	23.40	31.896	24.11
100.30.2	87	21.747	26.773	2.608	41.2	1.4	29.916	27.31	29.593	26.51
100.30.3	91	26.061	27.934	1.287	39.9	1	30.823	15.45	31.516	17.31
100.30.4	88	23.171	26.204	1.617	29.9	1	27.349	15.28	28.366	18.31
100.40.1	85	27.897	38.292	5.231	34.3	1.9	41.452	32.70	42.107	33.75
100.40.2	85	31.274	38.819	6.504	29.3	2	43.626	28.31	42.157	25.82
100.40.3	89	31.468	38.451	3.674	34.6	1.9	41.216	23.65	41.417	24.02
100.40.4	87	31.546	37.664	3.605	43.2	1.6	41.745	24.43	42.899	26.46
150.10.1	125	8.782	11.743	1.322	50.1	1	12.586	30.22	12.702	30.86
150.10.2	126	10.748	11.552	0.348	48.3	1	11.879	9.52	12.449	13.66
150.10.3	141	9.141	12.143	1.122	47.3	1	13.072	30.07	12.995	29.66
150.10.4	120	10.014	11.822	0.983	41	1	12.747	21.44	12.126	17.42
150.20.1	131	18.488	20.401	1.421	49.6	1	21.790	15.15	24.238	23.72
150.20.2	131	17.907	22.716	2.372	47.4	1	24.539	27.03	25.500	29.78
150.20.3	128	17.250	21.068	2.582	52.7	1	23.960	28.00	24.174	28.64
150.20.4	130	17.417	23.245	3.373	45.8	1	25.844	32.61	24.371	28.53
150.30.1	130	26.334	31.879	2.771	49.4	1	33.563	21.54	36.728	28.30

(continued on next page)

Table 3 – continued from previous page

Scenario	$\|C'\|$	z^{ALNS}	μ^{ALNS}	σ^{ALNS}	$\#C'$	$\#V$	z^{in}	$SI(\%)$	z^{VRP}	$SVRP(\%)$
150.30.2	125	26.179	34.555	4.927	45.8	1.8	38.557	32.10	36.319	27.92
150.30.3	130	27.539	33.327	3.645	49.5	2	35.948	23.39	36.374	24.29
150.30.4	125	28.564	33.528	2.063	39	2	34.652	17.57	38.421	25.66
150.40.1	137	42.374	43.322	0.538	44.1	2	43.953	3.59	49.023	13.56
150.40.2	124	50.505	52.253	0.648	35.2	2	52.689	4.14	53.564	5.71
150.40.3	125	38.545	43.396	2.754	50	2	46.491	17.09	51.242	24.78
150.40.4	129	35.569	45.856	3.946	48	2	48.668	26.91	48.551	26.74
200.10.1	173	10.223	12.236	0.763	62.7	1	12.680	19.38	13.840	26.13
200.10.2	173	10.134	12.821	1.188	64.8	1	13.568	25.31	14.578	30.48
200.10.3	177	10.187	13.036	0.999	62.5	1	13.537	24.75	13.907	26.75
200.10.4	176	11.050	12.847	0.927	66.5	1	13.846	20.19	15.236	27.48
200.20.1	178	21.738	25.684	1.991	64.9	1	26.855	19.05	29.785	27.02
200.20.2	168	22.654	26.329	2.347	62.1	2	28.510	20.54	27.926	18.88
200.20.3	176	21.491	25.110	1.843	59.4	1	26.556	19.07	27.108	20.72
200.20.4	172	21.634	25.182	1.415	57.1	1	26.088	17.07	26.675	18.90
200.30.1	171	31.472	37.362	2.621	60.8	2	39.471	20.26	40.857	22.97
200.30.2	176	32.970	37.018	2.889	62.8	2	39.730	17.01	41.744	21.02
200.30.3	171	32.955	40.382	2.529	56.4	2	41.680	20.93	41.836	21.23
200.30.4	172	36.083	39.986	1.667	63.3	2	41.221	12.46	42.244	14.58
200.40.1	172	42.426	50.289	7.338	64.2	2	57.647	26.40	56.278	24.61
200.40.2	178	42.773	48.054	5.418	67.3	2	56.316	24.05	56.101	23.76
200.40.3	165	43.870	53.564	6.301	62.9	2	59.729	26.55	61.641	28.83
200.40.4	178	42.571	48.694	4.261	70.1	2	53.444	20.34	54.638	22.09

B Performance of the metaheuristic for the VRP-MVD

Table 4: Performance of the metaheuristic for the VRP-MVD model.

Scenario	$\|C'\|$	z^{ALNS}	μ^{ALNS}	σ^{ALNS}	$\#C'$	$\#V$	z^{in}	$SI(\%)$
6.5.1	5	1.009	1.015	0.011	3.8	1	1.332	24.25
6.5.2	6	0.616	0.671	0.056	5	1	1.044	40.95
6.5.3	5	0.836	0.850	0.023	5	1	1.137	26.47
6.5.4	5	0.790	0.801	0.021	4.8	1	0.953	17.10
6.10.1	5	1.653	1.653	0.000	4	1	2.758	40.07
6.10.2	6	0.942	0.942	0.000	5	1	1.209	22.10
6.10.3	6	0.565	0.689	0.249	4.8	1	1.873	69.84
6.10.4	6	0.837	1.012	0.186	4.9	1	1.642	49.02
6.20.1	6	3.052	3.115	0.156	3.8	1.7	3.670	16.84
6.20.2	5	4.384	5.279	0.304	3.6	1.5	5.506	20.37
6.20.3	6	4.927	4.972	0.057	3	1	5.052	2.47
6.20.4	6	4.548	4.548	0.000	3	1	4.548	0.00
10.5.1	5	1.593	1.635	0.014	3.9	1	1.646	3.20
10.5.2	9	0.785	0.896	0.224	8.1	1	1.505	47.87
10.5.3	8	1.104	1.137	0.047	7	1	1.863	40.74
10.5.4	9	0.836	1.067	0.283	7.9	1	1.832	54.35
10.10.1	8	1.621	1.695	0.157	7	1	2.674	39.39
10.10.2	8	1.977	2.735	0.263	5.9	1	3.485	43.29
10.10.3	7	1.861	2.404	0.575	6	1.2	3.534	47.33
10.10.4	9	1.766	1.828	0.169	8.8	1	2.871	38.49
10.20.1	7	4.345	4.482	0.115	5.6	1.6	5.276	17.66
10.20.2	8	5.483	5.802	0.292	6.1	1.5	6.784	19.18
10.20.3	9	3.065	3.773	0.626	7.2	1	5.165	40.67
10.20.4	7	5.606	6.225	0.293	4.5	1.3	6.707	16.41
12.5.1	9	1.028	1.082	0.150	8.3	1	1.530	32.79
12.5.2	12	0.418	0.522	0.125	10	1	1.551	73.03
12.5.3	10	0.855	1.282	0.142	9.1	1	1.629	47.52
12.5.4	10	1.123	1.184	0.134	9.1	1	1.709	34.30
12.10.1	10	0.984	2.238	0.475	9	1.2	3.723	73.58
12.10.2	10	1.677	2.216	0.184	7.8	1	3.658	54.16
12.10.3	9	2.593	2.676	0.116	7.1	1	3.711	30.14
12.10.4	10	1.908	2.018	0.183	8.5	1	3.167	39.74
12.20.1	11	4.335	4.829	0.165	7.1	1	6.958	37.69
12.20.2	10	8.142	8.142	0.000	7	1	8.262	1.45
12.20.3	9	3.538	3.921	0.305	7.3	1.2	5.616	37.01
12.20.4	11	6.029	6.394	0.241	6.5	1.3	7.715	21.85
20.5.1	15	1.528	1.600	0.109	13.3	1	2.015	24.14
20.5.2	14	1.748	1.813	0.062	9.5	1	1.888	7.40
20.5.3	19	0.871	1.103	0.310	14.9	1	1.887	53.82
20.5.4	18	0.912	0.994	0.113	15.9	1	1.724	47.11
20.10.1	17	2.353	2.748	0.408	13.9	1	3.778	37.73
20.10.2	19	1.991	2.497	0.561	14	1	4.536	56.10
20.10.3	19	2.333	2.934	0.491	15.2	1	4.453	47.61

(continued on next page)

Table 4 – continued from previous page

Scenario	$\|C'\|$	z^{ALNS}	μ^{ALNS}	σ^{ALNS}	$\#C'$	$\#V$	z^{in}	$SI(\%)$
20.10.4	15	2.488	2.557	0.056	13.9	1	4.365	43.00
20.20.1	19	5.629	6.559	0.640	12.2	1	8.732	35.53
20.20.2	16	5.785	7.314	0.689	12.7	1.5	8.978	35.57
20.20.3	18	5.809	6.481	0.815	12.9	1.1	8.245	29.55
20.20.4	17	6.018	6.977	0.534	12	1.1	8.604	30.05
50.10.1	37	5.892	6.022	0.065	17.6	1	6.054	2.68
50.10.2	41	5.703	5.780	0.059	31.1	1	6.395	10.83
50.10.3	44	4.505	5.207	0.812	34.2	1	7.436	39.41
50.10.4	44	4.517	4.869	0.426	35.1	1	6.751	33.08
50.20.1	41	8.639	10.383	0.770	29	1	13.745	37.15
50.20.2	44	9.454	10.280	0.945	31.2	1	12.582	24.86
50.20.3	44	9.171	10.679	0.911	27.1	1	15.703	41.60
50.20.4	46	8.868	9.529	0.994	34.3	1	12.839	30.93
50.30.1	40	14.991	15.606	0.417	30.9	1	19.644	23.69
50.30.2	39	15.589	17.601	1.004	26.9	1	21.571	27.73
50.30.3	43	15.646	17.304	0.718	31.1	1	21.096	25.83
50.30.4	40	18.820	20.113	0.774	28	1	22.909	17.85
50.40.1	46	22.543	22.946	0.328	28	1	24.804	9.12
50.40.2	41	20.838	21.916	0.649	28.2	1.3	27.170	23.31
50.40.3	42	21.806	23.181	1.103	29.4	1.1	28.305	22.96
50.40.4	41	23.610	26.201	1.476	26.4	1.3	28.636	17.55
100.10.1	89	8.546	8.735	0.106	36.6	1	8.822	3.13
100.10.2	89	7.253	9.169	1.254	39.2	1	10.185	28.78
100.10.3	91	8.480	8.605	0.043	32.3	1	8.624	1.67
100.10.4	82	9.711	9.758	0.020	24	1	9.768	0.58
100.20.1	90	14.018	18.055	1.357	30.2	1	18.571	24.52
100.20.2	89	16.662	16.662	0.000	40	1	16.662	0.00
100.20.3	87	16.893	17.533	0.384	30.6	1	17.911	5.68
100.20.4	89	18.265	18.536	0.100	34.8	1	18.582	1.71
100.30.1	84	22.207	25.637	2.383	48.6	1	30.996	28.36
100.30.2	87	21.384	22.176	1.203	66.6	1	29.105	26.53
100.30.3	91	22.949	24.167	1.123	65.8	1	30.482	24.71
100.30.4	88	23.039	25.728	1.434	32.9	1	26.747	13.87
100.40.1	85	28.243	32.557	3.925	63.5	1.5	40.911	30.96
100.40.2	85	33.226	40.443	2.502	28.7	2	41.754	20.42
100.40.3	89	29.899	33.142	3.347	55.5	1.4	40.734	26.60
100.40.4	87	31.221	34.158	2.327	66.3	1.7	39.528	21.02
150.10.1	125	10.099	12.020	0.642	49.5	1	12.257	17.60
150.10.2	126	8.616	10.817	1.332	54.1	1	11.687	26.28
150.10.3	141	9.544	11.932	0.820	54.3	1	12.284	22.31
150.10.4	120	9.302	11.952	1.130	38.6	1	12.602	26.18
150.20.1	131	18.561	21.079	0.882	47.3	1	21.505	13.69
150.20.2	131	17.922	23.017	2.098	50.2	1	24.114	25.68
150.20.3	128	22.925	22.925	0.000	57	1	22.925	0.00
150.20.4	130	18.159	23.272	2.966	50	1	25.414	28.54
150.30.1	130	31.047	32.156	0.541	45.2	1.4	32.664	4.95

(continued on next page)

Table 4 – continued from previous page

Scenario	$\|C'\|$	z^{ALNS}	μ^{ALNS}	σ^{ALNS}	$\#C'$	$\#V$	z^{in}	$SI(\%)$
150.30.2	125	28.880	36.908	2.701	43.4	2	38.002	24.00
150.30.3	130	27.102	32.749	2.755	46.7	2	34.441	21.31
150.30.4	125	27.819	33.020	2.413	46.9	2	34.263	18.81
150.40.1	137	43.086	43.086	0.000	45	2	43.086	0.00
150.40.2	124	38.749	47.638	5.368	62.5	2	51.763	25.14
150.40.3	125	43.113	44.488	0.725	45	2	45.124	4.46
150.40.4	129	36.769	46.619	3.283	48	2	47.717	22.94
200.10.1	173	12.521	12.521	0.000	66	1	12.521	0.00
200.10.2	173	13.008	13.239	0.140	60.8	1	13.360	2.64
200.10.3	177	12.419	12.848	0.322	46.3	1	13.191	5.85
200.10.4	176	11.227	13.285	1.001	65.5	1	13.827	18.80
200.20.1	178	25.551	25.858	0.183	69.8	1	26.033	1.85
200.20.2	168	22.527	26.787	1.510	52	2	27.822	19.03
200.20.3	176	21.525	25.480	1.929	57.1	1	26.448	18.61
200.20.4	172	23.976	24.686	0.568	62	1	25.228	4.96
200.30.1	171	38.017	38.892	0.292	56.9	2	38.989	2.49
200.30.2	176	39.168	39.168	0.000	65	2	39.168	0.00
200.30.3	171	40.554	40.933	0.184	58.2	2	41.076	1.27
200.30.4	172	33.230	39.461	2.119	62.1	2	40.431	17.81
200.40.1	172	44.472	54.863	5.218	58.7	2.6	57.852	23.13
200.40.2	178	44.902	52.962	3.348	59.9	2	54.967	16.94
200.40.3	165	45.333	56.336	5.359	57.9	2	59.059	23.24
200.40.4	178	50.135	51.348	0.585	67.5	2	51.793	3.20

C Description of programming files

To obtain these results, we have implemented the ALNS metaheuristic in Java for both the VRP-D and the VRP-MVD. In total, their are 8 files used for this purpose. The first 4 files are used for the VRP-D and the last 4 are used for the VRP-MVD. First, we created "Thesis_initial_solution", in which we wrote methods for the nearest neighborhood algorithm, the improvement with the use of relocation moves, the `FindSortie` method, the inclusion of drone service and the heuristic that uses 2-opt and string relocation moves. Next, we have "Thesis_destroy_methods", where both the random destroy method and the cluster destroy method are implemented. The four different repair methods are written in "Thesis_repair_methods" and all of them use a destroy method from "Thesis_destroy_methods" as input. "Thesis_ALNS_algorithm" uses all three aforementioned classes in order to implement and run the ALNS metaheuristic 10 times for a given instance. For the VRP-MVD, we use the same methods as for VRP-D, except now they are adjusted such that a drone is allowed to deliver multiple packages within one trip. The classes used for this are called "Extension_initial_solution", "Extension_destroy_methods", "Extension_repair_methods" and "Extension_ALNS_algorithm".